```
 1: ******************************************************************************
 2:  Constructor Func and Proc List of maXbox 4.7.6. codeX signed
 3: ******************************************************************************
 4:
 5: ////////////////////////////////////////////////////////////////////////////
 6: ref Help Extract Functions of maXbox4.exe BigBitBox API HEX in BOX47
 7: ---------------------------------------------------------------------------
 8: File EXE:32.2 MB(33,5,592) V4.7.6.20 Oct.2022 EKON26/BASTA/JAX/IBZ/SWS/PASCON
 9: *****************Now the Funclist*************************************************
10: Funclist Func : 21314_; Source Compiled: 2'828'681 lines
11: *****************Now the Proclist*************************************************
12: Proclist Proc Size is: 50_
13: *****************Now Constructors*************************************************
14: Constructlist Constructor Size is: 1994_
15: def head: max: maXbox10: 17/10/2022 14:34:42
16: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist476_.txt
17: doc file: maxbox_extract_funclist476.txt(sort function)maxbox_functions_sortlist.txt
18: all unit file: http://www.softwareschule.ch/examples/packages.txt
19: all unit file2:http://www.softwareschule.ch/unitlist.htm
20: ---------------------------------------------------------------------------
21: Funclist total Size all is: 35958! Constructor, Func and Procedure
22: AExtraxt of EXE Functions of maXbox4.exe, locs of file = 39340
23: ASize of EXE: 33,805,592 bytes  CRC32: 9E995FA2
24: SHA1: of maXbox4.exe (4.7.6.20) D4619B18E231839334AB0FE0B90D4018DC6276A7
25: SHA- 4f4219830b728537ef6c4ddec92c2fbdfd8f6658a2f03c3981fdfcc833ae8c1e
26:
27: https://www.virustotal.
    com/gui/file/4f4219830b728537ef6c4ddec92c2fbdfd8f6658a2f03c3981fdfcc833ae8c1e/detection
28: https://www.virustotal.com/gui/file/c6e8c0861b0a05cb26ebdafe6985983eb8f6b4d92fb99ad2fa62d2ba50bed/detection
29: https://www.virustotal.com/gui/file/5d231d7b91a8e1d79ee2b1f19279a0dd2685c7c99974b0da9df749347fc58/detection
30: https://www.virustotal.
    com/gui/file/09eb95c4cc3b7fe60772405b170af4705085da89d190213228c65313c8733499/details
31: https://www.virustotal.
    com/gui/file/e35b4b6cde46d1c0c8a0c30bc125d76d6738e9d0541523214c12d0aa1ba91ca9/details
32: https://www.virustotal.
    com/gui/file/1a2ed2b64f9b7d68e1a16158d06742eef1cde94864a50ce29f08bfadaf402a44/detection
33: https://www.virustotal.com/gui/file/ec6aeb784bdf2dec91834378ef46be698cc0d0eddad9e6488c7fcafac7728/detection
34: https://www.virustotal.
    com/gui/file/9b4ce8468d70b11a0b111009ccf9b4a9e1daf6f9ec53680bf301256e5f69518a/detection
35: https://www.virustotal.
    com/gui/file/db0c72990863b4a08341410b51d76a00632519c0f1df9cc4d63a40c28f2c9778/detection
36: https://www.virustotal.com/gui/file/50710adac0234d55f80f0c211533f3490f72a4a7986b261e4672e56b589a2?nocache=1
37: https://www.virustotal.com/gui/file/52a07df33308c2ded182c50194a4c731f9921461ba1cf26625005ec0db672/detection
38: https://www.virustotal.
    com/gui/file/e99be9470f26f69169cc81dfbd8984c7a27dd3fb4b16941cb4028e5dbba66957/detection
39: https://www.virustotal.
    com/gui/file/9bb02566c31d45225d0f09f311b72365753dd48d5d7edd92823a4b1c3e24c001/detection
40: ////////////////////////////////////////////////////////////////////////////
41:
42: Func Metric of Script: 256_findfunctions476_of_EXE_80_1.txt
43: Func *************Now the Funclist*****************
44: Func  GetResStringChecked(Ident:Str; const Args: array of const):str
45: Func ( Index : Longint) : Integer
46: Func (Command: Word; Data: Longint; var CallHelp:Bool):Bool
47: Func _CheckAutoResult( ResultCode : HResult) : HResult
48: Func _T(Name: tbtString): Variant;
49: Func ABNFToText(const AText :Str) :Str
50: Func Abs(e : Extended) : Extended;
51: Func AbsInt(e : Extended) : Integer;
52: Func AbsInt( const B : integer) : integer;
53: Func AbsFloat( const B : double) : extended;
54: Func Ackermann( const A, B : Integer) : Integer
55: Func AcquireLayoutLock :Bool
56: Func ActionByName( const AName :Str) : TWebActionItem
57: Func ACTIVEBUFFER : PCHAR
58: Func Add:TAggregate
59: Func Add:TCollectionItem
60: Func Add:TColumn
61: Func Add:TComboExItem
62: Func Add:TCookie
63: Func Add:TCoolBand
64: Func Add:TFavoriteLinkItem
65: Func Add:TFileTypeItem
66: Func Add:THeaderSection
67: Func Add:THTMLTableColumn
68: Func Add:TIdEMailAddressItem
69: Func Add:TIdMessagePart
70: Func Add:TIdUserAccount
71: Func Add:TListColumn
72: Func Add:TListItem
73: Func Add:TStatusPanel
74: Func Add:TTaskDialogBaseButtonItem
75: Func Add:TWebActionItem
76: Func Add:TWorkArea
77: Func Add(AClass : TClass) : Integer
78: Func Add(AComponent : TComponent) : Integer
79: Func Add(AItem,AData : Integer) : Integer
80: Func Add(AItem,AData : Pointer) : Pointer
81: Func Add(AItem,AData : TObject) : TObject
```

```
 82: Func Add(AObject : TObject) : Integer
 83: Func Add(const Access, Count :Card; const Offset : Int64) : Integer
 84: Func Add(const S : WideString) : Int
 85: Func Add(Image, Mask : TBitmap) : Int
 86: Func Add(Index : LongInt; const Text :Str) : LongInt
 87: Func Add(Sibling : TTreeNode; const S :Str) : TTreeNode
 88: Func Add(const S:Str): Int
 89: Func Add(S:Str): Int;
 90: Func AddAt(const Access,Count:Card;const Offset:Int64;const Address:Pointer):Int
 91: Func ADDCHILD : TFIELDDEF
 92: Func AddChild( Index : LongInt; const Text :Str) : LongInt
 93: Func AddChild( Parent : TTreeNode; const S :Str) : TTreeNode
 94: Func AddChildFirst( Parent : TTreeNode; const S :Str) : TTreeNode
 95: Func AddChildObject(Index:LongInt;const Text:str;const Data:Pointer:LongInt
 96: Func AddChildObject( Parent:TTreeNode;const S:str;Ptr:Pointer) : TTreeNode
 97: Func AddChildObjectFirst(Parent:TTreeNode;const S:str;Ptr:Pointer):TTreeNode
 98: Func ADDFIELDDEF : TFIELDDEF
 99: Func AddFileExtIfNecessary( AFileName, AExt :Str) :Str
100: Func AddFirst( Sibling : TTreeNode; const S :Str) : TTreeNode
101: Func AddIcon( Image : TIcon) : Int
102: Func AddImage( Value : TCustomImageList; Index : Int) : Int
103: Func ADDINDEXDEF : TINDEXDEF
104: Func AddItem(const Caption:str;const ImageIdx,SelectImageIdx,OverlayImagIdx,
     Indent:Int;Data:Ptr):TComboExItem
105: Func AddItem( Item : THeaderSection; Index : Int) : THeaderSection
106: Func AddItem( Item : TListItem; Index : Int) : TListItem
107: Func AddItem( Item : TStatusPanel; Index : Int) : TStatusPanel
108: Func AddMapping( const FieldName :Str) :Bool
109: Func AddMasked( Image : TBitmap; MaskColor : TColor) : Int
110: Func AddModuleClass( AClass : TComponentClass) : TComponent
111: Func AddModuleName( const AClass :Str) : TComponent
112: Func AddNode(Node,Relative:TTreeNode;const S:str;Ptr:Pointer;Method:TNodeAttachMode):TTreeNode
113: Func AddObject( const S : WideString; AObject : TObject) : Int
114: Func AddObject(Index:LongInt;const Text:str; const Data:Pointer): LongInt
115: Func AddObject( Sibling : TTreeNode; const S :Str;Ptr: Pointer): TTreeNode
116: Func AddObject(S:str;AObject:TObject):Int
117: Func AddObjectFirst(Sibling:TTreeNode;const S string;Ptr:Pointer):TTreeNode
118: Func AddParameter : TParameter
119: Func AddParamSQLForDetail(Params:TParams;SQL:WideStr;Native:Bool;QuoteChar:WideString):WideString
120: Func Addr64ToAddr32(const Value: TJclAddr64): TJclAddr32;
121: Func Addr32ToAddr64(const Value: TJclAddr32): TJclAddr64;
122: Func AdjustLineBreaksS(const S:Str):Str) TTextLineBreakStyle', '(tlbsLF, tlbsCRLF)
123: Func AdjustLineBreaks(const S:Str; Style: TTextLineBreakStyle):Str;
124: Func AllData :Str
125: Func AllocMemCount: Int;
126: Func AllocMemSize: Int;
127: Func AllocPatternBitmap( BkColor, FgColor : TColor) : TBitmap
128: Func AllowRegKeyForEveryone( Key : HKEY; Path :Str) :Bool
129: Func AlphaComponent( const Color32 : TColor32) : Int
130: Func AlphaSort :Bool
131: Func AlphaSort( ARecurse :Bool) :Bool
132: Func AnsiCat( const x, y : Ansistr) : Ansistr
133: Func AnsiCompareFileName( S1, S2 :Str) : Int
134: Func AnsiCompareFileName(const S1:Str; const S2:Str): Int)
135: Func AnsiCompareStr( S1, S2 :Str) : Int
136: Func AnsiCompareStr(const S1:Str; const S2:Str): Int;)
137: Func AnsiCompareText( S1, S2 :Str) : Int
138: Func AnsiCompareText(const S1:Str; const S2:Str): Int;)
139: Func AnsiContainsStr( const AText, ASubText :Str) :Bool
140: Func AnsiContainsText( const AText, ASubText :Str) :Bool
141: Func AnsiCopy( const src : Ansistr; index, count : Int) : Ansistr
142: Func AnsiDequotedStr( S :Str; AQuote : Char) :Str
143: Func AnsiEndsStr( const ASubText, AText :Str) :Bool
144: Func AnsiEndsText( const ASubText, AText :Str) :Bool
145: Func AnsiExtractQuotedStr( var Src : PChar; Quote : Char) :Str
146: Func AnsiExtractQuotedStr(var Src: PChar; Quote: Char):Str)
147: Func AnsiIndexStr( const AText :Str; const AValues : array of string) : Int
148: Func AnsiIndexText(const AText:str; const AValues:array of string):Int
149: Func AnsiLastChar( S :Str) : PChar
150: Func AnsiLastChar(const S:Str): PChar)
151: Func AnsiLeftStr( const AText : Ansistr; const ACount : Int) : Ansistr
152: Func AnsiLowerCase( S :Str) :Str
153: Func AnsiLowercase(s :Str) :Str;
154: Func AnsiLowerCaseFileName( S :Str) :Str
155: Func AnsiMatchStr( const AText:str; const AValues:array of string):Bool
156: Func AnsiMatchText(const AText:str; const AValues: array of string):Bool
157: Func AnsiMidStr(const AText: Ansistr; const AStart, ACount: Int): Ansistr
158: Func AnsiPos( const src, sub : Ansistr) : Int
159: Func AnsiPos( Substr, S :Str) : Int
160: Func AnsiPos(const Substr:Str; const S:Str): Int;)
161: Func AnsiQuotedStr( S :Str; Quote : Char) :Str
162: Func AnsiReplaceStr( const AText, AFromText, AToText :Str) :Str
163: Func AnsiReplaceText( const AText, AFromText, AToText :Str) :Str
164: Func AnsiResemblesText( const AText, AOther :Str) :Bool
165: Func AnsiReverseString( const AText : Ansistr) : Ansistr
166: Func AnsiRightStr( const AText : Ansistr; const ACount : Int) : Ansistr
167: Func AnsiSameCaption(const Text1:Str; const Text2:Str):Bool)
168: Func AnsiSameStr( S1, S2 :Str) :Bool
169: Func AnsiSameStr(const S1:Str; const S2:Str):Bool)
```

```
170: Func AnsiSameText( const S1, S2 :Str) :Bool
171: Func AnsiSameText( S1, S2 :Str) :Bool
172: Func AnsiSameText(const S1:Str; const S2:Str):Bool)
173: Func AnsiStartsStr( const ASubText, AText :Str) :Bool
174: Func AnsiStartsText( const ASubText, AText :Str) :Bool
175: Func AnsiStrComp( S1, S2 : PChar) : Int
176: Func AnsiStrComp(S1: PChar; S2: PChar): Int)
177: Func AnsiStrIComp( S1, S2 : PChar) : Int
178: Func AnsiStrIComp(S1: PChar; S2: PChar): Int)
179: Func AnsiStrLastChar( P : PChar) : PChar
180: Func AnsiStrLastChar(P: PChar): PChar)
181: Func AnsiStrLComp( S1, S2 : PChar; MaxLen :Card) : Int
182: Func AnsiStrLIComp( S1, S2 : PChar; MaxLen :Card) : Int
183: Func AnsiStrLower( Str : PChar) : PChar
184: Func AnsiStrPos( Str, SubStr : PChar) : PChar
185: Func AnsiStrPos(Str: PChar; SubStr: PChar): PChar)
186: Func AnsiStrScan(Str: PChar; Chr: Char): PChar)
187: Func AnsiStrUpper( Str : PChar) : PChar
188: Func AnsiToUtf8( const S :Str) : UTF8String
189: Func AnsiToUtf8Ex( const S :Str; const cp : Int) : UTF8String
190: Func AnsiUpperCase( S :Str) :Str
191: Func AnsiUppercase(s :Str) :Str;
192: Func AnsiUpperCaseFileName( S :Str) :Str
193: Func ApplyUpdates(const Delta:OleVariant;MaxErrors:Int;out ErrorCount:Int): OleVariant
194: Func ApplyUpdates(const Delta:OleVariant;MaxErrors: Int;out ErrorCount:Int): OleVariant;
195: Func ApplyUpdates( MaxErrors : Int) : Int
196: Func ApplyUpdates1(const Delta:OleVar;MaxErrs:Int;out ErrCount:Int;var OwnerData:OleVar):OleVariant;
197: Func ArcCos(const X : Extended) : Extended
198: Func ArcCosh(const X : Extended) : Extended
199: Func ArcCot(const X : Extended) : Extended
200: Func ArcCotH(const X : Extended) : Extended
201: Func ArcCsc(const X : Extended) : Extended
202: Func ArcCscH(const X : Extended) : Extended
203: Func ArcSec(const X : Extended) : Extended
204: Func ArcSecH(const X : Extended) : Extended
205: Func ArcSin(const X : Extended) : Extended
206: Func ArcSinh(const X : Extended) : Extended
207: Func ArcTan(const X : Extended) : Extended
208: Func ArcTan2(const Y, X : Extended) : Extended
209: Func ArithmeticMean(const X : TDynDoubleArray) : Float
210: Func ArrayLength: Int;
211: Func AsHex( const AValue : T4x4LongWordRecord):Str
212: Func AsHex( const AValue : T5x4LongWordRecord):Str
213: Func ASNDecLen( var Start : Int; const Buffer :Str) : Int
214: Func ASNDecOIDItem( var Start : Int; const Buffer :Str) : Int
215: Func ASNEncInt( Value : Int) :Str
216: Func ASNEncLen( Len : Int) :Str
217: Func ASNEncOIDItem( Value : Int) :Str
218: Func ASNEncUInt( Value : Int) :Str
219: Func ASNItem(var Start: Int; const Buffer:str; var ValueType:Int):str
220: Func ASNObject( const Data :Str; ASNType : Int) :Str
221: Func Assigned(I: Longint):Bool;
222: Func AspectRatio(aWidth, aHeight: Int):Str;
223: Func AsWideString( Field : TField) : WideString
224: Func AtLeast( ACount : Int) :Bool
225: Func AttemptToUseSharedMemoryManager :Bool
226: Func Authenticate :Bool
227: Func AuthenticateUser( const AUsername, APassword :Str) :Bool
228: Func Authentication :Str
229: Func BatchMove( ASource : TBDEDataSet; AMode : TBatchMode) : Longint
230: Func BcdCompare( const bcd1, bcd2 : TBcd) : Int
231: Func BcdFromBytes( const AValue : TBytes) : TBcd
232: Func BcdPrecision( const Bcd : TBcd) : Word
233: Func BcdScale( const Bcd : TBcd) : Word
234: Func BcdToBytes( const Value : TBcd) : TBytes
235: Func BCDToCurr( const BCD : TBcd; var Curr : Currency) :Bool
236: Func BcdToDouble( const Bcd : TBcd) : Double
237: Func BcdToInt( const Bcd : TBcd; Truncate :Bool) : Int
238: Func BcdToStr( const Bcd : TBcd) :Str;
239: Func BcdToStrF(const Bcd:TBcd;Format:TFloatFormat;const Precision,Digits:Int):str
240: Func beep2(dwFreq, dwDuration: Int):Bool;
241: Func BeginPeriod( const Period :Card) :Bool
242: Func BeginTrans : Int
243: Func BeginTransaction : TDBXTransaction;
244: Func BeginTransaction1( Isolation : TDBXIsolation) : TDBXTransaction;
245: Func BigMulu(aone, atwo:Str):Str;
246: Func BigNumber(aone, atwo:Str):Str;
247: Func BigExp(aone,atwo:Str):Str;
248: Func BigMul(aone,atwo:Str):Str;
249: Func BigAdd(aone,atwo:Str):Str;
250: Func BigSub(aone,atwo:Str):Str;
251: Func BigFactorial(aone:Str):Str;
252: Func BinaryToDouble( ABinary :Str; DefValue : Double): Double
253: Func BinomialCoeff( N, R :Card) : Float
254: Func BinominalCoefficient(n, k: Int):Str;
255: Func BinStrToInt( const ABinary :Str) : Int
256: Func BinToByte(Binary:Str): Byte;
257: Func BinToHex2(Binary:Str):Str;
258: Func BinToInt(Binary:Str): Int;
```

```
259: Func BinToChar(St:Str): Char;
260: Func BinToStr(ans:Str):Str;
261: Func BitBlt(hdcDest:HDC;nXDest,nYDest,nWidth,nHeigh:Int;hdcSrc:HDC;nXSrc,nYSrc:Int;dwRop:DWORD):Bool;
262: Func BitmapsAreIdentical( ABitmap1, ABitmap2 : TBitmap) :Bool
263: Func BitsHighest( X: Byte) : Int;
264: Func BitsHighest1(X: ShortInt) : Int;
265: Func BitsHighest2(X: SmallInt) : Int;
266: Func BitsHighest3(X: Word) : Int;
267: Func BitsHighest4(X: Int) : Int;
268: Func BitsHighest5(X:Card) : Int;
269: Func BitsHighest6(X: Int64) : Int;
270: Func BitsLowest( X: Byte) : Int;
271: Func BitsLowest1(X: Shortint) : Int;
272: Func BitsLowest2(X: Smallint) : Int;
273: Func BitsLowest3(X: Word) : Int;
274: Func BitsLowest4(X:Card) : Int;
275: Func BitsLowest5(X: Int) : Int;
276: Func BitsLowest6(X: Int64) : Int;
277: Func BitsNeeded( const X : Byte) : Int;
278: Func BitsNeeded1(const X : Word) : Int;
279: Func BitsNeeded2(const X : Int) : Int;
280: Func BitsNeeded3(const X : Int64) : Int;
281: Func BlueComponent(const Color32 : TColor32) : Int
282: Func BooleanToInt( const Pb :Bool) : Int
283: Func BoolToStr(B:Bool; UseBoolStrs:Bool):Str)
284: Func BoolToStr1(value :Bool) :Str;
285: Func booltoint( aBool :Bool) : LongInt
286: Func inttobool( aInt : LongInt) :Bool
287: Func Bounds( ALeft, ATop, AWidth, AHeight : Int) : TRect
288: Func Bounds(ALeft, ATop, AWidth, AHeight: Int): TRect)
289: Func BreakApart(BaseString,BreakString:str;StringList: TStrings): TStrings
290: Func BrightColor( const Color : TColor; const Pct : Single) : TColor
291: Func BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
292: Func BufferRequest( Length : Int) : TStream
293: Func BuildFileList(const Path:Str;const Attr:Int;const List:TStrings):Bool
294: Func Buttons: PTaskDialogButton
295: Func BytesPerScanline(PixelsPerScanline,BitsPerPixel,Alignment:Longint):Longint
296: Func BytesToCardinal( const AValue : TIdBytes; const AIndex : Int) :Card
297: Func BytesToChar( const AValue : TIdBytes; const AIndex : Int) : Char
298: Func BytesToInt64( const AValue : TIdBytes; const AIndex : Int) : Int64
299: Func BytesToInt( const AValue : TIdBytes; const AIndex : Int) : Int
300: Func BytesToIPv6( const AValue : TIdBytes; const AIndex : Int) : TIdIPv6Address
301: Func BytesToShort( const AValue : TIdBytes; const AIndex : Int) : Short
302: Func BytesToString(ABytes:TIdBytes; AStartIndex:Int; AMaxCount:Int):Str;
303: Func BytesToStr(const Value: TBytes):Str;
304: Func BytesToWord( const AValue : TIdBytes; const AIndex : Int) : Word
305: Func ByteToBin(Int: Byte):Str;
306: Func ByteToCharIndex( S :Str; Index : Int) : Int
307: Func ByteToCharIndex(const S:Str; Index: Int): Int)
308: Func ByteToCharLen( S :Str; MaxLen : Int) : Int
309: Func ByteToCharLen(const S:Str; MaxLen: Int): Int)
310: Func ByteToHex( const AByte : Byte) :Str
311: Func ByteToOctal( const AByte : Byte) :Str
312: Func ByteType( S :Str; Index : Int) : TMbcsByteType
313: Func ByteType(const S:Str; Index: Int): TMbcsByteType)
314: Func CalcTitleRect( Col : TColumn; ARow : Int; var MasterCol:TColumn):TRect
315: Func CalculateDFAFingerprint( oStates : TList) : Int
316: Func CallTerminateProcs:Bool)
317: Func CANFOCUS:BOOLEAN
318: Func CanLoad( const Ext :Str) :Bool
319: Func CanParse( AWebRequest : TWebRequest) :Bool
320: Func CanSave( const Ext :Str) :Bool
321: Func CanStart( cChar : char) :Bool
322: Func CaptureScreen : TBitmap;
323: Func CaptureScreen1( Rec : TRect) : TBitmap;
324: Func CardinalToFourChar( ACardinal : LongWord) :Str
325: Func CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Bool):Bool
326: Func CastSoapToVariant1(SoapInfo:PTypeInfo;const SoapData:WideString):Variant;
327: Func Ceil( const X : Extended) : Int
328: Func Ceil16( X : Int) : Int
329: Func Ceil4( X : Int) : Int
330: Func Ceil8( X : Int) : Int
331: Func Ceiling( const X : Extended) : Int
332: Func CellRect( ACol, ARow : Longint) : TRect
333: Func CelsiusToFahrenheit( const AValue : Double) : Double
334: Func CenterPoint( const Rect : TRect) : TPoint
335: Func CenterPoint(const Rect: TRect): TPoint)
336: Func ChangeFileExt( FileName, Extension :Str) :Str
337: Func ChangeFileExt(const FileName:Str; const Extension:Str):Str)
338: Func CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) :Bool
339: Func CharInSet( const Ch : Char; const testSet: TSysCharSet):Bool
340: Func CharIsInEOF( const AString :Str; ACharPos : Int) :Bool
341: Func CharIsInSet(const AString:str;const ACharPos:Int;const ASet:str):Bool
342: Func CharLength( S :Str; Index : Int) : Int
343: Func CharRange( const AMin, AMax : Char) :Str
344: Func CharsetToIdent(Charset: Longint; var Ident:Str):Bool)
345: Func CharToBin(vChr: Char):Str;
346: Func CharNext(lpsz: PChar): PChar; stdcall;
347: Func CharToByteIndex( S :Str; Index : Int) : Int
```

```
348: Func CharToByteIndex(const S:Str; Index: Int): Int)
349: Func CharToByteLen( S :Str; MaxLen : Int) : Int
350: Func CharToByteLen(const S:Str; MaxLen: Int): Int)
351: Func CharToHex(const APrefix :Str; const cc : Char) : shortstring;
352: Func CharToHexStr(Value: char):Str);
353: Func CharToOem(ins, outs: PChar):boolean;
354: Func CharToUniCode(Value: Char):Str;
355: Func CheckMenuDropdown :Bool
356: Func CheckMessages : longint
357: Func CheckBox:Str;
358: Func CheckOpen( Status : DBIResult) :Bool
359: Func CheckPassword( const APassword :Str) :Bool
360: Func CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt):SmallInt
361: Func CheckCrc32( var X : array of Byte; N : Int; Crc :Card) : Int;
362: Func CheckSynchronize(Timeout: Int):Bool
363: Func CheckWin32Version( AMajor : Int; AMinor : Int) :Bool
364: Func CheckCom(AComNumber: Int): Int;;
365: Func CheckLPT1:Str;;
366: Func ChrA(const a: byte): char;
367: Func ClassIDToProgID(const ClassID: TGUID):Str;
368: Func ClassNameIs(const Name:Str):Bool
369: Func ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
370: Func ClearBit1(const Value : Shortint; const Bit : TBitRange) : Shortint;
371: Func ClearBit2(const Value : Smallint; const Bit : TBitRange) : Smallint;
372: Func ClearBit3(const Value : Word; const Bit : TBitRange) : Word;
373: Func ClearBit4(const Value : Int; const Bit : TBitRange) : Int;
374: Func ClearBit5(const Value :Card; const Bit : TBitRange) :Card;
375: Func ClearBit6(const Value : Int64; const Bit : TBitRange) : Int64;
376: Func CLIENTTOSCREEN(POINT:TPOINT):TPOINT
377: Func Clipboard : TClipboard
378: Func ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
379: Func ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
380: Func ClipLine( var X1, Y1, X2, Y2 : Int; const ClipRect : TRect) :Bool;
381: Func ClipLineToRect(var P1,P2:TFloatPoint;const Rect:TFloatRect) :Bool
382: Func Clone( out stm : IStream) : HResult
383: Func CloneConnection : TSQLConnection
384: Func CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
385: Func CLOSEQUERY:BOOLEAN
386: Func CloseVolume( var Volume : THandle) :Bool
387: Func CloseHandle(Handle: Int): Int; stdcall;
388: Func CPlApplet(hwndCPl:THandle;uMsg:DWORD;lParam1,lParam2:Longint):Longint
389: Func CmdLine: PChar;
390: Func CmdShow: Int;
391: // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
392: Func CoordinateStr(Idx: Int; PosInSec: Double; PosLn: TPos):Str;
393: Func Color32(const R, G, B : Byte; const A : Byte) : TColor32;
394: Func Color32(WinColor : TColor) : TColor32;
395: Func Color321(const Index : Byte; const Palette : TPalette32) : TColor32;
396: Func ColorAdjustLuma( clrRGB : TColor; n : Int; fScale :BOOLean) : TColor
397: Func ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
398: Func ColorToHTML( const Color : TColor) :Str
399: Func ColorToIdent(Color: Longint; var Ident:Str):Bool)
400: Func ColorToRGB(color: TColor): Longint
401: Func ColorToString(Color: TColor):Str)
402: Func ColorToWebColorName( Color : TColor) :Str
403: Func ColorToWebColorStr( Color : TColor) :Str
404: Func ColumnAtDepth( Col : TColumn; ADepth : Int) : TColumn
405: Func Combination(npr, ncr: Int): extended;
406: Func CombinationInt(npr, ncr: Int): Int64;
407: Func CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
408: Func CommaAdd( const AStr1, AStr2 :Str) :Str
409: Func CommercialRound( const X : Extended) : Int64
410: Func Commit( grfCommitFlags : Longint) : HResult
411: Func Compare( const NameExt :Str) :Bool
412: Func CompareDate(const A, B: TDateTime): TValueRelationship;
413: Func CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Int
414: Func CompareFiles(const FN1,FN2:str;Breathe:TNotifyEvent;BreathingSender:TObject):bool
415: Func CompareMemoryStreams( S1, S2 : TMemoryStream) :Bool
416: Func CompareStr( S1, S2 :Str) : Int
417: Func CompareStr(const S1:Str; const S2:Str): Int)
418: Func CompareString(const S1:Str; const S2:Str): Int)
419: Func CompareText( S1, S2 :Str) : Int
420: Func CompareText(const S1:Str; const S2:Str): Int)
421: Func CompareTextLike(cWildStr,cStr:str;const cWildChar:char;lCaseSensitive:bool):bool
422: Func CompareTime(const A, B: TDateTime): TValueRelationship;
423: Func CompareValueE(const A,B:Extended;Epsilon:Extended= 0):TValueRelationship; overload;
424: Func CompareValueD(const A, B:Double; Epsilon:Double = 0):TValueRelationship; overload;
425: Func CompareValueS(const A, B: Single; Epsilon: Single = 0): TValueRelationship; overload;
426: Func CompareValueI(const A, B: Int): TValueRelationship; overload;
427: Func CompareValueI64(const A, B: Int64): TValueRelationship; overload;
428: Func CompatibleConversionType(const AType:TConvType; const AFamily : TConvFamily):Boolean
429: Func CompatibleConversionTypes( const AFrom, ATo : TConvType) :Bool
430: Func ComponentTypeToString( const ComponentType : DWORD) :Str
431: Func ComposeDateTime(Date,Time : TDateTime) : TDateTime;;
432: Func ComponentToStringProc(Component: TComponent):Str;
433: Func StringToComponentProc(Value:Str): TComponent;
434: Func CompToCurrency( Value : Comp) : Currency
435: Func CompToDouble( Value : Comp) : Double
436: Func ComputeFileCRC32(const FileName :Str) : Int;
```

```
437: Func ComputeSHA256(astr:Str; amode: char):Str)  //mode F:File, S:str
438: Func ComputeSHA512(astr:Str; amode: char):Str)  //mode F:File, S:str
439: Func ComPortSelect: Int;  // Search for the first available port
440: Func Concat(s:Str):Str
441: Func ConnectAndGetAll :Str
442: Func Connected :Bool
443: Func constrain(x, a, b: Int): Int;
444: Func ConstraintCallBack(Req:DsInfoReq;var ADataSources:DataSources): DBIResult
445: Func ConstraintsDisabled :Bool
446: Func CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
447: Func ContainsState( oState : TniRegularExpressionState) :Bool
448: Func ContainsStr( const AText, ASubText :Str) :Bool
449: Func ContainsText( const AText, ASubText :Str) :Bool
450: Func ContainsTransition(oTransition:TniRegularExpressionTransition):boolean
451: Func Content:Str
452: Func ContentFromStream( Stream : TStream) :Str
453: Func ContentFromString( const S :Str) :Str
454: Func CONTROLSDISABLED :Bool
455: Func Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
456: Func Convert1(const AValue:Double;const AFrom1,AFrom2,ATo1,ATo2:TConvType):Double;
457: Func ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
458: Func ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Int) : Int
459: Func ConvertTo( const AValue : Double; const ATo : TConvType) : Double
460: Func ConvertWriteStream( Stream : TStream; Buffer: PChar; BufSize : Int) : Int
461: Func ConvFamilyToDescription( const AFamily : TConvFamily) :Str
462: Func ConvTypeToDescription( const AType : TConvType) :Str
463: Func ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
464: Func ConvTypeToFamily( const AType : TConvType) : TConvFamily;
465: Func ConvAdd(const AVal:Dbl;const AType1:TConvType;const AVal2:Dbl;const AType2,
     AResultType:TConvType):Double
466: Func ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
     AType2:TConvType):TValueRelationship
467: Func ConvDec( const AValue:Double;const AType,AAmountType:TConvType):Double;
468: Func ConvDec1(const AValue:Dbl;const AType:TConvType;const AAmount:Dble;const
     AAmountType:TConvType):Double;
469: Func ConvDiff(const AVal1:Dbl;const AType1:TConvType;const AVal2:Dble;const AType2,
     AResuType:TConvType):Double
470: Func ConvInc( const AValue:Double; const AType,AAmountType:TConvType):Double;
471: Func ConvInc1(const AValue:Dbl;const AType:TConvType;const AAmount:Double;const
     AAmountType:TConvType):Double;
472: Func ConvSameValue(const AValue1:Dbl;const AType1:TConvType;const AValue2:Dbl;const AType2:TConvType):Bool
473: Func ConvToStr( const AValue : Double; const AType : TConvType) :Str
474: Func ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmount : Double; const
     AAmountType : TConvType) :Bool
475: Func ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType;const AAmount:Double;const
     AAmountType: TConvType) :Bool
476: Func Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
477: Func CopyFile( Source, Dest :Str; CanOverwrite :Bool) : Bool
478: Func CopyFileEx( Source, Dest :Str; Flags : FILEOP_FLAGS) : Bool
479: Func CopyFileTo( const Source, Destination :Str) :Bool
480: Func CopyFrom(Source:TStream;Count:Int64):LongInt
481: Func CopyPalette( Palette : HPALETTE) : HPALETTE
482: Func CopyTo( Length : Int) :Str
483: Func CopyTo(stm:IStream;cb:Largeint;out cbRead:Largeint;out cbWritten:Largeint):HResult
484: Func CopyToEOF :Str
485: Func CopyToEOL :Str
486: Func Cos(e : Extended) : Extended;
487: Func Cosecant( const X : Extended) : Extended
488: Func Cot( const X : Extended) : Extended
489: Func Cotan( const X : Extended) : Extended
490: Func CotH( const X : Extended) : Extended
491: Func Count : Int
492: Func CountBitsCleared( X:Byte) : Int;
493: Func CountBitsCleared1(X:Shortint) : Int;
494: Func CountBitsCleared2(X:Smallint) : Int;
495: Func CountBitsCleared3(X:Word) : Int;
496: Func CountBitsCleared4(X:Int) : Int;
497: Func CountBitsCleared5(X:Card) : Int;
498: Func CountBitsCleared6(X:Int64) : Int;
499: Func CountBitsSet( X:Byte) : Int;
500: Func CountBitsSet1(X:Word) : Int;
501: Func CountBitsSet2(X:Smallint) : Int;
502: Func CountBitsSet3(X:ShortInt) : Int;
503: Func CountBitsSet4(X:Int) : Int;
504: Func CountBitsSet5(X:Card) : Int;
505: Func CountBitsSet6(X:Int64) : Int;
506: Func countDirfiles(const apath:Str): Int;
507: Func CountGenerations(Ancestor,Descendent: TClass): Int
508: Func Coversine( X : Float) : Float
509: Func CRC32(const fileName:Str): LongWord;
510: Func CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
511: Func CreateColumns : TDBGridColumns
512: Func CreateDataLink : TGridDataLink
513: Func CreateDir( Dir :Str) :Bool
514: Func CreateDir(const Dir:Str):Bool)
515: Func CreateDOSProcessRedirected(const CommandLine,InputFile,OutputFile:str): Bool
516: Func CreateEnvironmentBlock(const Options:TEnvironmentOptions;const AdditionalVars:TStrings):PChar
517: Func CREATEFIELD(OWNER:TCOMPONENT;PARENTFIELD:TOBJECTFIELD;const FIELDNAME:str;CREATECHILDREN:BOOL):TFIELD
518: Func CreateGlobber( sFilespec :Str) : TniRegularExpression
```

```
519: Func CreateGrayMappedBmp( Handle : HBITMAP) : HBITMAP
520: Func CreateGrayMappedRes( Instance : THandle; ResName : PChar) : HBITMAP
521: Func CreateGUID(out Guid: TGUID): HResult)
522: Func CreateInstance( const unkOuter: IUnknown;const iid:TGUID; out obj):HResult
523: Func CreateMappedBmp(Handle : HBITMAP; const OldColors,NewColors:array of TColor):HBITMAP
524: Func CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
525: Func CreateMessageDialog(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons):TForm;
526: Func CreateMessageDialog1(const Msg:str;DlgType:TMsgDlgType;Btns:TMsgDlgBtns;DefaultBtn:TMsgDlgBtn):TForm;
527: Func CreateOleObject(const ClassName:Str): IDispatch;
528: Func CREATEPARAM(FLDTYPE:TFIELDTYPE;const PARAMNAME:str; PARAMTYPE:TPARAMTYPE): TPARAM
529: Func CreateParameter(const
     Name:WideString;DataType:TDataType;Direction:TParameterDirection;Size:Int;Value: OleVariant):TParameter
530: Func CreateLocate( DataSet : TDataSet) : TJvLocateObject
531: Func CreateMappedBmp(Handle: HBITMAP;const OldColors,NewColors:array of TColor) :HBITMAP
532: Func CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
533: Func CreateMutex2(lpMutexAttributes:TObject;bInitialOwner:BOOL;lpName:PChar): THandle;
534: Func CreateSemaphore2(lpSemaphoreAttributes:TObject;lInitialCount,
     lMaximumCount:Longint;lpName:PChar):THandle;
535:  //Ex.: vMutex := CreateMutex2(Nil, True, MutexName);
536: Func IPAddrToName // ShowMessage(IPAddrToName(LocalIp));
537: Func CreateRecordBuffer( Length : Int) : TRecordBuffer
538: Func CreateValueBuffer( Length : Int) : TValueBuffer
539: Func CreatePopupCalculator(AOwner:TComponent;ABiDiMode:TBiDiMode):TWinControl
540: Func CreateRecordBuffer( Length : Int) : TRecordBuffer
541: Func CreateRotatedFont( Font : TFont; Angle : Int) : HFONT
542: Func CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
543: Func CreateValueBuffer( Length : Int) : TValueBuffer
544: Func CreateHexDump( AOwner : TWinControl) : THexDump
545: Func Csc( const X : Extended) : Extended
546: Func CscH( const X : Extended) : Extended
547: Func currencyDecimals: Byte
548: Func currencyFormat: Byte
549: Func currencyString:Str
550: Func CurrentProcessId : TIdPID
551: Func CurrentReadBuffer :Str
552: Func CurrentThreadId : TIdPID
553: Func CurrentYear : Word
554: Func CurrToBCD(const Curr:Currency;var BCD:TBcd;Precision:Int;Decimals:Int):Bool
555: Func CurrToStr( Value : Currency) :Str;
556: Func CurrToStrF(Value:Currency;FormatSettings:TFormatSettings;Digits:Int):str;
557: Func CurrToStrFS(Value:Currency;Format:TFloatFormat;Digits:Int;const FormatSettings:TFormatSettings):str;
558: Func CursorToIdent(cursor: Longint; var Ident:Str):Bool;
559: Func CursorToString(cursor: TCursor):Str;
560: Func CustomSort( SortProc : TLVCompare; lParam : Longint) :Bool
561: Func CustomSort(SortProc:TTVCompare;Data:Longint;ARecurse :Bool):Boolean
562: Func CycleToDeg( const Cycles : Extended) : Extended
563: Func CycleToGrad( const Cycles : Extended) : Extended
564: Func CycleToRad( const Cycles : Extended) : Extended
565: Func D2H( N : Longint; A : Byte) :Str
566: Func DarkColor( const Color : TColor; const Pct : Single) : TColor
567: Func DarkColorChannel( const Channel: Byte; const Pct : Single) : Byte
568: Func DataLinkDir :Str
569: Func DataRequest( Data : OleVariant) : OleVariant
570: Func DataRequest( Input : OleVariant) : OleVariant
571: Func DataToRawColumn( ACol : Int) : Int
572: Func Date : TDateTime
573: Func Date: TDateTime;
574: Func DateIsNull( const pdtValue : TDateTime; const pdtKind : TdtKind):Boolean
575: Func DateOf( const AValue : TDateTime) : TDateTime
576: Func DateSeparator: char;
577: Func DateTimeGMTToHttpStr( const GMTValue : TDateTime) :Str
578: Func DateTimeToFileDate( DateTime : TDateTime) : Int
579: Func DateTimeToFileDate(DateTime: TDateTime): Int;
580: Func DateTimeToGmtOffSetStr( ADateTime : TDateTime; SubGMT :Bool):Str
581: Func DateTimeToInternetStr(const Value:TDateTime;const AIsGMT:Boolean) String
582: Func DateTimeToJulianDate( const AValue : TDateTime) : Double
583: Func DateTimeToModifiedJulianDate( const AValue : TDateTime) : Double
584: Func DateTimeToStr( DateTime : TDateTime) :Str;
585: Func DateTimeToStr2(DateTime: TDateTime;FormatSettings:TFormatSettings):str;
586: Func DateTimeToTimeStamp(DateTime: TDateTime): TTimeStamp
587: Func DateTimeToUnix( const AValue : TDateTime) : Int64
588: Func DateTimeToUnix(D: TDateTime): Int64;
589: Func DateToStr( DateTime : TDateTime) :Str;
590: Func DateToStr(const DateTime: TDateTime):Str;
591: Func DateToStr(D: TDateTime):Str;
592: Func DateToStr2(DateTime:TDateTime;FormatSettings:TFormatSettings):str;
593: Func DayOf( const AValue : TDateTime) : Word
594: Func DayOfTheMonth( const AValue : TDateTime) : Word
595: Func DayOfTheMonth(const AValue: TDateTime): Word;
596: Func DayOfTheWeek( const AValue : TDateTime) : Word
597: Func DayOfTheYear( const AValue : TDateTime) : Word
598: Func DayOfTheYear(const AValue: TDateTime): Word;
599: Func DayOfWeek( DateTime : TDateTime) : Word
600: Func DayOfWeek(const DateTime: TDateTime): Word;
601: Func DayOfWeekStr( DateTime : TDateTime) :Str
602: Func DaysBetween( const ANow, AThen : TDateTime) : Int
603: Func DaysInAMonth( const AYear, AMonth : Word) : Word
604: Func DaysInAYear( const AYear : Word) : Word
605: Func DaysInMonth( const AValue : TDateTime) : Word
```

```
606: Func DaysInYear( const AValue : TDateTime) : Word
607: Func DaySpan( const ANow, AThen : TDateTime) : Double
608: Func DBUseRightToLeftAlignment( AControl : TControl; AField : TField) :Bool
609: Func DecimalSeparator: char;
610: Func DecLimit( var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;
611: Func DecLimit1(var B:Shortint;const Limit:Shortint const Decr:Shortint):Shortint;
612: Func DecLimit2(var B:Smallint;const Limit:Smallint;const Decr:Smallint):Smallint;
613: Func DecLimit3( var B : Word; const Limit : Word; const Decr : Word) : Word;
614: Func DecLimit4( var B : Int; const Limit : Int; const Decr : Int) : Int;
615: Func DecLimit5(var B Cardinal;const Limit Cardinal;const Decr:Card):Card;
616: Func DecLimit6(var B:Int64;const Limit : Int64; const Decr : Int64) : Int64;
617: Func DecLimitClamp(var B:Byte; const Limit : Byte; const Decr : Byte) : Byte;
618: Func DecLimitClamp1(var B:Shortint;const Limit:Shortint;const Decr:Shortint): Shortint;
619: Func DecLimitClamp2(var B:Smallint;const Limit: Smallint;const Decr: Smallint): Smallint;
620: Func DecLimitClamp3(var B:Word;const Limit: Word; const Decr : Word) : Word;
621: Func DecLimitClamp4( var B : Int; const Limit : Int; const Decr : Int) : Int;
622: Func DecLimitClamp5(var B:Card;const Limit:Card; const Decr:Card) Cardinal;
623: Func DecLimitClamp6(var B: Int64; const Limit:Int64; const Decr:Int64): Int64;
624: Func DecodeDateFully(DateTime: TDateTime; var Year,Month,Day,DOW:Word):Bool
625: Func DecodeSoundexInt( AValue : Int) :Str
626: Func DecodeSoundexWord( AValue : Word) :Str
627: Func DefaultAlignment : TAlignment
628: Func DefaultCaption :Str
629: Func DefaultColor : TColor
630: Func DefaultFont : TFont
631: Func DefaultImeMode : TImeMode
632: Func DefaultImeName : TImeName
633: Func DefaultReadOnly :Bool
634: Func DefaultWidth : Int
635: Func DegMinSecToFloat( const Degs, Mins, Secs : Float) : Float
636: Func DegToCycle(const Degrees : Extended) : Extended
637: Func DegToGrad(const Degrees : Extended) : Extended
638: Func DegToGrad(const Value : Extended) : Extended;
639: Func DegToGrad1(const Value : Double) : Double;
640: Func DegToGrad2(const Value : Single) : Single;
641: Func DegToRad(const Degrees : Extended) : Extended
642: Func DegToRad(const Value : Extended) : Extended;
643: Func DegToRad1(const Value : Double) : Double;
644: Func DegToRad2(const Value : Single) : Single;
645: Func DelChar(const pStr :Str; const pChar : Char) :Str
646: Func DelEnvironmentVar( const Name :Str) :Bool
647: Func Delete( const MsgNum : Int) :Bool
648: Func DeleteDirectory(const DirectoryName:str;MoveToRecycleBin:Boolean):Bool
649: Func DeleteFile(const FileName:Str):Bool)
650: Func DeleteFileEx( FileName :Str; Flags : FILEOP_FLAGS) :Bool
651: Func DelimiterPosn( const sString :Str; const sDelimiters:Str): Int;
652: Func DelimiterPosn1(const sString:str;const sDelimiters:str;out cDelimiter:char):Int;
653: Func DelSpace( const pStr :Str) :Str
654: Func DelString( const pStr, pDelStr :Str) :Str
655: Func DelTree( const Path :Str) :Bool
656: Func Depth: Int
657: Func Description :Str
658: Func DescriptionsAvailable :Bool
659: Func DescriptionToConvFamily(const ADescription:str;out AFamily:TConvFamily):Boolean
660: Func DescriptionToConvType(const ADescription:str;out AType:TConvType):Bool;
661: Func DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:str;out AType:TConvType):Bool;
662: Func DetectUTF8Encoding( const s : UTF8String) : TEncodeType
663: Func DialogsToPixelsX( const Dialogs : Word) : Word
664: Func DialogsToPixelsY( const Dialogs : Word) : Word
665: Func Digits( const X :Card) : Int
666: Func DirectoryExists( const Name :Str) :Bool
667: Func DirectoryExists( Directory :Str) :Bool
668: Func DiskFree( Drive : Byte) : Int64
669: Func DiskFree(Drive: Byte): Int64)
670: Func DiskInDrive( Drive : Char) :Bool
671: Func DiskSize( Drive : Byte) : Int64
672: Func DiskSize(Drive: Byte): Int64)
673: Func DISPATCHCOMMAND( ACOMMAND : WORD):BOOLEAN
674: Func DispatchEnabled :Bool
675: Func DispatchMask : TMask
676: Func DispatchMethodType : TMethodType
677: Func DISPATCHPOPUP( AHANDLE : HMENU) :Bool
678: Func DispatchRequest(Sender:TObject;Request:TWebRequest;Response:TWebResponse):Boolean
679: Func DisplayCase( const S :Str) :Str
680: Func DisplayRect( Code : TDisplayCode) : TRect
681: Func DisplayRect( TextOnly :Bool) : TRect
682: Func DisplayStream( Stream : TStream) :Str
683: TBufferCoord', 'record Char : Int; Line : Int; end
684: TDisplayCoord', 'record Column : Int; Row : Int; end
685: Func DisplayCoord( AColumn, ARow : Int) : TDisplayCoord
686: Func BufferCoord( AChar, ALine : Int) : TBufferCoord
687: Func DomainName( const AHost :Str) :Str
688: Func DownloadFile(SourceFile, DestFile:Str):Bool;  //fast!
689: Func DownloadFileOpen(SourceFile, DestFile:Str):Bool;  //open process
690: Func DosPathToUnixPath( const Path :Str) :Str
691: Func DottedLineTo( const Canvas : TCanvas; const X, Y : Int) :Bool;
692: Func DoubleDeclineingBalance(const Cost,Salvage:Extended;Life,Period:Int):Extended
693: Func DoubleToBcd( const AValue : Double) : TBcd;
694: Func DoubleToHex( const D : Double) :Str
```

```
695: Func DoUpdates :Bool
696: Func Dragging:Bool;
697: Func DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
698: Func DrawAnimatedRects(hwnd: HWND;idAni Int;const lprcFrom,lprcTo: TRect): BOOL
699: Func DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
700: Func DrawFrameControl(DC:HDC; const Rect:TRect; uType, uState: UINT) : BOOL
701: {Works like InputQuery but displays 2edits.If PasswordChar<>#0,second edits PasswordChar is set}
702: Func DualInputQuery(const ACapt,Prpt1,Prpt2:str;var AVal1,AVal2:str;PasswrdChar:Char=#0):Bool;
703: Func DupeString( const AText :Str; ACount : Int) :Str
704: Func Edit :Bool
705: Func EditCaption :Bool
706: Func EditText :Bool
707: Func EditFolderList( Folders : TStrings) :Bool
708: Func EditQueryParams(DataSet:TDataSet;List: TParams; AHelpContext:THelpContext): Bool
709: Func Elapsed( const Update :Bool) :Card
710: Func EnableProcessPrivilege(const Enable:Boolean; const Privilege:str):Bool
711: Func EnableThreadPrivilege(const Enable Boolean;const Privilege:str): Bool
712: Func EncodeDate( Year, Month, Day : Word) : TDateTime
713: Func EncodeDate(Year, Month, Day: Word): TDateTime;
714: Func EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
715: Func EncodeDateMonthWeek(const AYear,AMonth,AWeekOfMonth ADayOfWeek:Word): TDateTime
716: Func EncodeDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSec:Word):TDateTime
717: Func EncodeDateWeek(const AYear,AWeekOfYear:Word;const ADayOfWeek:Word):TDateTime
718: Func EncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek Word): TDateTime
719: Func EncodeString( s :Str) :Str
720: Func DecodeString( s :Str) :Str
721: Func EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
722: Func EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
723: Func EndIP :Str
724: Func EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
725: Func EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
726: Func EndOfAMonth( const AYear, AMonth : Word) : TDateTime
727: Func EndOfAWeek(const AYear,AWeekOfYear:Word; const ADayOfWeek:Word):TDateTime
728: Func EndOfAYear( const AYear : Word) : TDateTime
729: Func EndOfTheDay( const AValue : TDateTime) : TDateTime
730: Func EndOfTheMonth( const AValue : TDateTime) : TDateTime
731: Func EndOfTheWeek( const AValue : TDateTime) : TDateTime
732: Func EndOfTheYear( const AValue : TDateTime) : TDateTime
733: Func EndPeriod( const Period :Card) :Bool
734: Func EndsStr( const ASubText, AText :Str) :Bool
735: Func EndsText( const ASubText, AText :Str) :Bool
736: Func EnsureMsgIDBrackets( const AMsgID :Str) :Str
737: Func EnsureRange( const AValue, AMin, AMax : Int) : Int;
738: Func EnsureRange1( const AValue, AMin, AMax : Int64) : Int64;
739: Func EnsureRange2( const AValue, AMin, AMax : Double) : Double;
740: Func EOF:Bool
741: Func EOln:Bool
742: Func EqualRect( const R1, R2 : TRect) :Bool
743: Func EqualRect(const R1, R2: TRect):Bool)
744: Func Equals( Strings : TWideStrings) :Bool
745: Func Equals(Strings: TStrings):Bool;
746: Func EqualState(oState: TniRegularExpressionState) :Bool
747: Func ErrOutput: Text)
748: Func ExceptionParam:Str;
749: Func ExceptionPos:Card;
750: Func ExceptionProc:Card;
751: Func ExceptionToString(er: TIFException; Param:Str):Str;
752: Func ExceptionType: TIFException;
753: Func ExcludeTrailingBackslash( S :Str) :Str
754: Func ExcludeTrailingBackslash(const S:Str):Str)
755: Func ExcludeTrailingPathDelimiter( const APath :Str) :Str
756: Func ExcludeTrailingPathDelimiter( S :Str) :Str
757: Func ExcludeTrailingPathDelimiter(const S:Str):Str)
758: Func ExecConsoleApp(const AppName,Parameters:str;AppOutput:TStrings):bool;
759: Func ExecProc : Int
760: Func ExecSQL : Int
761: Func ExecSQL( ExecDirect :Bool) : Int
762: Func Execute: _Recordset;
763: Func Execute:Bool
764: Func Execute:Bool;
765: Func Execute(const SQL:str;Params:TParams;Cache:Boolean;cursor:phDBICur):Int
766: Func Execute(const SQL: WideString;Params:TParams;ResultSet: TPSResult):Int
767: Func Execute( ParentWnd : HWND) :Bool
768: Func Execute1(constCommText:WideString;const CType:TCommType;const ExecuteOpts:TExecuteOpts):_Recordset;
769: Func Execute1( const Parameters : OleVariant) : _Recordset;
770: Func Execute1( ParentWnd : HWND) :Bool;
771: Func Execute2( var RecordsAffected:Int;const Parameters:OleVariant):_Recordset;
772: Func ExecuteAction( Action : TBasicAction) :Bool
773: Func ExecuteDirect( const SQL : WideString) : Int
774: Func ExecuteFile(const FileName:str;const Params:str;const DefDir:str;ShowCmd:Int):THandle
775: Proc ExecuteThread2(afunc:TThreadFunction2;thrOK:bool);AddTypeS('TThreadFunction2','procedure
776: Func CreateThread2(ThreadFunc: TThreadFunction2) : THandle
777: Func ExeFileIsRunning(ExeFile:Str):Bool;
778: Func ExePath:Str;
779: Func ScriptPath:Str;
780: Func ScriptName:Str;
781: Func ExePathName:Str;
782: Func Exists( AItem : Pointer) :Bool
783: Func ExitWindows( ExitCode :Card) :Bool
```

```
784: Func Exit2( ExitCode :Card) :Bool
785: Func Exp(x: Extended): Extended;
786: Func ExpandEnvironmentVar( var Value :Str) :Bool
787: Func ExpandFileName( FileName :Str) :Str
788: Func ExpandFileName(const FileName:Str):Str)
789: Func ExpandUNCFileName( FileName :Str) :Str
790: Func ExpandUNCFileName(const FileName:Str):Str)
791: Func ExpJ( const X : Float) : Float;
792: Func Exsecans( X : Float) : Float
793: Func Extract( const AByteCount : Int) :Str
794: Func Extract( Item : TClass) : TClass
795: Func Extract( Item : TComponent) : TComponent
796: Func Extract( Item : TObject) : TObject
797: Func ExtractFileDir( FileName :Str) :Str
798: Func ExtractFileDir(const FileName:Str):Str)
799: Func ExtractFileDrive( FileName :Str) :Str
800: Func ExtractFileDrive(const FileName:Str):Str)
801: Func ExtractFileExt( FileName :Str) :Str
802: Func ExtractFileExt(const FileName:Str):Str)
803: Func ExtractFileExtNoDot( const FileName :Str) :Str
804: Func ExtractFileExtNoDotUpper( const FileName :Str) :Str
805: Func ExtractFileName( FileName :Str) :Str
806: Func ExtractFileName(const filename:Str):str;
807: Func ExtractFilePath( FileName :Str) :Str
808: Func ExtractFilePath(const filename:Str):str;
809: Func ExtractRelativePath( BaseName, DestName :Str) :Str
810: Func ExtractRelativePath(const BaseName:Str; const DestName:Str):Str)
811: Func ExtractShortPathName( FileName :Str) :Str
812: Func ExtractShortPathName(const FileName:Str):Str)
813: Func ExtractStrings(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strings:TStrings):Int
814: Func ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings):Int)
815: Func Fact(numb: Int): Extended;
816: Func FactInt(numb: Int): int64;
817: Func Factorial( const N : Int) : Extended
818: Func FahrenheitToCelsius( const AValue : Double) : Double
819: Func FalseBoolStrs: array of string
820: Func Fetch(var AInput:str;const ADelim:str;const ADelete:Bool;const ACaseSensitive:Bool):str
821: Func FetchCaseInsensitive(var AInput:str;const ADelim:str;const ADelete:Bool):str
822: Func Fibo(numb: Int): Extended;
823: Func FiboInt(numb: Int): Int64;
824: Func Fibonacci( const N : Int) : Int
825: Func FIELDBYNAME(const FIELDNAME :Str) : TFIELD
826: Func FIELDBYNAME(const FIELDNAME : WIDESTRING) : TFIELD
827: Func FIELDBYNAME(const NAME :Str) : TFIELD
828: Func FIELDBYNAME(const NAME :Str) : TFIELDDEF
829: Func FIELDBYNUMBER( FIELDNO : Int) : TFIELD
830: Func FileAge( FileName :Str) : Int
831: Func FileAge(const FileName:Str): Int)
832: Func FileCompareText( const A, B :Str) : Int
833: Func FileContains(const FileName:str;Text:str;CaseSensitive:Bool;ExceptionOnError:Bool):Bool
834: Func FileCreate(FileName:Str): Int;
835: //FileCreate2(FileName:str;Rights:Int):Int;
836: Func FileCreate(const FileName:Str): Int)
837: Func FileCreateTemp( var Prefix :Str) : THandle
838: Func FileDateToDateTime( FileDate : Int) : TDateTime
839: Func FileDateToDateTime(FileDate: Int): TDateTime;
840: Func FileExists( const FileName :Str) :Bool
841: Func FileExists( FileName :Str) :Bool
842: Func fileExists(const FileName:Str):Bool;
843: Func FileGetAttr( FileName :Str) : Int
844: Func FileGetAttr(const FileName:Str): Int)
845: Func FileGetDate( Handle : Int) : Int
846: Func FileGetDate(handle: Int): Int
847: Func FileGetDisplayName( const FileName :Str) :Str
848: Func FileGetSize( const FileName :Str) : Int
849: Func FileGetTempName( const Prefix :Str) :Str
850: Func FileGetTypeName( const FileName :Str) :Str
851: Func FileIsReadOnly( FileName :Str) :Bool
852: Func FileLoad( ResType:TResType;const Name:str;MaskColor:TColor):Boolean
853: Func FileOpen( FileName :Str; Mode : LongWord) : Int
854: Func FileOpen(const FileName:Str; mode:Int): Int)
855: Func FileRead(handle: Int; Buffer: PChar; count: LongWord): Int
856: Func FileSearch( Name, DirList :Str) :Str
857: Func FileSearch(const Name, dirList:Str):Str)
858: Func FileSeek( Handle : Int; Offset : Int64; Origin : Int) : Int64;
859: Func FileSeek( Handle, Offset, Origin : Int) : Int;
860: Func FileSeek(handle, offset, origin: Int): Int
861: Func FileSetAttr( FileName :Str; Attr : Int) : Int
862: Func FileSetAttr(const FileName:Str; Attr: Int): Int)
863: Func FileSetDate(FileName :Str; Age : Int) : Int;
864: Func FileSetDate(handle: Int; age: Int): Int
865: Func FileSetDate2(FileHandle : Int; Age : Int) : Int;
866: Func FileSetDateH( Handle : Int; Age : Int) : Int;
867: Func FileSetReadOnly( FileName :Str; ReadOnly :Bool) :Bool
868: Func FileSize( const FileName :Str) : int64
869: Func FileSizeByName( const AFilename :Str) : Longint
870: Func FileWrite(Handle: Int; const Buffer: pChar; Count: LongWord): Int)
871: Func FilterSpecArray : TComdlgFilterSpecArray
872: Func FIND(ACAPTION :Str) : TMENUITEM
```

```
873: Func Find(AItem : Pointer; out AData : Pointer) :Bool
874: Func FIND(const ANAME :Str) : TNAMEDITEM
875: Func Find(const DisplayName :Str) : TAggregate
876: Func Find(const Item : TBookmarkStr; var Index : Int) :Bool
877: Func FIND(const NAME :Str) : TFIELD
878: Func FIND(const NAME :Str) : TFIELDDEF
879: Func FIND(const NAME :Str) : TINDEXDEF
880: Func Find(const S : WideString; var Index : Int) :Bool
881: Func Find(S:str;var Index:Int):Boolean
882: Func FindAuthClass( AuthName :Str) : TIdAuthenticationClass
883: Func FindBand( AControl : TControl) : TCoolBand
884: Func FindBoundary( AContentType :Str) :Str
885: Func FindButton( AModalResult : TModalResult) : TTaskDialogBaseButtonItem
886: Func FindCaption(StartIndex:Int;Value:str;Partial,Inclusive,Wrap:Boolean):TListItem
887: Func FindCdLineSwitch( Switch :Str; IgnoreCase :Bool) :Bool;
888: Func FindCloseW(FindFile: Int): LongBool; stdcall;
889: Func FindCmdLineSwitch(Switch:str;Chars:TSysCharSet;IgnoreCase:Boolean):Bool;
890: Func FindCmmdLineSwitch( Switch :Str) :Bool;
891: Func FindComponent(AName:Str): TComponent;
892: Func FindComponent(vlabel:Str): TComponent;
893: Func FindComponent2(vlabel:Str): TComponent;
894: Func FindControl(Handle: HWnd): TWinControl;
895: Func FindData( StartIndex : Int; Value : Pointer; Inclusive, Wrap :Bool):TListItem
896: Func FindDatabase( const DatabaseName :Str) : TDatabase
897: Func FindDragTarget(const Pos: TPoint; AllowDisabled:Bool): TControl;
898: Func FINDFIELD( const FIELDNAME :Str) : TFIELD
899: Func FINDFIELD( const FIELDNAME : WideString) : TFIELD
900: Func FindFirst2(const Path:Str; Attr: Int; var F: TSearchRec):Int)
901: Func FindNext2(var F: TSearchRec): Int)
902: Proc FindClose2(var F: TSearchRec)
903: Func FINDFIRST :Bool
904: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
905:   sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms,sfRecent,sfSendTo,
     sfStartMenu, stStartUp, sfTemplates);
906:  FFolder: array [TJvSpecialFolder] of Int =
907:     (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
908:      CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
909:      CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
910:      CSIDL_STARTUP, CSIDL_TEMPLATES);
911: Func FindFilesDlg(StartIn:str;SpecialFolder:TJvSpecialFolder;UseFolder:Bool):Bool);
912: Func Findfirst(const filepath:Str; attr: Int): Int;
913: Func FindFirst2(const Path:Str; Attr: Int; var F: TSearchRec): Int)
914: Func FindFirstNotOf( AFind, AText :Str) : Int
915: Func FindFirstOf( AFind, AText :Str) : Int
916: Func FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
917: Func FINDINDEXFORFIELDS( const FIELDS :Str) : TINDEXDEF
918: Func FindInstanceOf( AClass : TClass; AExact :Bool; AStartAt : Int) : Int
919: Func FINDITEM( VALUE : Int; KIND : TFINDITEMKIND) : TMENUITEM
920: Func FindItemId( Id : Int) : TCollectionItem
921: Func FindKey( const KeyValues : array of const) :Bool
922: Func FINDLAST:BOOLEAN
923: Func FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
924: Func FindModuleClass( AClass : TComponentClass) : TComponent
925: Func FindModuleName( const AClass :Str) : TComponent
926: Func FINDNEXT :Bool
927: Func FindNext: Int;
928: Func FindNext2(var F: TSearchRec): Int)
929: Func FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible :Bool):TTabSheet
930: Func FindNextToSelect : TTreeNode
931: Func FINDPARAM( const VALUE :Str) : TPARAM
932: Func FindParam( const Value : WideString) : TParameter
933: Func FINDPRIOR :Bool
934: Func FindResource(ModuleHandle:HMODULE;ResourceName,ResourceType:PChar):TResourceHandle
935: Func FindSession( const SessionName :Str) : TSession
936: Func FindStringResource(Ident: Int):Str)
937: Func FindText(const SearchStr:str;StartPos,Length:Int;Options:TSearchTypes):Int
938: Func FindUnusedFileName( const FileName, FileExt, Suffix : Ansistr) : Ansistr
939: Func FindVCLWindow(const Pos: TPoint): TWinControl;
940: Func FindWindow(C1, C2: PChar): Longint;
941: Func FindInPaths(const fileName,paths:Str):Str;
942: Func Finger :Str
943: Func First: TClass
944: Func First: TComponent
945: Func First: TObject
946: Func FirstDelimiter( const delimiters :Str; const Str :Str) : Int;
947: Func FirstDelimiter1(const delimiters:WideString; const Str : WideString) : Int;
948: Func FirstInstance( const ATitle :Str) :Bool
949: Func FloatPoint( const X, Y : Float) : TFloatPoint;
950: Func FloatPoint1( const P : TPoint) : TFloatPoint;
951: Func FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) :Bool
952: Func FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
953: Func FloatRect1( const Rect : TRect) : TFloatRect;
954: Func FloatsEqual( const X, Y : Float) :Bool
955: Func FloatToBin(const D: Double):Str;      //doubletohex -> hextobin! in buffer
956: Func FloatToCurr( Value : Extended) : Currency
957: Func FloatToDateTime( Value : Extended) : TDateTime
958: Func FloatToStr( Value : Extended) :Str;
959: Func FloatToStr(e : Extended) :Str;
960: Func FloatToStrF(Value:Extended;Format:TFloatFormat;Precision,Digits:Int):str;
```

```
 961: Func FloatToStrF(Value:Extended;Format:TFloatFormat;Precision:Int;Digits:Int):Str)
 962: Func FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Int; FormatSettings :
      TFormatSettings) :Str;
 963: Func FloatToStrFS(Value:Extended;Format:TFloatFormat;Precision,
      Digits:Int;FormatSettings:TFormatSettings):str;
 964: Func FloatToText(BufferArg:PChar;const Value:Extended;ValueType:TFloatValue;Format:TFloatFormat;Precision,
      Digits: Int):Int)
 965: Func Floor( const X : Extended) : Int
 966: Func FloorInt( Value : Int; StepSize : Int) : Int
 967: Func FloorJ( const X : Extended) : Int
 968: Func Flush( const Count :Card) :Bool
 969: Func Flush(var t: Text): Int
 970: Func FmtLoadStr(Ident: Int; const Args: array of const):Str)
 971: Func FOCUSED:BOOLEAN
 972: Func ForceBackslash( const PathName :Str) :Str
 973: Func ForceDirectories( const Dir :Str) :Bool
 974: Func ForceDirectories( Dir :Str) :Bool
 975: Func ForceDirectories( Name :Str) :Bool
 976: Func ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
 977: Func ForceInRange( A, Min, Max : Int) : Int
 978: Func ForceInRangeR( const A, Min, Max : Double) : Double
 979: Func ForEach( AProc : TBucketProc; AInfo : Pointer) :Bool;
 980: Func ForEach1( AEvent : TBucketEvent) :Bool;
 981: Func ForegroundTask:Bool
 982: Func Format(const Format:Str; const Args: array of const):Str;
 983: Func FormatBcd( const Format :Str; Bcd : TBcd) :Str
 984: Func FormatBigInt(s:Str):Str;
 985: Func FormatByteSize(const bytes: int64):Str;
 986: Func FormatBuf(var Buffer:PChar;BufLen:Card;const Format:str;FmtLen:Card;const Args:array of const):Card
 987: Func FormatCount(iCount:Int;const sSingular:str;const sPlural:str):str
 988: Func FormatCurr( Format :Str; Value : Currency) :Str;
 989: Func FormatCurr(const Format:Str; Value: Currency):Str)
 990: Func FormatDateTime( Format :Str; DateTime : TDateTime) :Str;
 991: Func FormatDateTime(const fmt:Str; D: TDateTime):Str;
 992: Func FormatDateTime2(const Formatting:str;DateTime:TDateTime;const FormatSettings: TFormatSettings):str;
 993: Func FormatFloat( Format :Str; Value : Extended) :Str;
 994: Func FormatFloat(const Format:Str; Value: Extended):Str)
 995: Func FormatFloat( Format :Str; Value : Extended) :Str;
 996: Func FormatFloat2(Format:str;Value:Extended;FormatSettings:TFormatSettings):str;
 997: Func FormatF('%.2f - %.12f ',[notes[it],(ln(notes[it+1]/notes[it])*1200)/ln(2)]);  //with println - writeln
 998: Func FormatCurr( Format :Str; Value : Currency) :Str;
 999: Func FormatCurr2(Format:str;Value:Currency;FormatSettings:TFormatSettings):str;
1000: Func Format2(const Frmat:str;const Args:array of const;const FSettings:TFormatSettings):str
1001: Func FormatInt(i: Int):Str;
1002: Func FormatInt64(i: int64):Str;
1003: Func FormatMaskText( const EditMask :Str; const Value :Str) :Str
1004: Func FormatValue( AValue :Card) :Str
1005: Func FormatVersionString( const HiV, LoV : Word) :Str;
1006: Func FormatVersionString1( const Major, Minor, Build, Revision : Word) :Str;
1007: Func Frac(X: Extended): Extended);
1008: Func FreeResource( ResData : HGLOBAL) : LongBool
1009: Func FromCommon( const AValue : Double) : Double
1010: Func FromCommon(const AValue: Double): Double;
1011: Func FTPGMTDateTimeToMLS( const ATimeStamp:TDateTime; const AIncludeMSecs:Boolean):str
1012: Func FTPLocalDateTimeToMLS(const ATimeStamp:TDateTime;const AIncludeMSecs:Boolean):str
1013: Func FTPMLSToGMTDateTime( const ATimeStamp :Str) : TDateTime
1014: Func FTPMLSToLocalDateTime( const ATimeStamp :Str) : TDateTime
1015: Func FuncIn(AValue: Variant; ASet: Variant):Bool;
1016: //Func Funclist Size is: 6444 of mX3.9.8.9
1017: Func FutureValue(const Rate:Extended;NPeriods:Int;const Payment,
      PresentValue:Extended;PaymentTime:TPaymentTime):Extended
1018: Func FullTimeToStr(SUMTime: TDateTime):Str;;
1019: Func Gauss( const x, Spread : Double) : Double
1020: Func Gauss(const x,Spread: Double): Double;
1021: Func GCD(x, y : LongInt) : LongInt;
1022: Func GCDJ( X, Y :Card) :Card
1023: Func GDAL: LongWord
1024: Func GdiFlush : BOOL
1025: Func GdiSetBatchLimit( Limit : DWORD) : DWORD
1026: Func GdiGetBatchLimit : DWORD
1027: Func GenerateHeader : TIdHeaderList
1028: Func GeometricMean( const X : TDynFloatArray) : Float
1029: Func Get( AURL :Str) :Str;
1030: Func Get2( AURL :Str) :Str;
1031: Func Get8087CW : Word
1032: Func GetActiveOleObject(const ClassName:Str): IDispatch;
1033: Func GetAliasDriverName( const AliasName :Str) :Str
1034: Func GetAPMBatteryFlag : TAPMBatteryFlag
1035: Func GetAPMBatteryFullLifeTime : DWORD
1036: Func GetAPMBatteryLifePercent : Int
1037: Func GetAPMBatteryLifeTime : DWORD
1038: Func GetAPMLineStatus : TAPMLineStatus
1039: Func GetAppdataFolder :Str
1040: Func GetAppDispatcher : TComponent
1041: Func GetArrayLength: Int;
1042: Func GetASCII:Str;
1043: Func GetASCIILine:Str;
1044: Func GetAsHandle( Format : Word) : THandle
1045: Func GetAssociatedProgram(const Extension:str;var Filename,Description:str):bool;
```

```
1046: Func GetBackupFileName( const FileName :Str) :Str
1047: Func GetBaseAddress(PID:DWORD):DWORD;  //Process API
1048: Func GetBBitmap( Value : TBitmap) : TBitmap
1049: Func GetBIOSCopyright :Str
1050: Func GetBIOSDate : TDateTime
1051: Func GetBIOSExtendedInfo :Str
1052: Func GetBIOSName :Str
1053: Func getBitmap(apath:Str): TBitmap;
1054: Func GetBitmap( Index : Int; Image : TBitmap) :Bool  //object
1055: Func getBitMapObject(const bitmappath:Str): TBitmap;
1056: Func GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1057: Func GetCapsLockKeyState :Bool
1058: Func GetCaptureControl: TControl;
1059: Func GetCDAudioTrackList(var TrackList:TJclCdTrackInfoArray;Drive:Char):TJclCdTrackInfo;
1060: Func GetCDAudioTrackList1(TrackList:TStrings;IncludeTrackType:Boolean;Drive:Char):str;
1061: Func GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) :Str
1062: Func GetChangedText( const Text:str; SelStart, SelLength : Int; Key : Char):str
1063: Func GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1064: Func GetClockValue : Int64
1065: Func getCmdLine: PChar;
1066: Func getCmdShow: Int;
1067: Func GetCPUSpeed: Double;
1068: Func GetColField( DataCol : Int) : TField
1069: Func GetColorBlue( const Color : TColor) : Byte
1070: Func GetColorFlag( const Color : TColor) : Byte
1071: Func GetColorGreen( const Color : TColor) : Byte
1072: Func GetColorRed( const Color : TColor) : Byte
1073: Func GetComCtlVersion : Int
1074: Func GetComPorts: TStringlist;
1075: Func GetCommonAppdataFolder :Str
1076: Func GetCommonDesktopdirectoryFolder:Str
1077: Func GetCommonFavoritesFolder :Str
1078: Func GetCommonFilesFolder :Str
1079: Func GetCommonProgramsFolder :Str
1080: Func GetCommonStartmenuFolder :Str
1081: Func GetCommonStartupFolder :Str
1082: Func GetComponent( Owner, Parent : TComponent) : TComponent
1083: Func GetConnectionRegistryFile( DesignMode :Bool):Str
1084: Func GetCookiesFolder :Str
1085: Func GetCPUSpeed( var CpuSpeed : TFreqInfo) :Bool
1086: Func GetCurrent:TFavoriteLinkItem
1087: Func GetCurrent:TListItem
1088: Func GetCurrent:TTaskDialogBaseButtonItem
1089: Func GetCurrent:TToolButton
1090: Func GetCurrent:TTreeNode
1091: Func GetCurrent:WideString
1092: Func GetCurrentDir :Str
1093: Func GetCurrentDir:Str)
1094: Func GetCurrentFolder :Str
1095: Func GETCURRENTRECORD( BUFFER : PCHAR) :Bool
1096: Func GetCurrentProcessId : TIdPID
1097: Func GetCurrentThreadHandle : THandle
1098: Func GetCurrentThreadID: LongWord; stdcall;
1099: Func GetCustomHeader( const Name :Str) :Str
1100: Func GetDataItem( Value : Pointer) : Longint
1101: Func GetDataLinkFiles(FileNames:TWideStrings; Directory string): Int;
1102: Func GetDataLinkFiles1( FileNames : TStrings; Directory:Str):Int;
1103: Func GETDATASIZE : Int
1104: Func GetDC(hdwnd: HWND): HDC;
1105: Func GetDefaultFileExt( const MIMEType :Str) :Str
1106: Func GetDefaults :Bool
1107: Func GetDefaultSchemaName : WideString
1108: Func GetDefaultStreamLoader : IStreamLoader
1109: Func GetDesktopDirectoryFolder :Str
1110: Func GetDesktopFolder :Str
1111: Func GetDFAState( oStates : TList) : TniRegularExpressionState
1112: Func GetDirectorySize( const Path :Str) : Int64
1113: Func GetDisplayWidth : Int
1114: Func GetDLLVersion(const DLLName:str;var pdwMajor, pdwMinor:Int):Boolean
1115: Func GetDomainName :Str
1116: Func GetDriverRegistryFile( DesignMode :Bool) :Str
1117: Func GetDriveType(rootpath: pchar):Card;
1118: Func GetDriveTypeStr( const Drive : Char) :Str
1119: Func GetEnumerator : TFavoriteLinkItemsEnumerator
1120: Func GetEnumerator : TListItemsEnumerator
1121: Func GetEnumerator : TTaskDialogButtonsEnumerator
1122: Func GetEnumerator : TToolBarEnumerator
1123: Func GetEnumerator : TTreeNodesEnumerator
1124: Func GetEnumerator : TWideStringsEnumerator
1125: Func GetEnvVar( const VarName :Str) :Str
1126: Func GetEnvironmentVar( const AVariableName :Str) :Str
1127: Func GetEnvironmentVariable( const VarName :Str) :Str
1128: Func GetEnvironmentVar(const Name string; var Value:str;Expand:Boolean):Bool
1129: Func GetEnvironmentVars( const Vars : TStrings; Expand :Bool):Boolean
1130: Func getEnvironmentString:Str;
1131: Func GetExceptionHandler : TObject
1132: Func GetFavoritesFolder :Str
1133: Func GetFieldByName( const Name :Str) :Str
1134: Func GetFieldInfo(const Origin: Widestring;var FieldInfo TFieldInfo):Boolean
```

```
1135: Func GetFieldValue( ACol : Int) :Str
1136: Func GetFileAgeCoherence( const FileName :Str) :Bool
1137: Func GetFileCreation( const FileName :Str) : TFileTime
1138: Func GetFileCreationTime( const Filename :Str) : TDateTime
1139: Func GetFileInformation( const FileName :Str) : TSearchRec
1140: Func GetFileLastAccess( const FileName :Str) : TFileTime
1141: Func GetFileLastWrite( const FileName :Str) : TFileTime
1142: Func GetFileList(FileList: TStringlist; apath:Str): TStringlist;
1143: Func GetFileList1(apath:Str): TStringlist;
1144: Func GetFileMIMEType( const AFileName :Str) :Str
1145: Func GetFileSize( const FileName :Str) : Int64
1146: Func GetFileVersion( AFileName :Str) :Card
1147: Func GetFileVersion( const AFilename :Str) :Card
1148: Func GetFileSize2(Handle: Int; x: Int): Int; stdcall;
1149: Func GetFileDate(aFile:str; aWithTime:Boolean):str;
1150: Func GetFileCount(adirmask:Str): Int;  //files count in directory!
1151: Func GetFilterData( Root : PExprNode) : TExprData
1152: Func getFirstChild : LongInt
1153: Func getFirstChild : TTreeNode
1154: Func GetFirstDelimitedToken( const cDelim:char;const cStr:Str):str
1155: Func GetFirstNode : TTreeNode
1156: Func GetFontsFolder :Str
1157: Func GetFormulaValue( const Formula :Str) : Extended
1158: Func GetFreePageFileMemory : Int
1159: Func GetFreePhysicalMemory : Int
1160: Func GetFreeSystemResources( const ResourceType : TFreeSysResKind): Int;
1161: Func GetFreeSystemResources1 : TFreeSystemResources;
1162: Func GetFreeVirtualMemory : Int
1163: Func GetFromClipboard :Bool
1164: Func GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet):Str
1165: Func GetGBitmap( Value : TBitmap) : TBitmap
1166: Func GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
1167: Func GetGroupState( Level : Int) : TGroupPosInds
1168: Func GetHandle : HWND
1169: Func GETHELPCONTEXT( VALUE : Int; BYCOMMAND :Bool) : THELPCONTEXT
1170: Func GetHexArray(ahexdig: THexArray): THexArray;
1171: Func GetHighLightColor( const Color : TColor; Luminance : Int):TColor
1172: Func GetHINSTANCE: longword;
1173: Func GetHistoryFolder :Str
1174: Func GetHitTestInfoAt( X, Y : Int) : THitTests
1175: Func getHMODULE: longword;
1176: Func GetHostByName(const AComputerName:Str):str;
1177: Func GetHostName :Str
1178: Func getHostIP:Str;
1179: Func GetHotSpot : TPoint
1180: Func GetHueBitmap( Value : TBitmap) : TBitmap
1181: Func GetImageBitmap : HBITMAP
1182: Func GETIMAGELIST : TCUSTOMIMAGELIST
1183: Func GetIncome( const aNetto : Currency): Currency
1184: Func GetIncome( const aNetto : Extended): Extended
1185: Func GetIncome( const aNetto : Extended): Extended
1186: Func GetIncome(const aNetto : Extended) : Extended
1187: Func GetIncome(const aNetto: Currency): Currency
1188: Func GetIncome2( const aNetto : Currency) : Currency
1189: Func GetIncome2( const aNetto : Currency): Currency
1190: Func getIndex_Attrs( tag :Str; var idx : Int; var Attrs :Str) :Str
1191: Func GETINDEXFORFIELDS( const FIELDS :Str; CASEINSENSITIVE :Bool) : TINDEXDEF
1192: Func GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet): TIndexDef
1193: Func GetInstRes(Instance:THandle;ResType:TResType;const
      Name:str;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1194: Func
      GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1195: Func GetIntelCacheDescription( const D : Byte):Str
1196: Func GetInteractiveUserName :Str
1197: Func GetInternetCacheFolder :Str
1198: Func GetInternetFormattedFileTimeStamp( const AFilename :Str) :Str
1199: Func GetIPAddress( const HostName :Str) :Str
1200: Func GetIP( const HostName :Str) :Str
1201: Func GetIPHostByName(const AComputerName:Str):Str;
1202: Func GetIsAdmin:Bool;
1203: Func GetItem( X, Y : Int) : LongInt
1204: Func GetItemAt( X, Y : Int) : TListItem
1205: Func GetItemHeight(Font: TFont): Int;
1206: Func GetItemPath( Index : Int) :Str
1207: Func GetKeyFieldNames( List : TStrings) : Int;
1208: Func GetKeyFieldNames1( List : TWideStrings) : Int;
1209: Func GetKeyState( const VirtualKey :Card) :Bool
1210: Func GetLastChild : LongInt
1211: Func GetLastChild : TTreeNode
1212: Func GetLastDelimitedToken(const cDelim:char;const cStr:str):Str
1213: Func GetLastError: Int
1214: Func GetLAT_CONV_FACTOR: double;  //for WGS84 power(1 - 1 / 298.257223563, 2);
1215: Func GetLinesCount(sFileName :Str): Int;
1216: Func GetLoader( Ext :Str) : TBitmapLoader
1217: Func GetLoadFilter :Str
1218: Func GetLocalComputerName :Str
1219: Func GetLocaleChar( Locale, LocaleType : Int; Default : Char) : Char
1220: Func GetLocaleStr( Locale, LocaleType : Int; Default :Str) :Str
1221: Func GetLocalUserName :Str
```

```
1222: Func GetLoginUsername : WideString
1223: Func getLongDayNames:Str)
1224: Func GetLongHint(const hint:Str):Str
1225: Func getLongMonthNames:Str)
1226: Func GetMacAddresses( const Machine :Str; const Addresses : TStrings) : Int
1227: Func GetMainAppWndFromPid( PID : DWORD) : HWND
1228: Func GetMapX(C_form,apath:Str; const Data:Str):Bool; //c_form: [html/json/xml]
1229: //if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne then
1230: Proc GetGEOMap(C_form,apath:Str; const Data:Str);
1231: Func GetMapXGeoReverse(C_form:Str; const lat,long:Str):Str;
1232: //if GetMapXGeoReverse('XML',topPath,'47.0397826','7.62914761277888 then
1233: Func GetGeoCode(C_form,apath:Str; const data:Str; sfile:bool):Str;
1234: Func GetMaskBitmap : HBITMAP
1235: Func GetMaxAppAddress : Int
1236: Func GetMciErrorMessage( const MciErrNo : MCIERROR) :Str
1237: Func GetMemoryLoad : Byte
1238: Func GetMIMEDefaultFileExt( const MIMEType :Str) : TIdFileName
1239: Func GetMIMETypeFromFile( const AFile :Str) :Str
1240: Func GetMIMETypeFromFile( const AFile : TIdFileName) :Str
1241: Func GetMinAppAddress : Int
1242: Func GetModule : TComponent
1243: Func GetModuleHandle( ModuleName : PChar) : HMODULE
1244: Func GetModuleName( Module : HMODULE) :Str
1245: Func GetModulePath( const Module : HMODULE) :Str
1246: Func GetModuleFileName(Module: Int; Filename: PChar;Size: Int): Int; stdcall;
1247: Func GetMorseID(InChar : Char): Word;;
1248: Func GetMorseString2(InChar : Char):Str;;
1249: Func GetMorseLine(dots:Bool):Str;;  //whole table!  {1 or dots}
1250: Func GetMorseTable(dots:Bool):Str;;  //whole table!
1251: Func GetMorseSign(InChar : Char):Str;;
1252: Func GetCommandLine: PChar; stdcall;
1253: Func GetMonochromeBitmap( Value : TBitmap) : TBitmap
1254: Func GetMultiN(aval: Int):Str;
1255: Func GetName :Str
1256: Func GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1257: Func GetNethoodFolder :Str
1258: Func GetNext : TTreeNode
1259: Func GetNextChild( Value : LongInt) : LongInt
1260: Func GetNextChild( Value : TTreeNode) : TTreeNode
1261: Func GetNextDelimitedToken( const cDelim : char; var cStr :Str) :Str
1262: Func GetNextItem(StartItem:TListItem;Direction:TSearchDirection;States:TItemStates):TListItem
1263: Func GetNextPacket : Int
1264: Func getNextSibling : TTreeNode
1265: Func GetNextVisible : TTreeNode
1266: Func GetNode( ItemId : HTreeItem) : TTreeNode
1267: Func GetNodeAt( X, Y : Int) : TTreeNode
1268: Func GetNodeDisplayWidth( Node : TOutlineNode) : Int
1269: Func GetNumberOfProcessors: longint;
1270: Func GetNumLockKeyState :Bool
1271: Func GetObjectProperty( Instance : TPersistent; const PropName :Str) : TObject
1272: Func GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1273: Func GetOptionalParam( const ParamName :Str) : OleVariant
1274: Func GetOSName:Str;
1275: Func GetOSVersion:Str;
1276: Func GetOSNumber:Str;
1277: Func GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
1278: Func GetPackageModuleHandle( PackageName : PChar) : HMODULE
1279: Func GetPageSize:Card;
1280: Func GetParameterFileName :Str
1281: Func GetParams( var OwnerData : OleVariant) : OleVariant
1282: Func GETPARENTCOMPONENT : TCOMPONENT
1283: Func GetParentForm(control: TControl): TForm
1284: Func GETPARENTMENU : TMENU
1285: Func GetPassword:Bool
1286: Func GetPassword:Str
1287: Func GetPersonalFolder :Str
1288: Func GetPidFromProcessName( const ProcessName :Str) : DWORD
1289: Func getPI: extended;        //of const PI math
1290: Func GetPixel(dc:HDC; PixelCoords.X: TPoint, PixelCoords.Y: TPoint);
1291: Func GetPosition : TPoint
1292: Func GetPrev : TTreeNode
1293: Func GetPrevChild( Value : LongInt) : LongInt
1294: Func GetPrevChild( Value : TTreeNode) : TTreeNode
1295: Func getPrevSibling : TTreeNode
1296: Func GetPrevVisible : TTreeNode
1297: Func GetPrinthoodFolder :Str
1298: Func GetPrivilegeDisplayName( const PrivilegeName:Str):Str
1299: Func getProcessList: TStrings;
1300: Func GetProcessId : TIdPID
1301: Func GetProcessNameFromPid( PID : DWORD) :Str
1302: Func GetProcessNameFromWnd( Wnd : HWND) :Str
1303: Func GetProcessMemoryInfo(Process:THandle;ppsmemCounters:TProcessMemoryCounters;cb:DWORD):BOOL
1304: Func getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1305: Func getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1306: Func GetProgramFilesFolder :Str
1307: Func GetProgramsFolder :Str
1308: Func GetProxy :Str
1309: Func GetQuoteChar : WideString
1310: Func GetQrCode4(Width,Height:Word; Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
```

```
Func GetQrCodetoFile(Width,Height:Word;Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
Func GetRate : Double
Func getPerfTime:Str;
Func getRuntime:Str;
Func GetRBitmap( Value : TBitmap) : TBitmap
Func GetReadableName( const AName :Str) :Str
Func GetRecentDocs : TStringList
Func GetRecentFolder :Str
Func GetRecords( Count : Int; out RecsOut : Int; Options : Int) : OleVariant;
Func GetRecords1(Count:Int;out RecsOut:Int;Options:Int;const CommandText:WideString;var Params,
OwnerData:OleVariant):OleVariant;
Func GetRecordset(const CommandText: WideString;ConnectionString:WideString):_Recordset
Func GetRegisteredCompany :Str
Func GetRegisteredOwner :Str
Func GetResource(ResType:TResType;const Name:str;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor:Bool
Func GetResourceName( ObjStream : TStream; var AName :Str) :Bool
Func GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
Func GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
Func GetRValue(rgb: DWORD): Byte
Func GetGValue(rgb: DWORD): Byte
Func GetBValue(rgb: DWORD): Byte
Func GetCValue(cmyk: COLORREF): Byte
Func GetMValue(cmyk: COLORREF): Byte
Func GetYValue(cmyk: COLORREF): Byte
Func GetKValue(cmyk: COLORREF): Byte
Func CMYK( c, m, y, k : Byte) : COLORREF
Proc GetScreenShot(var ABitmap : TBitmap);
Func GetOSName:Str;
Func GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR):FARPROC
Func GetProcAddress(Module : HMODULE; Proc : PChar): Dword
Func GetSafeCallExceptionMsg :Str
Func GetSaturationBitmap( Value : TBitmap) : TBitmap
Func GetSaveFilter :Str
Func GetSaver( Ext :Str) : TBitmapLoader
Func GetScrollLockKeyState :Bool
Func GetSearchString :Str
Func GetSelections( AList : TList) : TTreeNode
Func GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:Int):Int
Func GetSendToFolder :Str
Func GetServer : IAppServer
Func GetServerList : OleVariant
Func GetShadowColor( const Color : TColor; Luminance:Int): TColor
Func GetShellProcessHandle : THandle
Func GetShellProcessName :Str
Func GetShellVersion :Card
Func getShortDayNames:Str)
Func GetShortHint(const hint:Str):Str
Func getShortMonthNames:Str)
Func GetSizeOfFile( const FileName :Str) : Int64;
Func GetSizeOfFile1( Handle : THandle) : Int64;
Func GetStdHandle(nStdHandle: Int): Int; stdcall;
Func GetStartmenuFolder :Str
Func GetStartupFolder :Str
Func GetStringProperty( Instance: TPersistent; const PropName:str):WideString
Func GetSuccessor( cChar : char): TniRegularExpressionState
Func GetSwapFileSize : Int
Func GetSwapFileUsage : Int
Func GetSystemLocale : TIdCharSet
Func GetSystemMetrics( nIndex : Int) : Int
Func GetSystemPathSH(Folder: Int): TFilename ;
Func GetTableNameFromQuery( const SQL : Widestring) : Widestring
Func GetTableNameFromSQL( const SQL : WideString) : WideString
Func GetTableNameFromSQLEx(const SQL:WideString; IdOption:IDENTIFIEROption):WideString
Func GetTasksList( const List : TStrings) :Bool
Func getTeamViewerID:Str;
Func GetTemplatesFolder :Str
Func GetText : PwideChar
Func GetText:PChar
Func GetTextBuf( Buffer : PChar; BufSize : Int) : Int
Func GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:Int):Int
Func GetTextItem( const Value :Str) : Longint
Func GETTEXTLEN:Int
Func GetThreadLocale: Longint; stdcall
Func GetCurrentThreadID: LongWord; stdcall;
Func GetTickCount :Card
Func GetTickDiff( const AOldTickCount, ANewTickCount :Card) :Card
Func GetTicketNr : longint
Func GetTime :Card
Func GetTime : TDateTime
Func GetTimeout : Int
Func GetTimeStr:Str
Func GetTimeString:Str
Func GetTimeZoneInformation(var lpTimeZoneInformation: TTimeZoneInformation):DWORD
  Ex: if GetTimeZoneInformation(tzi) >0 then writeln(itoa(tzi.bias));
Func GetTodayFiles(startdir, amask:Str): TStringlist;
Func getTokenCounts : Int
Func GetTotalPageFileMemory : Int
Func GetTotalPhysicalMemory : Int
Func GetTotalVirtualMemory : Int
```

```
1399: Func GetUniqueFileName( const APath, APrefix, AExt :Str) :Str
1400: Func GetUseNowForDate :Bool
1401: Func GetUserDomainName( const CurUser :Str) :Str
1402: Func GetUserName :Str
1403: Func GetUserName:Str;
1404: Func GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD):BOOL;
1405: Func GetUserObjectName( hUserObject : THandle) :Str
1406: Func GetValueBitmap( Value : TBitmap) : TBitmap
1407: Func GetValueMSec :Card
1408: Func GetValueStr :Str
1409: Func GetVersion: int;
1410: Func GetVersionString(FileName:Str):Str;
1411: Func getVideoDrivers:Str;
1412: Func GetVisibleNode( Index : LongInt) : TOutlineNode
1413: Func GetVolumeFileSystem( const Drive :Str) :Str
1414: Func GetVolumeName( const Drive :Str) :Str
1415: Func GetVolumeSerialNumber( const Drive :Str) :Str
1416: Func GetWebAppServices : IWebAppServices
1417: Func GetWebRequestHandler : IWebRequestHandler
1418: Func GetWindowCaption( Wnd : HWND) :Str
1419: Func GetWindowDC(hdwnd: HWND): HDC;
1420: Func GetWindowIcon( Wnd : HWND; LargeIcon :Bool) : HICON
1421: Func GetWindowRect(hwnd: HWND; arect: TRect):Bool
1422: Func GetWindowsComputerID :Str
1423: Func GetWindowsDirectory(lpBuffer: PChar; uSize: longword):longword;
1424: Func GetWindowsFolder :Str
1425: Func GetWindowsServicePackVersion : Int
1426: Func GetWindowsServicePackVersionString :Str
1427: Func GetWindowsSystemFolder :Str
1428: Func GetWindowsTempFolder :Str
1429: Func GetWindowsUserID :Str
1430: Func GetWindowsVersion : TWindowsVersion
1431: Func GetWindowsVersionString :Str
1432: Func GmtOffsetStrToDateTime( S :Str) : TDateTime
1433: Func GMTToLocalDateTime( S :Str) : TDateTime
1434: Func GotoKey :Bool
1435: Func GradToCycle( const Grads : Extended) : Extended
1436: Func GradToDeg( const Grads : Extended) : Extended
1437: Func GradToDeg( const Value : Extended) : Extended;
1438: Func GradToDeg1( const Value : Double) : Double;
1439: Func GradToDeg2( const Value : Single) : Single;
1440: Func GradToRad( const Grads : Extended) : Extended
1441: Func GradToRad( const Value : Extended) : Extended;
1442: Func GradToRad1( const Value : Double) : Double;
1443: Func GradToRad2( const Value : Single) : Single;
1444: Func Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1445: Func GreenComponent( const Color32 : TColor32) : Int
1446: Func GUIDToString(const GUID: TGUID):Str)
1447: Func HandleAllocated :Bool
1448: Func HandleAllocated:Bool;
1449: Func HandleRequest :Bool
1450: Func HandleRequest( Request : TWebRequest; Response : TWebResponse:Boolean
1451: Func HarmonicMean( const X : TDynFloatArray) : Float
1452: Func HasAsParent( Value : TTreeNode) :Bool
1453: Func HASCHILDDEFS :Bool
1454: Func HasCurValues :Bool
1455: Func HasExtendCharacter( const s : UTF8String) :Bool
1456: Func HasFormat( Format : Word) :Bool
1457: Func HashValue( AStream : TStream) : T5x4LongWordRecord;
1458: Func HashValue(AStream : TStream) : T4x4LongWordRecord
1459: Func HashValue(AStream: TStream): LongWord
1460: Func HashValue(AStream: TStream): Word
1461: Func HashValue1(AStream TStream; const ABeginPos,AEndPos:Int64):T5x4LongWordRecord;
1462: Func HashValue1(AStream : TStream): T4x4LongWordRecord
1463: Func HashValue128(const ASrc:Str): T4x4LongWordRecord;
1464: Func HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1465: Func HashValue16( const ASrc :Str) : Word;
1466: Func HashValue16stream( AStream : TStream) : Word;
1467: Func HashValue32( const ASrc :Str) : LongWord;
1468: Func HashValue32Stream( AStream : TStream) : LongWord;
1469: Func HasMergeConflicts :Bool
1470: Func hasMoreTokens :Bool
1471: Func HASPARENT :Bool
1472: Func HasParent:Bool
1473: Func HasTransaction( Transaction : TDBXTransaction) :Bool
1474: Func HasUTF8BOM( S : TStream) :Bool;
1475: Func HasUTF8BOM1( S : Ansistr) :Bool;
1476: Func Haversine( X : Float) : Float
1477: Func Head( s :Str; const subs :Str; var tail :Str):str
1478: Func HELPCOMMAND(COMMAND:Int;DATA:LONGINT):BOOLEAN
1479: Func HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1480: Func HELPJUMP(JUMPID:str):BOOLEAN
1481: Func HeronianMean( const a, b : Float) : Float
1482: Func HexStrToStr(Value:Str):Str;
1483: Func HexToBin(Text,Buffer:PChar; BufSize:Int):Int;
1484: Func HexToBin2(HexNum:Str):Str;
1485: Func HexToDouble( const Hex :Str) : Double
1486: Func HexToInt(hexnum:Str): LongInt;
1487: Func HexToStr(Value:Str):Str;
```

```
1488: Func HexifyBlock( var Buffer, BufferSize : Int) :Str
1489: Func Hi(vdat: word): byte;
1490: Func HiByte(W: Word): Byte)
1491: Func High: Int64;
1492: Func HighlightCell(DataCol,DataRow:Int;const Value:str;AState:TGridDrawState):Bool
1493: Func HINSTANCE: longword;
1494: Func HiWord(l: DWORD): Word)
1495: Func HMODULE: longword;
1496: Func HourOf( const AValue : TDateTime) : Word
1497: Func HourOfTheDay( const AValue : TDateTime) : Word
1498: Func HourOfTheMonth( const AValue : TDateTime) : Word
1499: Func HourOfTheWeek( const AValue : TDateTime) : Word
1500: Func HourOfTheYear( const AValue : TDateTime) : Word
1501: Func HoursBetween( const ANow, AThen : TDateTime) : Int64
1502: Func HourSpan( const ANow, AThen : TDateTime) : Double
1503: Func HSLToRGB1( const H, S, L : Single) : TColor32;
1504: Func HTMLDecode( const AStr :Str) :Str
1505: Func HTMLEncode( const AStr :Str) :Str
1506: Func HTMLEscape( const Str :Str) :Str
1507: Func HtmlTable(DataSet:TDataSet;DataSetHandler:TDSTableProducer;MaxRows:Int):str
1508: Func HTTPDecode( const AStr :Str) :Str
1509: Func HTTPEncode( const AStr :Str) :Str
1510: Func Hypot( const X, Y : Extended) : Extended
1511: Func IBMax( n1, n2 : Int) : Int
1512: Func IBMin( n1, n2 : Int) : Int
1513: Func IBRandomString( iLength : Int) :Str
1514: Func IBRandomInt( iLow, iHigh : Int) : Int
1515: Func IBStripString( st :Str; CharsToStrip :Str) :Str
1516: Func IBFormatIdentifier( Dialect : Int; Value :Str) :Str
1517: Func IBFormatIdentifierValue( Dialect : Int; Value :Str) :Str
1518: Func IBExtractIdentifier( Dialect : Int; Value :Str) :Str
1519: Func IBQuoteIdentifier( Dialect : Int; Value :Str) :Str
1520: Func IBAddIBParamSQLForDetail(Params:TParams;SQL:str;Native:Boolean;Dialect:Int):str
1521: Proc IBDecomposeDatabaseName(DatabaseName:Str;var ServerName,Protocol,DatabasePath:str)
1522: Func RandomString( iLength : Int) :Str;
1523: Func RandomInt( iLow, iHigh : Int) : Int;
1524: Func StripString( st :Str; CharsToStrip :Str) :Str;
1525: Func Strip(const SubString:Str; MainString:Str):Str;
1526: Func StripTags(const S:Str):Str;   //<''> of HTML
1527: Func SizeToString(size : Int64; const unitStr :Str) :Str;
1528: Func NumbertoString(No: Word):Str;
1529: Func FormatIdentifier( Dialect : Int; Value :Str) :Str;
1530: Func FormatIdentifierValue( Dialect : Int; Value :Str) :Str;
1531: Func ExtractIdentifier( Dialect : Int; Value :Str) :Str;
1532: Func QuoteIdentifier( Dialect : Int; Value :Str) :Str;
1533: Func AddIBParamSQLForDetail(Params:TParams;SQL:str;Native:Boolean;Dialect:Int):str;
1534: Proc DecomposeDatabaseName(DatabaseName:str;var ServerName,Protocol,DatabasePath:str;
1535: Func NextSQLToken(var p: PChar; var Token:Str; CurSection: TSQLToken): TSQLToken;
1536: Func IconToBitmap( Ico : HICON) : TBitmap
1537: Func IconToBitmap2( Ico : HICON; Size : Int; TransparentColor : TColor) : TBitmap
1538: Func IconToBitmap3( Ico : HICON; Size : Int; TransparentColor : TColor) : TBitmap
1539: Func IdentToCharset(const Ident:Str; var Charset: Longint):Bool)
1540: Func IdentToColor(const Ident:Str; var Color: Longint):Bool)
1541: Func IdentToCursor(const Ident:Str; var cursor: Longint):Bool;
1542: Func IdGetDefaultCharSet : TIdCharSet
1543: Func IDispatchInvoke(Self:IDispatch;ProperSet:Bool;const Name:Str;Par:array of variant):variant
1544: Func IdPorts2 : TStringList
1545: Func IdToMib( const Id :Str) :Str
1546: Func IdSHA1Hash(apath:Str):Str;
1547: Func IdHashSHA1(apath:Str):Str;
1548: Func IfStr(const bCondition:boolean;const sTrue:str;const sFalse:str):Str
1549: Func IfThen( AValue :Bool; const ATrue :Str; AFalse :Str) :Str;
1550: Func IfThenInt( AValue :Bool; const ATrue : Int; AFalse : Int): Int;;
1551: Func IfThenDouble( AValue :Bool; const ATrue : double; AFalse : double): double;;
1552: Func IfThenBool( AValue :Bool; const ATrue :Bool; AFalse :Bool):Bool;;
1553: Func iif1( ATest :Bool; const ATrue : Int; const AFalse : Int) : Int;
1554: Func iif2( ATest :Bool; const ATrue :Str; const AFalse :Str) :Str;
1555: Func iif3( ATest:Boolean; const ATrue:Bool;const AFalse:Bool):Bool;
1556: Func ImportTest(S1:Str;s2:longint; s3:Byte; s4:word; var s5:str):Str;
1557: Func IncDay( const AValue : TDateTime; const ANumberOfDays : Int) : TDateTime
1558: Func IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1559: Func IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1560: Func IncLimit1(var B:Shortint;const Limit:Shortint;const Incr:Shortint):Shortint;
1561: Func IncLimit2(var B:Smallint;const Limit:Smallint;const Incr:Smallint):Smallint;
1562: Func IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1563: Func IncLimit4( var B : Int; const Limit : Int; const Incr : Int) : Int;
1564: Func IncLimit5(var B:Card;const Limit:Card;const Incr:Card):Card;
1565: Func IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1566: Func IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1567: Func IncLimitClamp1(var B: Shortint;const Limit:Shortint;const Incr: Shortint): Shortint;
1568: Func IncLimitClamp2(var B:Smallint;const Limit:Smallint;const Incr: Smallint): Smallint;
1569: Func IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1570: Func IncLimitClamp4( var B : Int; const Limit : Int; const Incr : Int) : Int;
1571: Func IncLimitClamp5(var B:Card;const Limit:Card; const Incr:Card):Card;
1572: Func IncLimitClamp6(var B: Int64;const Limit:Int64; const Incr:Int64):Int64;
1573: Func IncludeTrailingBackslash( S :Str) :Str
1574: Func IncludeTrailingBackslash(const S:Str):Str)
1575: Func IncludeTrailingPathDelimiter( const APath :Str) :Str
1576: Func IncludeTrailingPathDelimiter( S :Str) :Str
```

```
1577: Func IncludeTrailingPathDelimiter(const S:Str):Str)
1578: Func IncludeTrailingSlash( const APath :Str) :Str
1579: Func IncMilliSecond(const AValue:TDateTime;const ANumberOfMilliSeconds:Int64):TDateTime
1580: Func IncMinute(const AValue:TDateTime;const ANumberOfMinutes:Int64):TDateTime
1581: Func IncMonth( DateTime : TDateTime; NumberOfMonths : Int) : TDateTime
1582: Func IncMonth(const DateTime: TDateTime; NumberOfMonths: Int): TDateTime)
1583: Func IncSecond(const AValue:TDateTime;const ANumberOfSeconds: nt64) : TDateTime
1584: Func IncWeek(const AValue : TDateTime; const ANumberOfWeeks : Int) : TDateTime
1585: Func IncYear(const AValue : TDateTime; const ANumberOfYears : Int) : TDateTime
1586: Func IndexOf(AClass : TClass) : Int
1587: Func IndexOf(AComponent : TComponent) : Int
1588: Func IndexOf(AObject : TObject) : Int
1589: Func INDEXOF(const ANAME :Str) : Int
1590: Func IndexOf(const DisplayName :Str) : Int
1591: Func IndexOf(const Item : TBookmarkStr) : Int
1592: Func IndexOf(const S : WideString) : Int
1593: Func IndexOf(const View : TJclFileMappingView) : Int
1594: Func INDEXOF(FIELD : TFIELD) : Int
1595: Func IndexOf(ID : LCID) : Int
1596: Func INDEXOF(ITEM : TMENUITEM) : Int
1597: Func IndexOf(Value : TListItem) : Int
1598: Func IndexOf(Value : TTreeNode) : Int
1599: Func IndexOf(const S:Str): Int;
1600: Func IndexOfName( const Name : WideString) : Int
1601: Func IndexOfName(Name:Str): Int;
1602: Func IndexOfObject( AObject : TObject) : Int
1603: Func IndexofObject(AObject:tObject):Int
1604: Func IndexOfTabAt( X, Y : Int) : Int
1605: Func IndexStr( const AText :Str; const AValues : array of string) : Int
1606: Func IndexText( const AText :Str; const AValues : array of string): Int
1607: Func IndexOfInt( AList : TStringList; Value : Variant) : Int
1608: Func IndexOfFloat( AList : TStringList; Value : Variant) : Int
1609: Func IndexOfDate( AList : TStringList; Value : Variant) : Int
1610: Func IndexOfString( AList : TStringList; Value : Variant) : Int
1611: Func IndyCompareStr( const A1 :Str; const A2 :Str) : Int
1612: Func IndyGetHostName :Str
1613: Func IndyInterlockedDecrement( var I : Int) : Int
1614: Func IndyInterlockedExchange( var A : Int; B : Int) : Int
1615: Func IndyInterlockedExchangeAdd( var A : Int; B : Int) : Int
1616: Func IndyInterlockedIncrement( var I : Int) : Int
1617: Func IndyLowerCase( const A1 :Str) :Str
1618: Func IndyStrToBool( const AString :Str) :Bool
1619: Func IndyUpperCase( const A1 :Str) :Str
1620: Func InitCommonControl( CC : Int) :Bool
1621: Func InitTempPath :Str
1622: Func InMainThread :Bool
1623: Func inOpArray(W : WideChar; sets:array of WideChar):Bool
1624: Func Input: Text)
1625: Func InputBox( const ACaption, APrompt, ADefault :Str) :Str
1626: Func InputBox(const ACaption:str;const APrompt:str;const ADefault:str):str)
1627: Func InputLn(const AMask:str;AEcho:Boolean;ATabWidth:Int;AMaxLineLength:Int):str
1628: Func InputQuery( const ACaption, APrompt :Str; var Value :Str) :Bool
1629: Func InputQuery(const ACaption:str; const APrompt:str; var Value:Str):Bool)
1630: Func InquireSignal( RtlSigNum : Int) : TSignalState
1631: Func InRangeR( const A, Min, Max : Double) :Bool
1632: Func Insert(Index:Int):TCollectionItem
1633: Func Insert(Index:Int):TComboExItem
1634: Func Insert(Index:Int):THeaderSection
1635: Func Insert(Index:Int):TListItem
1636: Func Insert(Index:Int):TStatusPanel
1637: Func Insert(Index:Int):TWorkArea
1638: Func Insert(Index:LongInt; const Text :Str) : LongInt
1639: Func Insert(Sibling : TTreeNode; const S :Str) : TTreeNode
1640: Func INSERTNEWLINEAFTER( AITEM : TMENUITEM) : Int
1641: Func INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : Int
1642: Func InsertNode( Node, Sibling : TTreeNode; const S :Str; Ptr : Pointer) : TTreeNode
1643: Func InsertObject( Index : LongInt; const Text :Str; const Data : Pointer) : LongInt
1644: Func InsertObject( Sibling : TTreeNode; const S :Str; Ptr : Pointer) : TTreeNode
1645: Func Instance : Longint
1646: Func InstanceSize: Longint
1647: Func Int(e : Extended) : Extended;
1648: Func Int64ToStr(i: Int64):Str;
1649: Func IntToBcd( const AValue : Int) : TBcd
1650: Func Intensity( const Color32 : TColor32) : Int;
1651: Func Intensity( const R, G, B : Single) : Single;
1652: Func InterestPayment(const Rate:Ext;Period,NPeriods:Int;const PresentValue,FutureValue:Ext;PaymentTime
      :TPaymentTime):Ext
1653: Func InterestRate(NPeriods:Int;const Payment,PresVal,FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1654: Func InternalDecodeDate(DateTime:TDateTime;var Year,Month,Day,DOW:Word):Boolean
1655: Func InternalRateOfReturn(const Guess:Extended;const CashFlows:array of Double):Extended
1656: Func InternalUpdateRecord( Tree : TUpdateTree) :Bool
1657: Func IntersectRect( out Rect : TRect; const R1, R2 : TRect) :Bool
1658: Func IntersectRect(out Rect: TRect; const R1, R2: TRect):Bool)
1659: Func IntMibToStr( const Value :Str) :Str
1660: Func IntPower( const Base : Extended; const Exponent : Int) : Extended
1661: Func IntToBin( Value :Card) :Str
1662: Func IntToHex( Value : Int; Digits : Int) :Str;
1663: Func IntToHex(a: Int; b: Int):Str;
1664: Func IntToHex64( Value : Int64; Digits : Int) :Str;
```

```
1665: Func IntToHex64(Value: Int64; Digits: Int):Str)
1666: Func IntTo3Str( Value : Longint; separator:Str) :Str
1667: Func inttobool( aInt : LongInt) :Bool
1668: Func IntToStr(i: Int64):Str;
1669: Func IntToStr64(Value: Int64):Str)
1670: Func IOResult: Int
1671: Func IPv6AddressToStr(const AValue: TIdIPv6Address):Str
1672: Func IPAddrToHostName(const IP:Str):Str;
1673: Func IsAccel(VK: Word; const Str:Str):Bool
1674: Func IsAddressInNetwork( Address :Str) :Bool
1675: Func IsAdministrator :Bool
1676: Func IsAlias( const Name :Str) :Bool
1677: Func IsApplicationRunning( const AClassName, ApplName:str):Bool
1678: Func IsASCII( const AByte : Byte) :Bool;
1679: Func IsASCIILDH( const AByte : Byte) :Bool;
1680: Func IsAssembly(const FileName:Str):Bool;
1681: Func IsBcdNegative( const Bcd : TBcd) :Bool
1682: Func IsBinary(const AChar : Char) :Bool
1683: Func IsConsole:Bool)
1684: Func IsDelimiter( Delimiters, S :Str; Index : Int) :Bool
1685: Func IsDelimiter(const Delimiters:Str;const S:Str;Index: Int):Boolean)
1686: Func IsDelphiDesignMode :Bool
1687: Func IsDelphiRunning :Bool
1688: Func IsDFAState :Bool
1689: Func IsDirectory( const FileName :Str) :Bool
1690: Func IsDomain( const S :Str) :Bool
1691: Func IsDragObject(Sender: TObject):Bool;
1692: Func IsEditing :Bool
1693: Func ISEMPTY :Bool
1694: Func IsEqual( Value : TParameters) :Bool
1695: Func ISEQUAL( VALUE : TPARAMS) :Bool
1696: Func IsEqualGUID(const guid1, guid2: TGUID):Bool)
1697: Func IsFirstNode :Bool
1698: Func IsFloatZero( const X : Float) :Bool
1699: Func IsFormatRegistered( Extension, AppID :Str) :Bool
1700: Func IsFormOpen(const FormName:Str):Bool;
1701: Func IsFQDN( const S :Str) :Bool
1702: Func IsGrayScale :Bool
1703: Func IsHex( AChar : Char) :Bool;
1704: Func IsHexString(const AString:Str):Bool;
1705: Func IsHostname( const S :Str) :Bool
1706: Func IsInfinite( const AValue : Double) :Bool
1707: Func IsInLeapYear( const AValue : TDateTime) :Bool
1708: Func IsInternet:Bool;
1709: Func IsLeadChar( ACh : Char) :Bool
1710: Func IsLeapYear( Year : Word) :Bool
1711: Func IsLeapYear(Year: Word):Bool)
1712: Func IsLibrary:Bool)
1713: Func ISLINE :Bool
1714: Func IsLinkedTo( DataSet : TDataSet) :Bool
1715: Func ISLINKEDTO( DATASOURCE : TDATASOURCE) :Bool
1716: Func IsLiteralChar( const EditMask :Str; Offset : Int) :Bool
1717: Func IsMatch( const Pattern, Text :Str) :Bool  //Grep like RegEx
1718: Func IsMainAppWindow( Wnd : HWND) :Bool
1719: Func IsMediaPresentInDrive( Drive : Char) :Bool
1720: Func IsMemoryManagerSet:Bool)
1721: Func IsMultiTableQuery( const SQL : WideString) :Bool
1722: Func IsMultiThread:Bool)
1723: Func IsNumeric( AChar : Char) :Bool;
1724: Func IsNumeric2( const AString :Str) :Bool;
1725: Func IsNTFS:Bool;
1726: Func IsOctal( AChar : Char) :Bool;
1727: Func IsOctalString(const AString:Str) :Bool;
1728: Func IsPathDelimiter( S :Str; Index : Int) :Bool
1729: Func IsPathDelimiter(const S:Str; Index: Int):Bool)
1730: Func IsPM( const AValue : TDateTime) :Bool
1731: Func IsPositiveFloatArray( const X : TDynFloatArray) :Bool
1732: Func IsPortAvailable( ComNum :Card) :Bool;
1733: Func IsCOMPortReal( ComNum :Card) :Bool;
1734: Func IsCOM( ComNum :Card) :Bool;
1735: Func IsCOMPort:Bool;
1736: Func IsPrimeFactor( const F, N :Card) :Bool
1737: Func IsPrimeRM( N :Card) :Bool  //rabin miller
1738: Func IsPrimeTD( N :Card) :Bool  //trial division
1739: Func IsPrivilegeEnabled( const Privilege :Str) :Bool
1740: Func ISqrt( const I : Smallint) : Smallint
1741: Func IsReadOnly(const Filename:Str):Bool;
1742: Func IsRectEmpty( const Rect : TRect) :Bool
1743: Func IsRectEmpty(const Rect: TRect):Bool)
1744: Func IsRelativePrime( const X, Y :Card) :Bool
1745: Func ISRIGHTTOLEFT :Bool
1746: Func IsRightToLeft:Bool
1747: Func IsSameDay( const AValue, ABasis : TDateTime) :Bool
1748: Func ISSEQUENCED :Bool
1749: Func IsSystemModule( const Module : HMODULE) :Bool
1750: Func IsSystemResourcesMeterPresent :Bool
1751: Func IsTCPPortOpen(dwPort : Word; ipAddressStr:Str):Bool;
1752: Func IsToday( const AValue : TDateTime) :Bool
1753: Func IsToday(const AValue: TDateTime):Bool;
```

```
1754: Func IsTopDomain( const AStr :Str) :Bool
1755: Func IsUTF8LeadByte( Lead : Char) :Bool
1756: Func IsUTF8String( const s : UTF8String) :Bool
1757: Func IsUTF8TrailByte( Lead : Char) :Bool
1758: Func ISVALIDCHAR( INPUTCHAR : CHAR) :Bool
1759: Func IsValidDate( const AYear, AMonth, ADay : Word) :Bool
1760: Func IsValidDateDay( const AYear, ADayOfYear : Word) :Bool
1761: Func IsValidDateMonthWeek(const AYear,AMonth,AWeekOfMonth,ADayOfWeek:Word): Bool
1762: Func IsValidDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond:Word): Bool
1763: Func IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) :Bool
1764: Func IsValidIdent( Ident :Str) :Bool
1765: Func IsValidIdent1(const Ident:Str; AllowDots:Bool):Bool)
1766: Func IsValidIP( const S :Str) :Bool
1767: Func IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) :Bool
1768: Func IsValidPNG(stream: TStream):Bool;
1769: Func IsValidJPEG(stream: TStream):Bool;
1770: Func IsValidISBN( const ISBN : Ansistr) :Bool
1771: Func IsVariantManagerSet:Bool; //deprecated;
1772: Func IsVirtualPcGuest :Bool;
1773: Func IsVmWareGuest :Bool;
1774: Func IsVCLControl(Handle: HWnd):Bool;
1775: Func IsWhiteString( const AStr :Str) :Bool
1776: Func IsWindowResponding( Wnd : HWND; Timeout : Int) :Bool
1777: Func IsWoW64:Bool;
1778: Func IsWin64:Bool;
1779: Func IsWow64String(var s:Str):Bool;
1780: Func IsWin64String(var s:Str):Bool;
1781: Func IsWindowsVista:Bool;
1782: Func isPowerof2(num: int64):Bool;
1783: Func powerOf2(exponent: Int): int64;
1784: Func IsZero(const A: Extended; Epsilon: Extended):Bool //overload;
1785: Func IsZero1(const A: Double; Epsilon: Double):Bool //overload;
1786: Func IsZero2(const A: Single; Epsilon: Single):Bool //overload;
1787: Func ItemAtPos( Pos : TPoint; IgnoreTabHeight :Bool) : Int
1788: Func ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):Int
1789: Func ItemRect( Index : Int) : TRect
1790: Func ITEMRECT(INDEX:Int):TRECT
1791: Func ItemWidth( Index : Int) : Int
1792: Func JavahashCode(val:Str): Int;
1793: Func JosephusG(n,k: Int; var graphout:Str): Int;
1794: Func JulianDateToDateTime( const AValue : Double) : TDateTime
1795: Func JustName(PathName :Str) :Str;  //in path and ext
1796: Func JvMessageBox( const Text, Caption :Str; Flags : DWORD) : Int;
1797: Func JvMessageBox1( const Text :Str; Flags : DWORD) : Int;
1798: Func KeepAlive :Bool
1799: Func KeysToShiftState(Keys: Word): TShiftState;
1800: Func KeyDataToShiftState(KeyData: Longint): TShiftState;
1801: Func KeyboardStateToShiftState2(const KeyboardState: TKeyboardState):TShiftState;
1802: Func KeyboardStateToShiftState: TShiftState; overload;
1803: Func Languages : TLanguages
1804: Func Last: TClass
1805: Func Last: TComponent
1806: Func Last: TObject
1807: Func LastDelimiter( Delimiters, S :Str) : Int
1808: Func LastDelimiter(const Delimiters:Str; const S:Str): Int)
1809: Func LastPos( const ASubstr :Str; const AStr :Str) : Int
1810: Func Latitude2WGS84(lat: double): double;
1811: Func LCM(m,n:longint):longint;
1812: Func LCMJ( const X, Y :Card) :Card
1813: Func Ldexp( const X : Extended; const P : Int) : Extended
1814: Func LeftStr( const AText : Ansistr; const ACount : Int) : Ansistr;
1815: Func LeftStr( const AText : WideString; const ACount : Int) : WideString;
1816: Func Length: Int;
1817: Proc LetFileList(FileList: TStringlist; apath:Str);
1818: Func lengthmp3(mp3path:Str):Int;
1819: Func LineInRect( const P1, P2 : TPoint; const Rect : TRect) :Bool;
1820: Func LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) :Bool;
1821: Func LinesCount(sfilename:str):extended;
1822: Func TextFileLineCount(const FileName:Str): Int;
1823: Func LineSegmentIntersection(const L1P1:TFloatPoint;L1P2:TFloatPoint;const L2P1:TFloatPoint;
      L2P2:TFloatPoint;var P:TFloatPoint):Boolean
1824: Func LineStart(Buffer, BufPos: PChar): PChar
1825: Func LineStart(Buffer, BufPos: PChar): PChar)
1826: Func ListSeparator: char;
1827: Func Ln(x: Extended): Extended;
1828: Func LnXP1( const X : Extended) : Extended
1829: Func Lo(vdat: word): byte;
1830: Func LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1831: Func LoadedModulesList(const List:TStrings;ProcessID:DWORD;HandlesOnly:Boolean):Bool
1832: Func LoadFileAsString( const FileName :Str) :Str
1833: Func LoadFromFile( const FileName :Str) : TBitmapLoader
1834: Func LoadFile(const FileName: TFileName):Str;
1835: Func LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1836: Func LoadPackage(const Name:Str): HMODULE
1837: Func LoadResource( ModuleHandle : HMODULE; ResHandle: TResourceHandle):HGLOBAL
1838: Func LoadStr( Ident : Int) :Str
1839: Func LoadString(Instance: Longint; IDent: Int; Buffer: PChar; Size: Int):Int;stdcall;
1840: Func LoadWideStr( Ident : Int) : WideString
1841: Func LOCATE(const KEYFIELDS:str;const KEYVALUES:VARIANT;OPTIONS:TLOCATEOPTIONS):BOOLEAN
```

```
1842: Func LockRegion(libOffset:Longint;cb:Largeint;dwLockType:Longint): HResult
1843: Func LockServer( fLock : LongBool) : HResult
1844: Func LockVolume( const Volume :Str; var Handle : THandle) :Bool
1845: Func Log( const X : Extended) : Extended
1846: Func Log10( const X : Extended) : Extended
1847: Func Log2( const X : Extended) : Extended
1848: Func LogBase10(X: Float): Float;
1849: Func LogBase2(X: Float): Float;
1850: Func LogBaseN(Base, X: Float): Float;
1851: Func LogN( const Base, X : Extended) : Extended
1852: Func LogOffOS :Bool
1853: Func LoginDialog( const ADatabaseName :Str; var AUserName, APassword :Str):Bool
1854: Func LoginDialogEx(const ADbName:str;var AUserName,APassword:str;NameReadOnly:Bool):Bool;
1855: Func LongDateFormat:Str;
1856: Func LongTimeFormat:Str;
1857: Func LongWordToFourChar( ACardinal : LongWord) :Str
1858: Func LOOKUP(const KEYFIELDS:Str;const KEYVALUES:VARIANT;const RESULTFIELDS:str):VARIANT
1859: Func LookupName( const name :Str) : TInAddr
1860: Func LookupService( const service :Str) : Int
1861: Func Low: Int64;
1862: Func LowerCase( S :Str) :Str
1863: Func Lowercase(s : AnyString) : AnyString;
1864: Func LRot( const Value : Byte; const Count : TBitRange) : Byte;
1865: Func LRot1( const Value : Word; const Count : TBitRange) : Word;
1866: Func LRot2( const Value : Int; const Count : TBitRange) : Int;
1867: Func MainInstance: longword
1868: Func MainThreadID: longword
1869: Func Map(x, in_min, in_max, out_min, out_max: Int): Int;  //arduino
1870: Func mapMax(ax, in_min, in_max, out_min, out_max: Int): Int;
1871: Func MakeCanonicalIPv4Address( const AAddr :Str) :Str
1872: Func MakeCanonicalIPv6Address( const AAddr :Str) :Str
1873: Func MakeDIB( out Bitmap : PBitmapInfo) : Int
1874: Func MakeDWordIntoIPv4Address( const ADWord :Card) :Str
1875: Func MakeFile(const FileName:Str): Int);
1876: Func MakeLong(A, B: Word): Longint)
1877: Func MakeTempFilename( const APath :Str) :Str
1878: Func MakeValidFileName( const Str :Str) :Str
1879: Func MakeValueMap( Enumeration :Str; ToCds :Bool) :Str
1880: Func MakeWord(A, B: Byte): Word)
1881: Func MakeYear4Digit( Year, Pivot : Int) : Int
1882: Func MapDateTime(const DateFormatType:str;DateFormat:str;Value:str;ToCds:Bool):str
1883: Func MapValues( Mapping :Str; Value :Str) :Str
1884: Func MaskDoFormatText(const EditMask:str;const Value:str;Blank:Char):str
1885: Func MaskGetCharType( const EditMask :Str; MaskOffset : Int) : TMaskCharType
1886: Func MaskGetCurrentDirectives( const EditMask :Str; MaskOffset:Int):TMaskDirectives
1887: Func MaskGetFldSeparator( const EditMask :Str) : Int
1888: Func MaskGetMaskBlank( const EditMask :Str) : Char
1889: Func MaskGetMaskSave( const EditMask :Str) :Bool
1890: Func MaskIntlLiteralToChar( IChar : Char) : Char
1891: Func MaskOffsetToOffset( const EditMask :Str; MaskOffset : Int) : Int
1892: Func MaskOffsetToWideOffset( const EditMask :Str; MaskOffset : Int) : Int
1893: Func MaskString( Mask, Value :Str) :Str
1894: Func Match( const sString :Str) : TniRegularExpressionMatchResul
1895: Func Match1( const sString:Str;iStart:Int):TniRegularExpressionMatchResult
1896: Func Matches( const Filename :Str) :Bool
1897: Func MatchesMask( const Filename, Mask :Str) :Bool
1898: Func MatchStr( const AText :Str; const AValues : array of string):Bool
1899: Func MatchText( const AText :Str; const AValues : array of string):Bool
1900: Func Max( AValueOne, AValueTwo : Int) : Int
1901: Func Max(const x,y: Int): Int;
1902: Func Max1(const B1, B2 : Shortint) : Shortint;
1903: Func Max2(const B1, B2 : Smallint) : Smallint;
1904: Func Max3(const B1, B2 : Word) : Word;
1905: Func Max3(const x,y,z: Int): Int;
1906: Func Max4(const B1, B2 : Int) : Int;
1907: Func Max5(const B1, B2 :Card) :Card;
1908: Func Max6(const B1, B2 : Int64) : Int64;
1909: Func Max64(const AValueOne, AValueTwo : Int64) : Int64
1910: Func MaxFloat(const X, Y : Float) : Float
1911: Func MaxFloatArray(const B : TDynFloatArray) : Float
1912: Func MaxFloatArrayIndex( const B : TDynFloatArray) : Int
1913: Func MaxIntValue(const Data: array of Int):Int)
1914: Func MaxJ( const B1, B2 : Byte) : Byte;
1915: Func MaxPath:Str;
1916: Func MaxValue(const Data: array of Double): Double)
1917: Func MaxCalc( const Formula :Str) : Extended  //math expression parser
1918: Proc MaxCalcF( const Formula :Str);   //out to console memo2
1919: Func MaxCalcS( const Formula :Str):Str;
1920: Func MD5(const fileName:Str):Str;
1921: Func Mean( const Data : array of Double) : Extended
1922: Func Median( const X : TDynFloatArray) : Float
1923: Func Memory : Pointer
1924: Func MemoryPos( const ASubStr :Str; MemBuff : PChar; MemorySize: Int): Int
1925: Func MessageBox(hndl:Card; text, caption:Str; utype:Card): Int;
1926: Func MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):Int
1927: Func MessageDlg(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint):Int;
1928: Func MessageDlg1(const
      Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn):Int;
1929: Func MessageDlgPos(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y:Int):Int;
```

```
1930: Func MessageDlgPos1(const Msg:str; DlgType:TMsgDlgType; Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y:
      Int;DefaultButton:TMsgDlgBtn):Int;
1931: Func MessageDlgPosHelp(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y:
      Int;const HelpFileName:str):Int;
1932: Func MessageDlgPosHelp1(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y :
      Int;const HelpFileName:str; DefaultButton:TMsgDlgBtn):Int;
1933: Func MibToId( Mib :Str) :Str
1934: Func MidStr( const AText : Ansistr; const AStart, ACount : Int) : Ansistr;
1935: Func MidStr( const AText : WideString; const AStart, ACount : Int) : WideString;
1936: Func microsecondsToCentimeters(mseconds:longint):longint;//340m/s speed of sound
1937: Func Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1938: Func MIDIOut( DeviceID :Card) : IJclMIDIOut
1939: Proc GetMidiOutputs( const List : TStrings)
1940: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne
1941: Proc GetGEOMap(C_form,apath:str; const Data:Str);//c_form:[html/json/xml]
1942: Func MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1943: Func MIDINoteToStr( Note : TMIDINote) :Str
1944: Func WinMidiOut( DeviceID :Card) : IJclWinMidiOut
1945: Proc GetMidiOutputs( const List : TStrings)
1946: Proc MidiOutCheck( Code : MMResult)
1947: Proc MidiInCheck( Code : MMResult)
1948: Func MilliSecondOf( const AValue : TDateTime) : Word
1949: Func MilliSecondOfTheDay( const AValue : TDateTime): LongWord
1950: Func MilliSecondOfTheHour( const AValue : TDateTime): LongWord
1951: Func MilliSecondOfTheMinute( const AValue : TDateTime): LongWord
1952: Func MilliSecondOfTheMonth( const AValue : TDateTime): LongWord
1953: Func MilliSecondOfTheSecond( const AValue : TDateTime): Word
1954: Func MilliSecondOfTheWeek( const AValue : TDateTime): LongWord
1955: Func MilliSecondOfTheYear( const AValue : TDateTime): Int64
1956: Func MilliSecondsBetween( const ANow, AThen : TDateTime): Int64
1957: Func MilliSecondSpan( const ANow, AThen : TDateTime): Double
1958: Func milliToDateTime( MilliSecond : LongInt) : TDateTime;
1959: Func Micros( const Timer : THPTimer; const TimerRunning:Boolean):Int64
1960: Func millis: int64;
1961: Func Min( AValueOne, AValueTwo : Int) : Int
1962: Func Min1(const B1,B2: Shortint) : Shortint;
1963: Func Min2(const B1,B2: Smallint) : Smallint;
1964: Func Min3(const B1,B2: Word) : Word;
1965: Func Min4(const B1,B2: Int) : Int;
1966: Func Min5(const B1,B2:Card) :Card;
1967: Func Min6(const B1,B2: Int64) : Int64;
1968: Func Min64( const AValueOne, AValueTwo : Int64) : Int64
1969: Func MinClientRect : TRect;
1970: Func MinClientRect1( IncludeScroller :Bool) : TRect;
1971: Func MinClientRect2( TabCount : Int; IncludeScroller :Bool) : TRect;
1972: Func MinFloat( const X, Y : Float) : Float
1973: Func MinFloatArray( const B : TDynFloatArray) : Float
1974: Func MinFloatArrayIndex( const B : TDynFloatArray) : Int
1975: Func MinimizeName(const Filename:str;Canvas : TCanvas; MaxLen:Int):Str
1976: Func MinimizeName(const Filename:TFileName;Canvas:TCanvas; MaxLen:Int):FileName
1977: Func MinimizeName(const Filename:Str; Canvas: TCanvas;MaxLen: Int):TFileName
1978: Func MinIntValue( const Data : array of Int) : Int
1979: Func MinIntValue(const Data: array of Int):Int)
1980: Func MinJ( const B1, B2 : Byte) : Byte;
1981: Func MinuteOf( const AValue : TDateTime) : Word
1982: Func MinuteOfTheDay( const AValue : TDateTime) : Word
1983: Func MinuteOfTheHour( const AValue : TDateTime) : Word
1984: Func MinuteOfTheMonth( const AValue : TDateTime) : Word
1985: Func MinuteOfTheWeek( const AValue : TDateTime) : Word
1986: Func MinuteOfTheYear( const AValue : TDateTime) : LongWord
1987: Func MinutesBetween( const ANow, AThen : TDateTime) : Int64
1988: Func MinuteSpan( const ANow, AThen : TDateTime) : Double
1989: Func MinValue( const Data : array of Double) : Double
1990: Func MinValue(const Data: array of Double): Double)
1991: Func MixerLeftRightToArray( Left, Right :Card) : TDynCardinalArray
1992: Func MMCheck( const MciError : MCIERROR; const Msg :Str) : MCIERROR
1993: Func ModFloat( const X, Y : Float) : Float
1994: Func ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1995: Func Modify( const Key :Str; Value : Int) :Bool
1996: Func ModuleCacheID :Card
1997: Func ModuleFromAddr( const Addr : Pointer) : HMODULE
1998: Func MonitorFromPoint(const Point:TPoint;MonitorDefault:TMonitorDefaultTo): TMonitor
1999: Func MonitorFromRect(const Rect:TRect;MonitorDefault:TMonitorDefaultTo):TMonitor
2000: Func MonitorFromWindow(const Handle:THandle; MonitorDefault:TMonitorDefaultTo): TMonitor
2001: Func MonthOf( const AValue : TDateTime) : Word
2002: Func MonthOfTheYear( const AValue : TDateTime) : Word
2003: Func MonthsBetween( const ANow, AThen : TDateTime) : Int
2004: Func MonthSpan( const ANow, AThen : TDateTime) : Double
2005: Func MonthStr( DateTime : TDateTime) :Str
2006: Func MouseCoord( X, Y : Int) : TGridCoord
2007: Func MOVEBY( DISTANCE : Int) : Int
2008: Func MoveFile( Source, Dest :Str; Flags : FILEOP_FLAGS) :Bool
2009: Func MoveNext :Bool
2010: Func MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
2011: Func MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
2012: Func Name :Str
2013: Func NetPresentValue(const Rate:Extended;const CashFlows:array of Double;PaymentTime:TPaymentTime):Extended
2014: Func NetworkVolume(DriveChar: Char):Str
2015: Func NEWBOTTOMLINE : Int
```

```
2016: Func NewCompareNode(Field: TField;Operator:TCANOperator; const Value:Variant) : PExprNode
2017: Func NEWITEM(const ACAPTION:str;ASHORTCUT:TSHORTCUT;ACHECKED,AENABLED:BOOL;AONCLICK:
      TNOTIFYEVENT;HCTX:WORD;const ANAME:str):TMENUITEM
2018: Func NEWLINE : TMENUITEM
2019: Func NEWMENU( OWNER : TCOMPONENT;const ANAME:str;ITEMS: array of TMenuItem): TMAINMENU
2020: Func NewNode(Kind: TExprNodeKind;Operator:TCANOperator;const Data:Variant;Left,Right:PExprNode):PExprNode
2021: Func NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:str; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN; const
      ITEMS : array of TCMENUITEM) : TPOPUPMENU
2022: Func NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
2023: Func NEWSUBMENU(const ACAPT:str;HCTX:WORD;const ANAME:str;ITEMS:array of TMenuItem;AENABLED:BOOL):
      TMENUITEM
2024: Func NEWTOPLINE : Int
2025: Func Next : TIdAuthWhatsNext
2026: Func NextCharIndex( S :Str; Index : Int) : Int
2027: Func NextRecordSet : TCustomSQLDataSet
2028: Func NextRecordset( var RecordsAffected : Int) : _Recordset
2029: Func NextSQLToken1(var p: WideChar;out Token:WideString;CurSection:TSQLToken):TSQLToken;
2030: Func NextToken : Char
2031: Func nextToken : WideString
2032: Func NextToken:Char
2033: Func Norm( const Data : array of Double) : Extended
2034: Func NormalizeAngle( const Angle : Extended) : Extended
2035: Func NormalizeBcd(const InBcd TBcd;var OutBcd:TBcd;const Prec,Scale Word):Bool
2036: Func NormalizeRect( const Rect : TRect) : TRect
2037: Func NormalizeRect(const Rect: TRect): TRect;
2038: Func Now :TDateTime
2039: Func Now2:tDateTime
2040: Func NumProcessThreads:Int
2041: Func NumThreadCount : Int
2042: Func NthDayOfWeek( const AValue : TDateTime) : Word
2043: Func NtProductType : TNtProductType
2044: Func NtProductTypeString :Str
2045: Func Null: Variant;
2046: Func NullPoint : TPoint
2047: Func NullRect : TRect
2048: Func Null2Blank(aString:str):str;
2049: Func NumberOfPeriods(const Rate:Extended;Payment:Extended;const PresentValue,
      FutureValue:Extended;PaymentTime:TPaymentTime):Extended
2050: Func NumIP:Int
2051: Func Odd(x: Longint):Bool;
2052: Func OffsetFromUTC : TDateTime
2053: Func OffsetPoint( const P, Offset : TPoint) : TPoint
2054: Func OffsetRect( var Rect : TRect; DX : Int; DY : Int) :Bool
2055: Func OffsetRect(var Rect: TRect; DX:Int; DY:Int):Bool)
2056: Func OffsetToMaskOffset( const EditMask :Str; Offset : Int) : Int
2057: Func OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment):Bool
2058: Func OldBCDToCurr( const BCD : TBcd; var Curr : Currency) :Bool
2059: Func OldCurrToBCD(const Curr:Currency;var BCD:TBcd;Precision:Int;Decimals:Int):Bool
2060: Func OpenBit:Int
2061: Func OpenDatabase : TDatabase
2062: Func OpenDatabase( const DatabaseName:str):TDatabase
2063: Proc OpenDir(adir:Str);
2064: Func OpenGLColorToWinColor( const Red, Green, Blue:Float):TColor
2065: Func OpenMap(const Data:Str):Bool;
2066: Func OpenMapX(const Data:Str):Bool;
2067: Func OpenObject( Value : PChar) :Bool;
2068: Func OpenObject1( Value :Str) :Bool;
2069: Proc Openweb('http://snippets.delphidabbler.com/#')
2070: Proc OpenURL( const AText : TKkString);
2071: Proc OpenWeb( const AText : TKkString);
2072: Proc OpenBrowser( const AText : TKkString);
2073: Func OpenSession( const SessionName :Str) : TSession
2074: Func OpenVolume( const Drive : Char) : THandle
2075: Func OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte):Card
2076: Func OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte): LongWord
2077: Func OrdToBinary( const Value: Byte) :Str;
2078: Func OrdToBinary1(const Value: Shortint) :Str;
2079: Func OrdToBinary2(const Value: Smallint) :Str;
2080: Func OrdToBinary3(const Value: Word) :Str;
2081: Func OrdToBinary4(const Value: Int) :Str;
2082: Func OrdToBinary5(const Value:Card) :Str;
2083: Func OrdToBinary6(const Value: Int64) :Str;
2084: Func OSCheck( RetVal :Bool) :Bool
2085: Func OSFileTypeToString( const OSFileType:DWORD;const OSFileSubType:DWORD):str
2086: Func OSIdentToString( const OSIdent : DWORD) :Str
2087: Func Output: Text)
2088: Func Overlay( ImageIndex : Int; Overlay : TOverlay):Boolean
2089: Func Owner : TCustomListView
2090: Func Owner : TPersistent
2091: Func PadInputLiterals(const EditMask:str;const Value:str;Blank:Char):str
2092: Func PadL( pStr :Str; pLth : Int) :Str
2093: Func Padl(s : AnyString;I : longInt) : AnyString;
2094: Func PadLCh( pStr :Str; pLth : Int; pChr : char) :Str
2095: Func PadR( pStr :Str; pLth : Int) :Str
2096: Func Padr(s : AnyString;I : longInt) : AnyString;
2097: Func PadRCh( pStr :Str; pLth : Int; pChr : char) :Str
2098: Func PadString(const AString:Str;const ALen:Int; const AChar:Char):str
2099: Func Padz(s : AnyString;I : longInt) : AnyString;
2100: Func PaethPredictor( a, b, c : Byte) : Byte
```

```
2101: Func PARAMBYNAME( const VALUE :Str) : TPARAM
2102: Func ParamByName( const Value : WideString): TParameter
2103: Func ParamCount: Int
2104: Func ParamsEncode( const ASrc :Str):Str
2105: Func ParamStr(Index: Int):Str)
2106: Func ParseDate( const DateStr :Str): TDateTime
2107: Func PARSESQL( SQL :Str; DOCREATE :Bool) :Str
2108: Func ParseSQL( SQL : WideString; DoCreate :Bool): WideString
2109: Func PathAddExtension( const Path, Extension :Str):str
2110: Func PathAddSeparator( const Path :Str) :Str
2111: Func PathAppend( const Path, Append :Str) :Str
2112: Func PathBuildRoot( const Drive : Byte) :Str
2113: Func PathCanonicalize( const Path :Str) :Str
2114: Func PathCommonPrefix( const Path1, Path2 :Str) : Int
2115: Func PathCompactPath(const DC:HDC;const Path:str;const Width:Int;CmpFmt:TCompactPath):str;
2116: Func PathCompactPath1(const Canv:TCanvas;const Path:str;const Width:Int;CmpFmt:TCompactPath):str;
2117: Func PathEncode( const ASrc :Str) :Str
2118: Func PathExtractFileDirFixed( const S : Ansistr): Ansistr
2119: Func PathExtractFileNameNoExt( const Path :Str):Str
2120: Func PathExtractPathDepth( const Path :Str; Depth: Int):str
2121: Func PathGetDepth( const Path :Str): Int
2122: Func PathGetLongName( const Path :Str):Str
2123: Func PathGetLongName2( Path :Str):Str
2124: Func PathGetShortName( const Path :Str):Str
2125: Func PathIsAbsolute( const Path :Str): Bool
2126: Func PathIsChild( const Path, Base : Ansistr): Bool
2127: Func PathIsDiskDevice( const Path :Str): Bool
2128: Func PathIsUNC( const Path :Str) : Bool
2129: Func PathRemoveExtension( const Path:Str):Str
2130: Func PathRemoveSeparator( const Path:Str):Str
2131: Func Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2132: Func Peek: Pointer
2133: Func Peek: TObject
2134: Func PERFORM(MSG:Card;WPARAM,LPARAM:LONGINT):LONGINT
2135: Func PeriodPayment(const Rate:Extended;Period,NPeriods:Int; const PresentValue,FutureValue:
      Extended;PaymentTime:TPaymentTime):Extended
2136: Func Permutation(npr, k: Int): extended;
2137: Func PermutationInt(npr, k: Int): Int64;
2138: Func PermutationJ( N, R :Card) : Float
2139: Func Pi : Extended;
2140: Func PiE : Extended;
2141: Func PixelsToDialogsX( const Pixels : Word) : Word
2142: Func PixelsToDialogsY( const Pixels : Word) : Word
2143: Func PlaySound(s: pchar; flag,syncflag: Int):Bool;
2144: Func Point( X, Y : Int) : TPoint
2145: Func Point(X, Y: Int): TPoint)
2146: Func PointAssign( const X, Y : Int) : TPoint
2147: Func PointDist( const P1, P2 : TPoint) : Double;
2148: Func PointDist(const P1,P2: TFloatPoint): Double;
2149: Func PointDist1( const P1, P2 : TFloatPoint) : Double;
2150: Func PointDist2(const P1,P2: TPoint): Double;
2151: Func PointEqual( const P1, P2 : TPoint) :Bool
2152: Func PointIsNull( const P : TPoint) :Bool
2153: Func PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint): Double
2154: Func Poly( const X : Extended; const Coefficients : array of Double): Extended
2155: Func PortTCPIsOpen(dwPort : Word; ipAddressStr:Str):bool;
2156: Func IsTCPPortOpen(dwPort : Word; ipAddressStr:Str):bool;
2157: Func Pop : Pointer
2158: Func Pop : TObject
2159: Func PopnStdDev( const Data : array of Double) : Extended
2160: Func PopnVariance( const Data : array of Double) : Extended
2161: Func PopulationVariance( const X : TDynFloatArray) : Float
2162: Func Pos(SubStr, S: AnyString): Longint;
2163: Func PosEqual( const Rect : TRect) :Bool
2164: Func PosEx( const SubStr, S :Str; Offset : Int) : Int
2165: Func PosInSmallIntArray(const ASearchInt:SmallInt;AArray:array of SmallInt):Int
2166: Func PosInStrArray(const SearchStr:str;Contents:array of str;const CaseSensitive:Bool):Int
2167: Func Post1( AURL:str; const ASource : TStrings) :Str;
2168: Func Post2( AURL:str; const ASource : TStream) :Str;
2169: Func Post3( AURL:str; const ASource : TIdMultiPartFormDataStream) :Str;
2170: Func PostData( const UserData : WideString; const CheckSum : DWORD) :Bool
2171: Func PostData( const UserData : WideString; const CheckSum : Int):Bool
2172: Func PostMessage(hWnd:HWND;Msg:longword;wParam: ongint;lParam:longint):Bool;
2173: Func Power( const Base, Exponent : Extended) : Extended
2174: Func PowerBig(aval, n:Int):Str;
2175: Func PowerIntJ( const X : Float; N : Int) : Float;
2176: Func PowerJ( const Base, Exponent : Float) : Float;
2177: Func PowerOffOS :Bool
2178: Func PreformatDateString( Ps :Str) :Str
2179: Func PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2180: Func PrimeFactors( N :Card) : TDynCardinalArray
2181: Func Printer : TPrinter
2182: Func ProcessPath2(const ABasePath:str;const APath:str;const APathDelim:str):str
2183: Func ProcessResponse : TIdHTTPWhatsNext
2184: Func ProduceContent :Str
2185: Func ProduceContentFromStream( Stream : TStream) :Str
2186: Func ProduceContentFromString( const S :Str) :Str
```

```
2187: Func ProgIDToClassID(const ProgID:Str): TGUID;
2188: Func PromptDataLinkFile(ParentHandle:THandle;InitialFile:WideString):WideString
2189: Func PromptDataSource(ParentHandle:THandle;InitialString:WideString):WideString
2190: Func PromptForFileName(var AFileName:str;const AFilter:str;const ADefaultExt:str;const ATitle:str;const
      AInitialDir:str;SaveDialog:Bool):Bool
2191: Func PromptForFileName(var AFileName:str;const AFilter:str;const ADefaultExt:str;const ATitle:str;const
      AInitialDir:Str;SaveDialog:Bool):Bool)
2192: Func PSScriptNeedFile(Sender:TObject;const OrginFileName:Str;var FileName,Output:Str):Bool
2193: Func PtInRect( const Rect : TRect; const P : TPoint) :Bool
2194: Func PtInRect(const Rect: TRect; const P: TPoint):Bool)
2195: Func Push( AItem : Pointer) : Pointer
2196: Func Push( AObject : TObject) : TObject
2197: Func Put1( AURL :Str; const ASource : TStream) :Str;
2198: Func Pythagoras( const X, Y : Extended) : Extended
2199: Func queryDLLInterface( var queryList : TStringList) : TStringList
2200: Func queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2201: Func QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2202: Func queryPerformanceCounter2(mse: int64): int64;
2203: //Func QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2204: //Func QueryPerformanceFrequency(mse: int64):Bool;
2205: Func QueryPerformanceCounter(var lcount: Int64):Bool; stdcall;
2206: Func QueryPerformanceFrequency(var lfreq: int64):Bool; stdcall;
2207: Proc QueryPerformanceCounter1(var aC: Int64);
2208: Func QueryPerformanceFrequency1(var freq: int64):Bool;
2209: Func Quote( const ACommand :Str) : SmallInt
2210: Func QuotedStr( S :Str) :Str
2211: Func RadToCycle( const Radians : Extended) : Extended
2212: Func RadToDeg( const Radians : Extended) : Extended
2213: Func RadToDeg( const Value : Extended) : Extended;
2214: Func RadToDeg1(const Value : Double) : Double;
2215: Func RadToDeg2(const Value : Single) : Single;
2216: Func RadToGrad(const Radians : Extended) : Extended
2217: Func RadToGrad(const Value : Extended) : Extended;
2218: Func RadToGrad1( const Value : Double) : Double;
2219: Func RadToGrad2( const Value : Single) : Single;
2220: Func RandG( Mean, StdDev : Extended) : Extended
2221: Func Random(const ARange: Int): Int;
2222: Func random2(a: Int): double
2223: Func RandomE: Extended;
2224: Func RandomF: Extended;
2225: Func RandomFrom( const AValues : array of string):Str;
2226: Func RandomRange( const AFrom, ATo : Int) : Int
2227: Func randSeed: longint
2228: Proc setRandseed; //no params!
2229: Func RawToDataColumn( ACol : Int) : Int
2230: Func Read : Char
2231: Func Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2232: Func Read(Buffer:str;Count:LongInt):LongInt
2233: Func ReadBinaryStream( const Section, Name :Str; Value : TStream) : Int
2234: Func ReadBool( const Section, Ident :Str; Default :Bool) :Bool
2235: Func ReadCardinal( const AConvert :Bool) :Card
2236: Func ReadChar : Char
2237: Func ReadClient( var Buffer, Count : Int) : Int
2238: Func ReadDate( const Section, Name :Str; Default : TDateTime) : TDateTime
2239: Func ReadDateTime( const Section, Name :Str; Default : TDateTime) : TDateTime
2240: Func ReadFloat( const Section, Name :Str; Default : Double) : Double
2241: Func ReadFromStack(const ARaiseExceptfDiscon:Bool;ATimeout:Int;const ARaiseExceptTimeout:Bool):Int
2242: Func ReadInt( const AConvert :Bool) : Int
2243: Func ReadInt( const Section, Ident :Str; Default : Longint) : Longint
2244: Func ReadLn :Str
2245: Func ReadLn(ATerminator:Str;const ATimeout:Int;AMaxLineLength:Int):str
2246: Func Readln(question:Str):Str;
2247: Func readm:Str;  //read last line in memo2 - console!
2248: Func ReadLnWait( AFailCount : Int) :Str
2249: Func ReadReg(Base: HKEY; KeyName, ValueName:Str):Str;
2250: Func ReadRegistry(Base: HKEY; KeyName, ValueName:Str):Str;
2251: Func ReadSmallInt( const AConvert :Bool) : SmallInt
2252: Func ReadString( const ABytes : Int) :Str
2253: Func ReadString( const Section, Ident, Default :Str) :Str
2254: Func ReadString( Count : Int) :Str
2255: Func ReadTime( const Section, Name :Str; Default : TDateTime) : TDateTime
2256: Func ReadTimeStampCounter : Int64
2257: Func RebootOS :Bool
2258: Func Receive( ATimeOut : Int) : TReplyStatus
2259: Func ReceiveBuf( var Buf, Count : Int) : Int
2260: Func ReceiveLength : Int
2261: Func ReceiveText :Str
2262: Func ReceiveSerialData(var Data: TByteArray; DataSize:Card):Card
2263: Func ReceiveSerialText:Str
2264: Func RecodeDate(const AValue:TDateTime;const AYear,AMonth,ADay:Word):TDateTime
2265: Func RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,AMilliSec:Word):TDateTime
2266: Func RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2267: Func RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2268: Func RecodeMilliSecond(const AValue:TDateTime;const AMilliSecond:Word):TDateTime
2269: Func RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2270: Func RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2271: Func RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2272: Func RecodeTime(const AValue:TDateTime;const AHour,AMinute,ASecd,AMilliSec:Word):TDateTime
2273: Func RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
```

```
2274: Func Reconcile( const Results : OleVariant) :Bool
2275: Func Rect( Left, Top, Right, Bottom : Int) : TRect
2276: Func Rect(ALeft: Int; ATop: Int; ARight: Int; ABottom: Int): TRect)
2277: Func Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2278: Func RectAssign( const Left, Top, Right, Bottom : Int) : TRect
2279: Func RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2280: Func RectBounds( const Left, Top, Width, Height : Int) : TRect
2281: Func RectCenter( const R : TRect) : TPoint
2282: Func RectEqual( const R1, R2 : TRect) :Bool
2283: Func RectHeight( const R : TRect) : Int
2284: Func RectIncludesPoint( const R : TRect; const Pt : TPoint) :Bool
2285: Func RectIncludesRect( const R1, R2 : TRect) :Bool
2286: Func RectIntersection( const R1, R2 : TRect) : TRect
2287: Func RectIntersectRect( const R1, R2 : TRect) :Bool
2288: Func RectIsEmpty( const R : TRect) :Bool
2289: Func RectIsNull( const R : TRect) :Bool
2290: Func RectIsSquare( const R : TRect) :Bool
2291: Func RectIsValid( const R : TRect) :Bool
2292: Func RectsAreValid( R : array of TRect) :Bool
2293: Func RectUnion( const R1, R2 : TRect) : TRect
2294: Func RectWidth( const R : TRect) : Int
2295: Func RedComponent( const Color32 : TColor32) : Int
2296: Func Refresh :Bool
2297: Func RefStringListCopy(aRefArray:TStringlist):TStringList;
2298: Func RegisterConversionFamily( const ADescription :Str) : TConvFamily
2299: Func RegisterConversionType(AConvTypeInfo: TConvTypeInfo; out AType : TConvType) :Bool;
2300: Func RegisterConversionType(const AFam:TConvFamil;const ADescr:str;const AFact:Double):TConvType
2301: Func RegistryRead(keyHandle: Longint; keyPath, myField:Str):Str;
2302: Func ReleaseDC(hdwnd: HWND; hdc: HDC): Int;
2303: Func ReleaseHandle : HBITMAP
2304: Func ReleaseHandle : HENHMETAFILE
2305: Func ReleaseHandle : HICON
2306: Func ReleasePalette : HPALETTE
2307: Func RemainderFloat( const X, Y : Float) : Float
2308: Func Remove(AClass : TClass) : Int
2309: Func Remove(AComponent : TComponent) : Int
2310: Func Remove(AItem : Int) : Int
2311: Func Remove(AItem : Pointer) : Pointer
2312: Func Remove(AItem : TObject) : TObject
2313: Func Remove(AObject : TObject) : Int
2314: Func RemoveBackslash( const PathName :Str) :Str
2315: Func RemoveDF( aString :Str) :Str  //removes thousand separator
2316: Func RemoveDir( Dir :Str) :Bool
2317: Func RemoveDir(const Dir:Str):Bool)
2318: Func RemoveDirectory(PathName: PChar): WordBool; stdcall;
2319: Func RemoveFileExt( const FileName :Str) :Str
2320: Func RemoveHeaderEntry( AHeader, AEntry :Str) :Str
2321: Func RenameFile( OldName, NewName :Str) :Bool
2322: Func RenameFile(const OldName:Str; const NewName:Str):Bool)
2323: Func ReplaceStr( const AText, AFromText, AToText :Str) :Str
2324: Func ReplaceText( const AText, AFromText, AToText :Str):Str
2325: Func Replicate(c : char;I : longInt) :Str;
2326: Func Request : TWebRequest
2327: Func ResemblesText( const AText, AOther :Str):Bool
2328: Func Reset :Bool
2329: Func Reset2(mypath:Str): TStringlist //string;
2330: Func ResInstLoad(Instance:THandle;ResType:TResType;const Name:str;MaskColor:TColor):Bool
2331: Func ResourceLoad(ResType:TResType;const Name string;MaskColor:TColor):oolean
2332: Func Response : TWebResponse
2333: Func ResumeSupported :Bool
2334: Func RETHINKHOTKEYS :Bool
2335: Func RETHINKLINES :Bool
2336: Func Retrieve( const MsgNum : Int; AMsg : TIdMessage) :Bool
2337: Func RetrieveCurrentDir :Str
2338: Func RetrieveDeltas( const cdsArray : array of TClientDataset): Variant
2339: Func RetrieveHeader( const MsgNum : Int; AMsg : TIdMessage):Bool
2340: Func RetrieveMailBoxSize : Int
2341: Func RetrieveMsgSize( const MsgNum : Int) : Int
2342: Func RetrieveProviders( const cdsArray : array of TClientDataset): Variant
2343: Func RetrieveRaw( const MsgNum : Int; const Dest: TStrings):bool
2344: Func ReturnMIMEType( var MediaType, EncType :Str):Bool
2345: Func ReverseBits( Value:Byte) : Byte;
2346: Func ReverseBits1(Value:Shortint) : Shortint;
2347: Func ReverseBits2(Value:Smallint) : Smallint;
2348: Func ReverseBits3(Value:Word) : Word;
2349: Func ReverseBits4(Value:Card) :Card;
2350: Func ReverseBits4(Value:Int) : Int;
2351: Func ReverseBits5(Value:Int64) : Int64;
2352: Func ReverseBytes(Value:Word) : Word;
2353: Func ReverseBytes1(Value:Smallint):Smallint;
2354: Func ReverseBytes2(Value:Int): Int;
2355: Func ReverseBytes3(Value:Card):Card;
2356: Func ReverseBytes4(Value:Int64): Int64;
2357: Func ReverseString(const AText :Str) :Str
2358: Func ReverseDNSLookup(const IPAddrs:Str;DNSServer:Str;Timeout,Retries:Int;var HName:Str):Bool;
2359: Func Revert : HResult
2360: Func RGB(R,G,B: Byte): TColor;
2361: Func RGB2BGR( const Color : TColor) : TColor
2362: Func RGB2TColor( R, G, B : Byte) : TColor
```

```
2363: Func RGBToWebColorName( RGB : Int) :Str
2364: Func RGBToWebColorStr( RGB : Int) :Str
2365: Func RgbToHtml( Value : TColor) :Str
2366: Func HtmlToRgb(const Value:Str): TColor;
2367: Func RightStr(const AStr :Str; Len : Int) :Str
2368: Func RightStr(const AText : Ansistr; const ACount : Int): Ansistr;
2369: Func RightStr(const AText : WideString; const ACount : Int): WideString;
2370: Func ROL( AVal: LongWord; AShift : Byte): LongWord
2371: Func ROR( AVal: LongWord; AShift : Byte): LongWord
2372: Func RotatePoint(Point:TFloatPoint;const Center:TFloatPoint;const Angle:Float):TFloatPoint
2373: Func RotatePoint(Point:TFloatPoint;const Center:TFloatPoint;const Angle:Double):TFloatPoint;
2374: Func Round(e : Extended) : Longint;
2375: Func Round64(e: extended): Int64;
2376: Func RoundAt( const Value :Str; Position : SmallInt) :Str
2377: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2378: Func RoundTo(const AValue:Extende;const ADigit:TRoundToEXRangeExtended):Extended;;
2379: Func SimpleRoundTo(const AValue:Extended;const ADigit: TRoundToRange): Extended;;
2380: Func RoundFrequency( const Frequency : Int) : Int
2381: Func RoundInt( Value : Int; StepSize : Int) : Int
2382: Func RoundPoint( const X, Y : Double) : TPoint
2383: Func RoundRect( const ALeft, ATop, ARight,ABottom :Double):TRect
2384: Func RowCount : Int
2385: Func RowRequest(const Row:OleVariant;RequestType:Int;var OwnerData:OleVariant):OleVariant
2386: Func RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2387: Func RPos( const ASub, AIn :Str; AStart : Int) : Int
2388: Func RRot( const Value : Byte; const Count : TBitRange) : Byte;
2389: Func RRot1( const Value : Word; const Count : TBitRange) : Word;
2390: Func RRot2( const Value : Int; const Count : TBitRange) : Int;
2391: Func RunDLL32(const ModuleNa,FuncName,CmdLine:str;WaitForCompletion:Bool;CmdShow:Int):Bool
2392: Func RunningProcessesList( const List : TStrings; FullPath :Bool) :Bool
2393: Func RunByteCode(Bytecode: Ansistr; out RuntimeErrors: Ansistr):Bool;;
2394: Func RunCompiledScript2(Bytecode: Ansistr;out RuntimeErrors:Ansistr): Bool;
2395: Func S_AddBackSlash( const ADirName :Str) :Str
2396: Func S_AllTrim( const cStr :Str) :Str
2397: Func S_AtRepl( const cAT, cStr, cRepl :Str) :Str
2398: Func S_Cut( const cStr :Str; const iLen : Int) :Str
2399: Func S_DecryptCRC32( const crc :Str; StartKey, MultKey, AddKey : Int): Int
2400: Func S_DirExists( const ADir :Str) :Bool
2401: Func S_Empty( const cStr :Str) :Bool
2402: Func S_EncryptCRC32(const crc:LongWORD;StartKey,MultKey,AddKey:Int):str
2403: Func S_LargeFontsActive :Bool
2404: Func S_LimitDigits( AValue : Extended; ANumDigits : Int) : Extended
2405: Func S_LTrim( const cStr :Str) :Str
2406: Func S_ReadNextTextLineFromStream( stream : TStream) :Str
2407: Func S_RepeatChar( const iLen : Int; const AChar : Char) :Str
2408: Func S_ReplFirst( const cAT, cStr, cRepl :Str) :Str
2409: Func S_RoundDecimal( AValue : Extended; APlaces : Int) : Extended
2410: Func S_RTrim( const cStr :Str) :Str
2411: Func S_RTrimCopy( const cStr :Str; iPos, iLen : Int) :Str
2412: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2413: Func S_ShellExecute(aFilename:str;aParameters:str;aCommand:TS_ShellExecuteCmd):str
2414: Func S_Space( const iLen : Int) :Str
2415: Func S_StrBlanks( const cStr :Str; const iLen : Int) :Str
2416: Func S_StrBlanksCuttooLong( const cStr :Str; const iLen : Int) :Str
2417: Func S_StrCRC32( const Text :Str) : LongWORD
2418: Func S_StrDecrypt96(const InString:Str;StartKey,MultKey,AddKey:Int):str
2419: Func S_StrEncrypt96(const InString:Str;StartKey,MultKey,AddKey:int):Str
2420: Func S_StringtoUTF_8( const AString :Str) :Str
2421: Func S_StrLBlanks( const cStr :Str; const iLen : Int) :Str
2422: Func S_StrToReal(const cStr:Str; var R: Double):Bool
2423: Func S_TokenEnd( cBuffer : PChar; lEmptyToken :Bool) :Bool
2424: Func S_TokenNext( cBuffer : PChar; lEmptyToken :Bool):Str
2425: Func S_UTF_8ToString( const AString :Str) :Str
2426: Func S_WBox( const AText :Str) : Int
2427: Func SameDate( const A, B : TDateTime) :Bool
2428: Func SameDate(const A, B: TDateTime):Bool;
2429: Func SameDateTime( const A, B : TDateTime) :Bool
2430: Func SameDateTime(const A, B: TDateTime):Bool;
2431: Func SameFileName( S1, S2 :Str) :Bool
2432: Func SameText( S1, S2 :Str) :Bool
2433: Func SameText(const S1:Str; const S2:Str):Bool)
2434: Func SameTime( const A, B : TDateTime) :Bool
2435: Func SameTime(const A, B: TDateTime):Bool;
2436: Func SameValue(const A, B: Extended; Epsilon: Extended):Bool //overload;
2437: Func SameValue1(const A, B: Double; Epsilon: Double):Bool //overload;
2438: Func SameValue2(const A, B: Single; Epsilon: Single):Bool //overload;
2439: Func SampleVariance( const X : TDynFloatArray) : Float
2440: Func Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2441: Func Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2442: Func Sar2( const Value : Int; const Count : TBitRange) : Int;
2443: Func SaveToFile( const AFileName : TFileName) :Bool
2444: Func SaveAsExcelFile(AGrid:TStringGrid;ASheetName,AFileName:str;open:bool):Bool;
2445: Func SaveAsExcel(aGrid:TStringGrid;aSheetName,aFileName:str;openexcel:boolean):Bool;
2446: Func ScanF(const aformat:Str; const args: array of const):Str;
2447: Func SCREENTOCLIENT(POINT:TPOINT):TPOINT
2448: Func SearchBuf(Buf:PChar;BufLen:Int;SelStart,
      SelLength:Int;SearchString:str;Options:TStringSearchOptions):PChar
2449: Func SearchBuf2(Buf:str;SelStart,SelLength:Int;SearchString:str;Options:TStringSearchOptions):Int;
2450: Func SearchRecattr: Int;
```

```
2451: Func SearchRecExcludeAttr: Int;
2452: Func SearchRecFileSize64( const SearchRec: TSearchRec):Int64
2453: Func SearchRecname:Str;
2454: Func SearchRecsize: Int;
2455: Func SearchRecTime: Int;
2456: Func Sec( const X : Extended) : Extended
2457: Func Secant( const X : Extended) : Extended
2458: Func SecH( const X : Extended) : Extended
2459: Func SecondOf( const AValue : TDateTime) : Word
2460: Func SecondOfTheDay( const AValue : TDateTime) : LongWord
2461: Func SecondOfTheHour( const AValue : TDateTime) : Word
2462: Func SecondOfTheMinute( const AValue : TDateTime) : Word
2463: Func SecondOfTheMonth( const AValue : TDateTime) : LongWord
2464: Func SecondOfTheWeek( const AValue : TDateTime) : LongWord
2465: Func SecondOfTheYear( const AValue : TDateTime) : LongWord
2466: Func SecondsBetween( const ANow, AThen : TDateTime) : Int64
2467: Func SecondSpan( const ANow, AThen : TDateTime) : Double
2468: Func SectionExists( const Section :Str) :Bool
2469: Func Seek( const KeyValues : Variant; SeekOption : TSeekOption) :Bool
2470: Func Seek(dlibMove: Longint; dwOrigin: Longint; out libNewPosition : Largeint):HResult
2471: Func Seek(Offset:LongInt;Origin:Word):LongInt
2472: Func SelectDirectory(var Directory:str; Options:TSelectDirOpts;HelpCtx:Longint):Bool;
2473: Func SelectDirectory1(const Caption:str;const Root:WideString;var Directory:str;
      Options:TSelectDirExtOpts;Parent:TWinControl):Bool;
2474: Func SelectImage(var AFileName:str;const Extensions,Filter:str):Boolean
2475: Func SendAppMessage(Msg:Card; WParam, LParam: Longint): Longint
2476: Func SendBuf( var Buf, Count : Int) : Int
2477: Func SendCmd( const AOut :Str; const AResponse : SmallInt) : SmallInt;
2478: Func SendCmd1( const AOut :Str; const AResponse : array of SmallInt) : SmallInt;
2479: Func SendKey( AppName :Str; Key : Char) :Bool
2480: Func SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint):Bool;
2481: Func SendStream( AStream : TStream) :Bool
2482: Func SendStreamThenDrop( AStream : TStream) :Bool
2483: Func SendText( const S :Str) : Int
2484: Func SendSerialData(Data: TByteArray; DataSize:Card):Card
2485: Func SendSerialText(Data:Str):Card
2486: Func Sent :Bool
2487: Func ServicesFilePath:Str
2488: Func SetAlpha( const Color32 : TColor32; NewAlpha : Int) : TColor32
2489: Func SetBit( const Value:Byte; const Bit : TBitRange) : Byte;
2490: Func SetBit1(const Value:Shortint; const Bit : TBitRange) : Shortint;
2491: Func SetBit2(const Value:Smallint; const Bit : TBitRange) : Smallint;
2492: Func SetBit3(const Value:Word; const Bit : TBitRange) : Word;
2493: Func SetBit4(const Value:Card; const Bit : TBitRange) Cardinal;
2494: Func SetBit4(const Value:Int; const Bit : TBitRange) : Int;
2495: Func SetBit5(const Value:Int64; const Bit : TBitRange) : Int64;
2496: Func SetClipboard( NewClipboard : TClipboard) : TClipboard
2497: Func SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2498: Func SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2499: Func SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2500: Func SetColorRed( const Color : TColor; const Red : Byte) : TColor
2501: Func SetCurrentDir( Dir :Str) :Bool
2502: Func SetCurrentDir(const Dir:Str):Bool)
2503: Func SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2504: Func SetDirCreation( const DirName :Str; const DateTime : TDateTime):Bool
2505: Func SetDirLastAccess(const DirName:str;const DateTime: TDateTime) : Bool
2506: Func SetDirLastWrite(const DirName:str; const DateTime : TDateTime): Bool
2507: Func SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2508: Func SetEndOfFile(Handle: Int): LongBool; stdcall;
2509: Func SetEnvironmentVar( const Name, Value :Str) :Bool
2510: Func SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2511: Func SetFileCreation(const FileName:str; const DateTime : TDateTime): Bool
2512: Func SetFileLastAccess( const FileName:str; const DateTime:TDateTime): Bool
2513: Func SetFileLastWrite(const FileName:str;const DateTime:TDateTime):BoolFunc SetFileTimeStamp(const
      FileName:str;TimeStamp:Int):Bool
2514: Func SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2515: Func SetLocalTime( Value : TDateTime) :Bool
2516: Func SetPrecisionTolerance( NewTolerance : Float) : Float
2517: Func SetPrinter( NewPrinter : TPrinter) : TPrinter
2518: Func SetPrivilege(privilegeName:Str; enable:Bool):Bool;
2519: Func SetRGBValue( const Red, Green, Blue : Byte) : TColor
2520: Func SetSequence( S, Localizar, Substituir : shortstring): shortstring
2521: Func SetSize( libNewSize : Longint) : HResult
2522: Func SetUserObjectFullAccess( hUserObject : THandle) :Bool
2523: Func Sgn( const X : Extended) : Int
2524: Func SHA1(const fileName:Str):Str;
2525: Func SHA256(astr:Str; amode: char):Str)
2526: Func SHA512(astr:Str; amode: char):Str)
2527: Func ShareMemoryManager :Bool
2528: Func ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:str;ShowCmd:Int):Int;stdcall;
2529: Func Shellexecute2(hwnd: HWND; const FileName:Str):Int; stdcall;
2530: Func ShellExecute3(aFilename:str;aParameters:str;aCommand:TS_ShellExecuteCmd):str;
2531: Func SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE): TSHORTCUT
2532: Func SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT):Str
2533: Func ShortDateFormat:str;
2534: Func ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2535: Func ShortTimeFormat:Str;
2536: Func SHOWMODAL:Int
```

```
2537: Func
      ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:TColor;BW:Int;Title:str;
2538: Func ShowModalPanel(aPnl:TCustomPanel;Titl:str;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2539: Func ShowWindow(C1: HWND; C2: Int):Bool;
2540: Proc ShowMemory      //in Dialog
2541: Func ShowMemory2:Str;
2542: Func ShutDownOS :Bool
2543: Func Signe( const X, Y : Extended) : Extended
2544: Func Sign( const X : Extended) : Int
2545: Func Sin(e : Extended) : Extended;
2546: //Assuming trigonometric arguments in degrees //| Use instead
2547: PrintF('in deg: %.18f',[sin(degtorad(60.0))]);  //0.866025403784438647
2548: Func sinc( const x : Double) : Double
2549: Func SinJ( X : Float) : Float
2550: Func Size( const AFileName :Str) : Int
2551: Func SizeOf: Longint;
2552: Func SizeofResource( ModuleHandle : HMODULE; ResHandle: TResourceHandle): Int
2553: Func SlashSep(const Path, S:Str):Str
2554: Func SLNDepreciation( const Cost, Salvage : Extended; Life : Int) : Extended
2555: Func SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2556: Func SmallPoint(X, Y: Int): TSmallPoint)
2557: Func Soundex( const AText :Str; ALength : TSoundexLength) :Str
2558: Func SoundexCompare( const AText, AOther :Str; ALength : TSoundexLength) : Int
2559: Func SoundexInt( const AText :Str; ALength : TSoundexIntLength) : Int
2560: Func SoundexProc( const AText, AOther :Str) :Bool
2561: Func SoundexSimilar(const AText,AOther:str;ALength:TSoundexLength):Boolean
2562: Func SoundexWord( const AText :Str) : Word
2563: Func SourcePos : Longint
2564: Func SourcePos:LongInt
2565: Func Split0( Str :Str; const substr :Str) : TStringList
2566: Proc SplitNameValue( const Line :Str; var Name, Value :Str)
2567: Func SQLRequiresParams( const SQL : WideString) :Bool
2568: Func Sqr(e : Extended) : Extended;
2569: Func Sqrt(e : Extended) : Extended;
2570: Func StartIP :Str
2571: Func StartPan( WndHandle : THandle; AControl : TControl) :Bool
2572: Func StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2573: Func StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2574: Func StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2575: Func StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek:Word):TDateTime
2576: Func StartOfAYear( const AYear : Word) : TDateTime
2577: Func StartOfTheDay( const AValue : TDateTime) : TDateTime
2578: Func StartOfTheMonth( const AValue : TDateTime) : TDateTime
2579: Func StartOfTheWeek( const AValue : TDateTime) : TDateTime
2580: Func StartOfTheYear( const AValue : TDateTime) : TDateTime
2581: Func StartsStr( const ASubText, AText :Str) :Bool
2582: Func StartsText( const ASubText, AText :Str) :Bool
2583: Func StartsWith( const ANSIStr, APattern :Str) :Bool
2584: Func StartsWith( const str :Str; const sub :Str) :Bool
2585: Func StartsWithACE( const ABytes : TIdBytes) :Bool
2586: Func StatusString( StatusCode : Int) :Str
2587: Func StdDev( const Data : array of Double) : Extended
2588: Func Stop : Float
2589: Func StopCount( var Counter : TJclCounter) : Float
2590: Func StoreColumns :Bool
2591: Func StrAfter( const sString :Str; const sDelimiters :Str):Str;
2592: Func StrAfter1(const sString:str; const sDelimiters:str;out cDelimiter:char):str;
2593: Func StrAlloc( Size :Card) : PChar
2594: Func StrAlloc(Size:Card): PChar)
2595: Func StrBefore( const sString :Str; const sDelimiters :Str) :Str;
2596: Func StrBefore1(const sString:str;const sDelimiters:str;out cDelimiter:char):str;
2597: Func StrBufSize( Str : PChar) :Card
2598: Func StrBufSize(const Str: PChar):Card)
2599: Func StrByteType( Str : PChar; Index :Card) : TMbcsByteType
2600: Func StrByteType(Str: PChar; Index:Card): TMbcsByteType)
2601: Func StrCat( Dest : PChar; Source : PChar) : PChar
2602: Func StrCat(Dest: PChar; const Source: PChar): PChar)
2603: Func StrCharLength( Str : PChar) : Int
2604: Func StrComp( Str1, Str2 : PChar) : Int
2605: Func StrComp(const Str1: PChar; const Str2: PChar): Int)
2606: Func StrCopy( Dest : PChar; Source : PChar) : PChar
2607: Func StrCopy(Dest: PChar; const Source: PChar): PChar)
2608: Func Stream_to_Ansistr( Source : TStream) : Ansistr
2609: Func Stream_to_Base64( Source : TStream) : Ansistr
2610: Func Stream_to_decimalbytes( Source : TStream) :Str
2611: Func Stream2WideString( oStream : TStream) : WideString
2612: Func StreamtoAnsistr( Source : TStream) : Ansistr
2613: Func StreamToByte( Source : TStream) :Str
2614: Func StreamToDecimalbytes( Source : TStream) :Str
2615: Func StreamtoOrd( Source : TStream) :Str
2616: Func StreamToString( Source: TStream) :Str
2617: Func StreamToString2(Source: TStream) :Str
2618: Func StreamToString3(Source: TStream) :Str
2619: Func StreamToString4(Source: TStream) :Str
2620: Func StrECopy( Dest : PChar; Source : PChar) : PChar
2621: Func StrEmpty( const sString :Str) :Bool
2622: Func StrEnd( Str : PChar) : PChar
2623: Func StrEnd(const Str: PChar): PChar)
2624: Func StrFilter( const sString :Str; xValidChars : TCharSet) :Str
```

```
2625: Func StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2626: Func StrGet(var S :Str; I : Int) : Char;
2627: Func StrGet2(S :Str; I : Int) : Char;
2628: Func StrHasPrefix( const sString :Str; const sPrefix :Str) :Bool
2629: Func StrHasSuffix( const sString :Str; const sSuffix :Str) :Bool
2630: Func StrHtmlDecode( const AStr :Str) :Str
2631: Func StrHtmlEncode( const AStr :Str) :Str
2632: Func StrToBytes(const Value:Str): TBytes;
2633: Func StrIComp( Str1, Str2 : PChar) : Int
2634: Func StringOfChar(c : char;I : longInt) :Str;
2635: Func StringOfChar2( ch : WideChar; Count : Int) : WideString;
2636: Func StringPad(InputStr,FillChar:Str; StrLen:Int; StrJustify:Boolean):Str;
2637: Func StringRefCount(const s:Str): Int;
2638: Func StringReplace(S,OldPattern,NewPattern:str;Flags:TReplaceFlags):str
2639: Func JStringReplace(const S,OldPattern, NewPattern:Str; Flags : TReplaceFlags):str
2640: Func StringReplace(const SourceString,OldPattern,NewPattern:str;Flags:TReplaceFlags):str;
2641: Func StringRemove( const S, Pattern :Str; Flags : TReplaceFlags) :Str
2642: Func StringToBoolean( const Ps :Str) :Bool
2643: Func StringToColor(const S:Str): TColor)
2644: Func StringToCursor(const S:Str): TCursor;
2645: Func StringToGUID(const S:Str): TGUID)
2646: Func StringTokenizer( const str :Str; const delim :Str) : IStringTokenizer
2647: Func StringToStringArray( const str :Str; const delim:Str): TStringDynArray
2648: Func StringWidth( S :Str) : Int
2649: Func StrInternetToDateTime( Value :Str) : TDateTime
2650: Func StrIsDateTime( const Ps :Str) :Bool
2651: Func StrIsFloatMoney( const Ps :Str) :Bool
2652: Func StrIsInt( const S :Str) :Bool
2653: Func StrLCat( Dest : PChar; Source : PChar; MaxLen :Card) : PChar
2654: Func StrLComp( Str1, Str2 : PChar; MaxLen :Card) : Int
2655: Func StrLCopy( Dest : PChar; Source : PChar; MaxLen :Card) : PChar
2656: Func StrLen( Str : PChar) :Card
2657: Func StrLen(const Str: PChar):Card)
2658: Func StrLessPrefix( const sString :Str; const sPrefix:Str):Str
2659: Func StrLessSuffix( const sString :Str; const sSuffix:Str):Str
2660: Func StrLIComp( Str1, Str2 : PChar; MaxLen :Card) : Int
2661: Func StrLower( Str : PChar) : PChar
2662: Func StrMove( Dest : PChar; Source : PChar; Count :Card) : PChar
2663: Func StrMove(Dest: PChar; const Source: PChar; Count:Card): PChar)
2664: Func StrNew( Str : PChar) : PChar
2665: Func StrNew(const Str: PChar): PChar)
2666: Func StrNextChar( Str : PChar) : PChar
2667: Func StrPad( const sString:str;const sPad:str;const iLength:Int):Str
2668: Func StrParse( var sString :Str; const sDelimiters :Str) :Str;
2669: Func StrParse1(var sString:str;const sDelimiters:str;out cDelimiter:char) string;
2670: Func StrPas( Str : PChar) :Str
2671: Func StrPas(const Str: PChar):Str)
2672: Func StrPCopy( Dest : PChar; Source :Str) : PChar
2673: Func StrPCopy(Dest: PChar; const Source:Str): PChar)
2674: Func StrPLCopy( Dest : PChar; Source :Str; MaxLen :Card) : PChar
2675: Func StrPos( Str1, Str2 : PChar) : PChar
2676: Func StrScan(const Str: PChar; Chr: Char): PChar)
2677: Func StrRScan(const Str: PChar; Chr: Char): PChar)
2678: Func StrToBcd( const AValue :Str) : TBcd
2679: Func StrToBool( S :Str) :Bool
2680: Func StrToBoolDef( S :Str; Default :Bool) :Bool
2681: Func StrToCard( const AStr :Str) :Card
2682: Func StrToConv( AText :Str; out AType : TConvType) : Double
2683: Func StrToCurr( S :Str) : Currency;
2684: Func StrToCurr(const S:Str): Currency)
2685: Func StrToCurrDef( S :Str; Default : Currency) : Currency;
2686: Func StrToDate( S :Str) : TDateTime;
2687: Func StrToDate(const s:Str): TDateTime;
2688: Func StrToDateDef( S :Str; Default : TDateTime) : TDateTime;
2689: Func StrToDateTime( S :Str) : TDateTime;
2690: Func StrToDateTime(const S:Str): TDateTime)
2691: Func StrToDateTimeDef( S :Str; Default : TDateTime) : TDateTime;
2692: Func StrToDay( const ADay :Str) : Byte
2693: Func StrToFloat( S :Str) : Extended;
2694: Func StrToFloat(s:Str): Extended;
2695: Func StrToFloatDef( S :Str; Default : Extended) : Extended;
2696: Func StrToFloatDef(const S:Str; const Default: Extended): Extended)
2697: Func StrToFloat( S :Str) : Extended;
2698: Func StrToFloat2( S :Str; FormatSettings : TFormatSettings) : Extended;
2699: Func StrToFloatDef( S :Str; Default : Extended) : Extended;
2700: Func StrToFloatDef2(S:str;Default:Extended;FormatSettings:TFormatSettings):Extended;
2701: Func StrToCurr( S :Str) : Currency;
2702: Func StrToCurr2( S :Str; FormatSettings : TFormatSettings) : Currency;
2703: Func StrToCurrDef( S :Str; Default : Currency) : Currency;
2704: Func StrToCurrDef2(S:str; Default:Currency; FormatSettings:TFormatSettings):Currency;
2705: Func StrToTime2( S :Str; FormatSettings : TFormatSettings): TDateTime;
2706: Func StrToTimeDef( S :Str; Default : TDateTime) : TDateTime;
2707: Func StrToTimeDef2(S:str; Default:TDateTime;FormatSettings:TFormatSettings):TDateTime;
2708: Func TryStrToTime( S :Str; Value : TDateTime) :Bool;
2709: Func StrToDateTime( S :Str) : TDateTime;
2710: Func StrToDateTime2( S :Str; FormatSettings : TFormatSettings) : TDateTime;
2711: Func StrToDateTimeDef( S :Str; Default : TDateTime) : TDateTime;
2712: Func StrToFloatRegionalIndependent(aValue:str;aDecimalSymbol:Char;aDigitGroupSymbol:Char):Extended
2713: Func StrToInt( S :Str) : Int
```

```
2714: Func StrToInt(s:Str): Longint;
2715: Func StrToInt64( S :Str) : Int64
2716: Func StrToInt64(s:Str): int64;
2717: Func StrToInt64Def( S :Str; Default : Int64) : Int64
2718: Func StrToInt64Def(const S:Str; const Default: Int64):Int64)
2719: Func StrToIntDef( S :Str; Default : Int) : Int
2720: Func StrToIntDef(const S:Str; Default: Int): Int)
2721: Func StrToIntDef(s:Str; def: Longint): Longint;
2722: Func StrToMonth( const AMonth :Str) : Byte
2723: Func StrToTime( S :Str) : TDateTime;
2724: Func StrToTime(const S:Str): TDateTime)
2725: Func StrToTimeDef( S :Str; Default : TDateTime) : TDateTime;
2726: Func StrToWord( const Value :Str) : Word
2727: Func StrToXmlDate( const DateStr :Str; const Format :Str) :Str
2728: Func StrToXmlDateTime( const DateStr :Str; const Format :Str) :Str
2729: Func StrToXmlTime( const TimeStr :Str; const Format :Str) :Str
2730: Func StrUpper( Str : PChar) : PChar
2731: Func StuffString(const AText:str;AStart, ALength:Card;const ASubText:str):str
2732: Func Sum( const Data : array of Double) : Extended
2733: Func SumFloatArray( const B : TDynFloatArray) : Float
2734: Func SumInt( const Data : array of Int) : Int
2735: Func SumOfSquares( const Data : array of Double) : Extended
2736: Func SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2737: Func SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2738: Func SumSquareFloatArray( const B : TDynFloatArray) : Float
2739: Func Supports( CursorOptions : TCursorOptions) :Bool
2740: Func SupportsClipboardFormat( AFormat : Word) :Bool
2741: Func SwapWord(w : word): word)
2742: Func SwapInt(i : Int): Int)
2743: Func SwapLong(L : longint): longint)
2744: Func Swap(i : Int): Int)
2745: Func SYDDepreciation(const Cost,Salvage:Extended;Life,Period:Int): Extended
2746: Func SyncTime :Bool
2747: Func SysErrorMessage( ErrorCode : Int) :Str
2748: Func SysErrorMessage(ErrorCode: Int):Str)
2749: Func SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2750: Func SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2751: Func SysStringLen(const S: WideString): Int; stdcall;
2752: Func TabRect( Index : Int) : TRect
2753: Func Tan( const X : Extended) : Extended
2754: Func TaskMessageDlg(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint):Int;
2755: Func TaskMessageDlg1(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;
      HelpCtx:Longint;DefaultButton:TMsgDlgBtn):Int;
2756: Func TaskMessageDlgPos(const Title,Msg:str;DlgType:TMsgDlgType;Bts:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Int):Int;
2757: Func TaskMessageDlgPos1(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Int;DefaultButton:TMsgDlgBtn):Int;
2758: Func TaskMessageDlgPosHelp(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:
      TMsgDlgButtons;HelpCtx:Longint;X,Y:Int;const HelpFileName:str): Int;
2759: Func TaskMessageDlgPosHelp1(const Title, Msg:str;DlgType: TMsgDlgType; Buttons :
      TMsgDlgButtons;HelpCtx:Longint;X,Y:Int;const HelpFileName:str;DefaultButton:TMsgDlgBtn):Int;
2760: Func TenToY( const Y : Float) : Float
2761: Func TerminateApp( ProcessID : DWORD; Timeout : Int) : TJclTerminateAppResult
2762: Func TerminateTask( Wnd : HWND; Timeout : Int) : TJclTerminateAppResult
2763: Func TestBit( const Value:Byte; const Bit : TBitRange) : Bool;
2764: Func TestBit2(const Value:Shortint; const Bit : TBitRange): Bool;
2765: Func TestBit3(const Value:Smallint; const Bit : TBitRange): Bool;
2766: Func TestBit4(const Value:Word; const Bit : TBitRange):Bool;
2767: Func TestBit5(const Value:Card; const Bit : TBitRange): Bool;
2768: Func TestBit6(const Value:Int; const Bit : TBitRange): Bool;
2769: Func TestBit7(const Value:Int64; const Bit : TBitRange): Bool;
2770: Func TestBits(const Value, Mask:Byte) :Bool;
2771: Func TestBits1(const Value,Mask:Shortint) :Bool;
2772: Func TestBits2(const Value,Mask:Smallint) :Bool;
2773: Func TestBits3(const Value,Mask:Word) :Bool;
2774: Func TestBits4(const Value,Mask:Card) :Bool;
2775: Func TestBits5(const Value,Mask:Int) :Bool;
2776: Func TestBits6(const Value,Mask:Int64) :Bool;
2777: Func TestFDIVInstruction :Bool
2778: Func TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2779: Func TextExtent( const Text :Str) : TSize
2780: Func TextHeight(Text:Str): Int;
2781: Func TextIsSame( const A1 :Str; const A2 :Str) :Bool
2782: Func TextStartsWith( const S, SubS :Str) :Bool
2783: Func TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue):Boolean)
2784: Func ConvInt(i : Int):str;
2785: Func IntToText(i : Int):str;
2786: Func TEXTTOSHORTCUT( TEXT :Str) : TSHORTCUT
2787: Func TextWidth(Text:Str): Int;
2788: Func ThreadCount : Int
2789: Func ThousandSeparator: char;
2790: Func Ticks :Card
2791: Func Time : TDateTime
2792: Func Time: TDateTime;
2793: Func TimeGetTime: int64;
2794: Func TimeOf( const AValue:TDateTime):TDateTime
2795: Func TimeSeparator: char;
2796: Func TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2797: Func TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
```

```
2798: Func TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2799: Func TimeToStr( DateTime : TDateTime) :Str;
2800: Func TimeToStr(const DateTime: TDateTime):Str;
2801: Func TimeZoneBias : TDateTime
2802: Func ToCommon( const AValue : Double) : Double
2803: Func ToCommon(const AValue: Double): Double;
2804: Func Today : TDateTime
2805: Func ToggleBit( const Value:Byte; const Bit : TBitRange) : Byte;
2806: Func ToggleBit1(const Value:Shortint; const Bit : TBitRange): Shortint;
2807: Func ToggleBit2(const Value:Smallint; const Bit : TBitRange): Smallint;
2808: Func ToggleBit3(const Value:Word; const Bit : TBitRange) : Word;
2809: Func ToggleBit4(const Value:Card; const Bit : TBitRange):Card;
2810: Func ToggleBit5(const Value:Int; const Bit : TBitRange) : Int;
2811: Func ToggleBit6(const Value:Int64; const Bit : TBitRange) : Int64;
2812: Func TokenComponentIdent:str
2813: Func TokenFloat : Extended
2814: Func TokenFloat:Extended
2815: Func TokenInt : Longint
2816: Func TokenInt:LongInt
2817: Func TokenString :Str
2818: Func TokenString:str
2819: Func TokenSymbolIs( const S :Str) :Bool
2820: Func TokenSymbolIs(S:str):Boolean
2821: Func Tomorrow : TDateTime
2822: Func ToRightOf( const pc : TControl; piSpace : Int) : Int
2823: Func ToString :Str
2824: Func TotalVariance( const Data : array of Double) : Extended
2825: Func Trace2( AURL :Str) :Str;
2826: Func TrackMenu( Button : TToolButton) :Bool
2827: Func TRANSLATE( SRC, DEST : PCHAR; TOOEM :Bool) : Int
2828: Func TranslateURI( const URI :Str) :Str
2829: Func TranslationMatchesLanguages( Exact :Bool) :Bool
2830: Func TransparentStretchBlt(DstDC:HDC;DstX,DstY,DstW,DstH:Int;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Int;MaskDC:HDC;MaskX, MaskY:Int):Bool
2831: Func Trim( S :Str) :Str;
2832: Func Blank( S :Str) :Str; //alias to Trim
2833: Func Trim( S : WideString) : WideString;
2834: Func Trim(s : AnyString) : AnyString;
2835: Func TrimAllOf( ATrim, AText :Str):str
2836: Func TrimLeft( S :Str) :Str;
2837: Func TrimLeft( S : WideString) : WideString;
2838: Func TrimLeft(const S:Str):Str)
2839: Func TrimRight( S:Str) :Str;
2840: Func TrimRight( S: WideString) : WideString;
2841: Func TrimRight(const S:Str):Str)
2842: Func TrueBoolStrs: array of string
2843: Func Trunc(e : Extended) : Longint;
2844: Func Trunc64(e: extended): Int64;
2845: Func TruncPower( const Base, Exponent : Float) : Float
2846: Func TryConvTypeToFamily(const AFrom,ATo:TConvType; out AFamily:TConvFamily):Bool;
2847: Func TryConvTypeToFamily1(const AType:TConvType;out AFamily:TConvFamily):Bool;
2848: Func TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime):Bool;
2849: Func TryEncodeDateDay(const AYear,ADayOfYear:Word out AValue:TDateTime): Bool
2850: Func TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime):Bool
2851: Func TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word;out AValue:TDateTime):Bool
2852: Func TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word):Bool
2853: Func TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out AVal:TDateTime):Bool
2854: Func TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime):Bool;
2855: Func TryFloatToDateTime( Value : Extended; AResult : TDateTime) :Bool
2856: Func TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) :Bool
2857: Func TryLock :Bool
2858: Func TryModifiedJulianDateToDateTime(const AValue:Double;out ADateTime:TDateTime):Boolean
2859: Func TryRecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHour,AMinute,ASecond,
      AMilliSecond:Word; out AResult:TDateTime):Boolean
2860: Func TryStrToBcd( const AValue :Str; var Bcd : TBcd) :Bool
2861: Func TryStrToConv( AText :Str; out AValue : Double; out AType : TConvType) :Bool
2862: Func TryStrToDate( S :Str; Value : TDateTime) :Bool;
2863: Func TryStrToDateTime( S :Str; Value : TDateTime) :Bool;
2864: Func TryStrToTime( S :Str; Value : TDateTime) :Bool;
2865: Func TryStrToInt(const S: Ansistr; var I: Int):Bool;
2866: Func TryStrToInt64(const S: Ansistr; var I: Int64):Bool;
2867: Func TryStrToBool(const S:Str; out Value:Bool):Bool;
2868: Func TwoByteToWord( AByte1, AByte2 : Byte) : Word
2869: Func TwoCharToWord( AChar1, AChar2 : Char) : Word
2870: Func TwoToY( const Y : Float) : Float
2871: Func UCS4StringToWideString( const S : UCS4String) : WideString
2872: Func UIDL( const ADest : TStrings; const AMsgNum : Int) :Bool
2873: Func Unassigned: Variant;
2874: Func UndoLastChange( FollowChange :Bool) :Bool
2875: Func UniCodeToStr(Value:Str):Str;
2876: Func UnionRect( out Rect : TRect; const R1, R2 : TRect) :Bool
2877: Func UnionRect(out Rect: TRect; const R1, R2: TRect):Bool)
2878: Func UnixDateTimeToDelphiDateTime( UnixDateTime :Card) : TDateTime
2879: Func UnixPathToDosPath( const Path :Str) :Str
2880: Func UnixToDateTime( const AValue : Int64) : TDateTime
2881: Func UnixToDateTime(U: Int64): TDateTime;
2882: Func UnlockRegion(libOffset Longint;cb: Largeint;dwLockType:Longint):HResult
2883: Func UnlockResource( ResData : HGLOBAL) : LongBool
2884: Func UnlockVolume( var Handle : THandle) :Bool
```

```
2885: Func UnMaskString( Mask, Value :Str) :Str
2886: Func UpCase(ch : Char ) : Char;
2887: Func UpCaseFirst( const AStr :Str) :Str
2888: Func UpCaseFirstWord( const AStr :Str) :Str
2889: Func UpdateAction( Action : TBasicAction) :Bool
2890: Func UpdateKind : TUpdateKind
2891: Func UPDATESTATUS : TUPDATESTATUS
2892: Func UpperCase( S :Str) :Str
2893: Func Uppercase(s : AnyString) : AnyString;
2894: Func URLDecode( ASrc :Str) :Str
2895: Func URLEncode( const ASrc :Str) :Str
2896: Func UseRightToLeftAlignment :Bool
2897: Func UseRightToLeftAlignmentForField(const AField:TField;Alignment:TAlignment):Bool
2898: Func UseRightToLeftReading :Bool
2899: Func UTF8CharLength( Lead : Char) : Int
2900: Func UTF8CharSize( Lead : Char) : Int
2901: Func UTF8Decode( const S : UTF8String) : WideString
2902: Func UTF8Encode( const WS : WideString) : UTF8String
2903: Func UTF8LowerCase( const S : UTF8string) : UTF8string
2904: Func Utf8ToAnsi( const S : UTF8String) :Str
2905: Func Utf8ToAnsiEx( const S : UTF8String; const cp : Int) :Str
2906: Func UTF8UpperCase( const S : UTF8string) : UTF8string
2907: Func ValidFieldIndex( FieldIndex : Int) :Bool
2908: Func ValidParentForm(control: TControl): TForm
2909: Func Value : Variant
2910: Func ValueExists( const Section, Ident :Str) :Bool
2911: Func ValueOf( const Key :Str) : Int
2912: Func ValueInSet(AValue: Variant; ASet: Variant):Bool;
2913: Func VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2914: Func VarArrayFromStrings( Strings : TStrings) : Variant
2915: Func VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2916: Func VarArrayGet(var S : Variant; I : Int) : Variant;
2917: Func VarFMTBcd : TVarType
2918: Func VarFMTBcdCreate1 : Variant;
2919: Func VarFMTBcdCreate2(const AValue :Str; Precision, Scale : Word) : Variant;
2920: Func VarFMTBcdCreate3(const AValue: Double;Precision: Word; Scale: Word):Variant;
2921: Func VarFMTBcdCreate4(const ABcd : TBcd) : Variant;
2922: Func Variance(const Data : array of Double) : Extended
2923: Func VariantAdd2( const V1 : Variant; const V2 : Variant):Variant
2924: Func VariantAnd2( const V1 : Variant; const V2 : Variant):Variant
2925: Func VariantDiv2( const V1 : Variant; const V2 : Variant):Variant
2926: Func VariantGetElement( const V : Variant; i1 : Int) : Variant;
2927: Func VariantGetElement1( const V : Variant; i1, i2 : Int) : Variant;
2928: Func VariantGetElement2( const V : Variant; i1, i2, i3 : Int) : Variant;
2929: Func VariantGetElement3( const V : Variant; i1, i2, i3, i4 : Int):Variant;
2930: Func VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5: Int):Variant;
2931: Func VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2932: Func VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2933: Func VariantNeg(const V1: Variant) : Variant
2934: Func VariantNot(const V1: Variant) : Variant
2935: Func VariantOr2(const V1: Variant; const V2 : Variant) :Variant
2936: Func VariantShl2(const V1: Variant; const V2 : Variant):Variant
2937: Func VariantShr2(const V1: Variant; const V2 : Variant):Variant
2938: Func VariantSub2(const V1: Variant; const V2 : Variant):Variant
2939: Func VariantXor2(const V1: Variant; const V2 : Variant):Variant
2940: Func VarIsEmpty(const V: Variant):Bool;
2941: Func VarIsFMTBcd( const AValue : Variant) :Bool;
2942: Func VarIsNull(const V: Variant):Bool;
2943: Func VarToBcd( const AValue : Variant) : TBcd
2944: Func VarType(const V: Variant): TVarType;
2945: Func VarType( const V : Variant) : TVarType
2946: Func VarAsType( const V : Variant; AVarType : TVarType): Variant
2947: Func VarIsType( const V : Variant; AVarType : TVarType):Bool;
2948: Func VarIsType1(const V: Variant;const AVarTypes: array of TVarType):Bool;
2949: Func VarIsByRef( const V : Variant) : Bool
2950: Func VarIsEmpty( const V : Variant) : Bool
2951: Proc VarCheckEmpty( const V : Variant)
2952: Func VarIsNull( const V : Variant) : Bool
2953: Func VarIsClear( const V : Variant) : Bool
2954: Func VarIsCustom( const V : Variant) : Bool
2955: Func VarIsOrdinal( const V : Variant) : Bool
2956: Func VarIsFloat( const V : Variant) : Bool
2957: Func VarIsNumeric( const V : Variant) : Bool
2958: Func VarIsStr( const V : Variant) :Bool
2959: Func VarToStr( const V : Variant) :Str
2960: Func VarToStrDef( const V : Variant; const ADefault :Str) :Str
2961: Func VarToWideStr( const V : Variant) : WideString
2962: Func VarToWideStrDef( const V : Variant; const ADefault : WideString):WideString
2963: Func VarToDateTime( const V : Variant) : TDateTime
2964: Func VarFromDateTime( const DateTime : TDateTime) : Variant
2965: Func VarInRange( const AValue, AMin, AMax : Variant) :Bool
2966: Func VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2967: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual)
2968: Func VarSameValue( const A, B : Variant) :Bool
2969: Func VarCompareValue( const A, B : Variant) : TVariantRelationship
2970: Func VarIsEmptyParam( const V : Variant) :Bool
2971: Func VarIsError( const V : Variant; out AResult : HRESULT):Boolean;
2972: Func VarIsError1( const V : Variant) :Bool;
2973: Func VarAsError( AResult : HRESULT) : Variant
```

```
2974: Proc VarCopyNoInd( var Dest : Variant; const Source : Variant)
2975: Func VarIsArray( const A : Variant) :Bool;
2976: Func VarIsArray1( const A : Variant; AResolveByRef :Bool) :Bool;
2977: Func VarArrayCreate( const Bounds : array of Int; AVarType : TVarType):Variant
2978: Func VarArrayOf( const Values : array of Variant) : Variant
2979: Func VarArrayRef( const A : Variant) : Variant
2980: Func VarTypeIsValidArrayType( const AVarType : TVarType):Bool
2981: Func VarTypeIsValidElementType( const AVarType : TVarType):Boolean
2982: Func VarArrayDimCount( const A : Variant) : Int
2983: Func VarArrayLowBound( const A : Variant; Dim : Int) : Int
2984: Func VarArrayHighBound( const A : Variant; Dim : Int) : Int
2985: Func VarArrayLock( const A : Variant) : ___Pointer
2986: Proc VarArrayUnlock( const A : Variant)
2987: Func VarArrayGet( const A : Variant; const Indices : array of Int) : Variant
2988: Proc VarArrayPut(var A:Variant;const Value:Variant;const Indices: array of Int)
2989: Proc DynArrayToVariant(var V: Variant;const DynArray: ___Pointer;TypeInfo: ___Pointer)
2990: Proc DynArrayFromVariant(var DynArray:___Pointer;const V: Variant;TypeInfo:___Pointer)
2991: Func Unassigned : Variant
2992: Func Null : Variant
2993: Func VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2994: Func VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2995: Func VectorDot( const V1, V2 : TFloatPoint) : Double
2996: Func VectorDot(const V1,V2: TFloatPoint): Double;
2997: Func VectorLengthSqr( const V : TFloatPoint) : Double
2998: Func VectorLengthSqr(const V: TFloatPoint): Double;
2999: Func VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
3000: Func VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
3001: Func VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
3002: Func VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
3003: Func Verify( AUserName :Str) :Str
3004: Func Versine( X : Float) : Float
3005: Func VersionCheck:Bool;
3006: Func VersionCheckAct:Str;
3007: Func VersionLanguageId( const LangIdRec : TLangIdRec) :Str
3008: Func VersionLanguageName( const LangId : Word) :Str
3009: Func VersionResourceAvailable( const FileName :Str):Boolean
3010: Func Visible :Bool
3011: Func VolumeID(DriveChar: Char):Str
3012: Func WaitFor( const AString :Str) :Str
3013: Func WaitFor( const TimeOut :Card): TJclWaitResult
3014: Func WaitFor1 : TWaitResult;
3015: Func WaitForData( Timeout : Longint) :Bool
3016: Func WebColorNameToColor( WebColorName :Str) : TColor
3017: Func WebColorStrToColor( WebColor :Str) : TColor
3018: Func WebColorToRGB( WebColor : Int) : Int
3019: Func wGet(aURL, afile:Str):Bool;'
3020: Func wGet2(aURL, afile:Str):Bool;'  //without file open
3021: Func wGetX(aURL, afile:Str):Bool;
3022: Func wGetX2(aURL, afile:Str):Bool;'  //without file open
3023: Func WebGet(aURL, afile:Str):Bool;'
3024: Func WebExists:Bool;   //alias to isinternet
3025: Func WeekOf( const AValue : TDateTime) : Word
3026: Func WeekOfTheMonth( const AValue : TDateTime) : Word;
3027: Func WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
3028: Func WeekOfTheYear( const AValue : TDateTime) : Word;
3029: Func WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
3030: Func WeeksBetween( const ANow, AThen : TDateTime) : Int
3031: Func WeeksInAYear( const AYear : Word) : Word
3032: Func WeeksInYear( const AValue : TDateTime) : Word
3033: Func WeekSpan( const ANow, AThen : TDateTime): Double
3034: Func WideAdjustLineBreaks( const S : WideString; Style:TTextLineBreakStyle):WideString
3035: Func WideCat( const x, y : WideString) : WideString
3036: Func WideCompareStr( S1, S2 : WideString) : Int
3037: Func WideCompareStr(const S1: WideString; const S2: WideString): Int)
3038: Func WideCompareText( S1, S2 : WideString) : Int
3039: Func WideCompareText(const S1: WideString; const S2: WideString): Int)
3040: Func WideCopy( const src : WideString; index, count : Int) : WideString
3041: Func WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
3042: Func WideEqual( const x, y : WideString) :Bool
3043: Func WideFormat(const Format: WideString; const Args: array of const):WideString)
3044: Func WideGreater( const x, y : WideString) :Bool
3045: Func WideLength( const src : WideString) : Int
3046: Func WideLess( const x, y : WideString) :Bool
3047: Func WideLowerCase( S : WideString) : WideString
3048: Func WideLowerCase(const S: WideString): WideString)
3049: Func WidePos( const src, sub : WideString) : Int
3050: Func WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
3051: Func WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
3052: Func WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
3053: Func WideSameStr( S1, S2 : WideString) :Bool
3054: Func WideSameStr(const S1: WideString; const S2: WideString):Bool)
3055: Func WideSameText( S1, S2 : WideString) :Bool
3056: Func WideSameText(const S1: WideString; const S2: WideString):Bool)
3057: Func WideStringReplace(const S,OldPattern,NewPattern:Widestr;Flags:TReplaceFlags):Widestr
3058: Func WideStringToUCS4String( const S : WideString) : UCS4String
3059: Func WideUpperCase( S : WideString) : WideString
3060: Func Win32BackupFile( const FileName :Str; Move :Bool) :Bool
3061: Func Win32Check(RetVal:Bool):Bool)
3062: Func Win32DeleteFile(const FileName:str; MoveToRecycleBin:Boolean):Bool
```

```
3063: Func Win32RestoreFile( const FileName :Str) :Bool
3064: Func Win32Type : TIdWin32Type
3065: Func WinColor( const Color32 : TColor32) : TColor
3066: Func winexec(FileName: pchar; showCmd: Int): Int;
3067: Func WinExec32( const Cmd :Str; const CmdShow : Int) :Bool
3068: Func WinExec32AndWait( const Cmd :Str; const CmdShow : Int) :Card
3069: Func WithinPastDays( const ANow, AThen : TDateTime; const ADays : Int) : Bool
3070: Func WithinPastHours(const ANow, AThen:TDateTime; const AHours : Int64): Bool
3071: Func WithinPastMilliSeconds(const ANow,AThen:TDateTime;const AMilliSeconds:Int64):Bool
3072: Func WithinPastMinutes(const ANow,AThen:TDateTime;const AMinutes:Int64):Bool
3073: Func WithinPastMonths(const ANow, AThen:TDateTime; const AMonths:Int): Bool
3074: Func WithinPastSeconds(const ANow,AThen: DateTime;const ASeconds:Int64): Bool
3075: Func WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Int): Bool
3076: Func WithinPastYears( const ANow, AThen : TDateTime; const AYears : Int): Bool
3077: Func WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar): DWORD
3078: Func WordToStr( const Value : Word) :Str
3079: Func WordGridFormatIdentToInt(const Ident:Str; var Value:Longint): Bool
3080: Func IntToWordGridFormatIdent( Value : Longint; var Ident :Str): Bool
3081: Proc GetWordGridFormatValues( Proc : TGetStrProc)
3082: Func WorkArea : Int
3083: Func WrapText( Line :Str; MaxCol : Int) :Str;
3084: Func WrapText2(Line, BreakStr:str;BreakChars:TSysCharSet;MaxCol:Int):str;
3085: Func Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3086: Func Write(Buffer:str;Count:LongInt):LongInt
3087: Func WriteClient( var Buffer, Count : Int) : Int
3088: Func WriteFile(const AFile:str; const AEnableTransferFile:Boolean):Card
3089: Func WriteHeaders(StatusCode: Int; const ReasonString, Headers:Str):Boolean
3090: Func WriteString( const AString :Str): Bool
3091: Func WStrGet(var S : AnyString; I : Int): WideChar;
3092: Func wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Int
3093: Func wsprintf( Output : PChar; Format : PChar) : Int
3094: Func XmlDateTimeToStr( const XmlDateTime :Str; const Format:Str):str
3095: Func XmlTimeToStr( const XmlTime :Str; const Format :Str):Str
3096: Func XorDecode( const Key, Source :Str) :Str
3097: Func XorEncode( const Key, Source :Str) :Str
3098: Func XorString( const Key, Src : ShortString): ShortString
3099: Func Yield : Bool
3100: Func YearOf( const AValue : TDateTime) : Word
3101: Func YearsBetween( const ANow, AThen : TDateTime) : Int
3102: Func YearSpan( const ANow, AThen : TDateTime) : Double
3103: Func Yesterday : TDateTime
3104: Func YesNoDialog(const ACaption, AMsg:Str):Bool;
3105: Function(const Name :Str; Proc : TUserFunction)
3106: Func using Special_Scholz from V 3.8.5.0
3107: Func TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3108: Func FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3109: Func FloatToTime2Dec(value:Extended):Extended;
3110: Func MinToStd(value:Extended):Extended;
3111: Func MinToStdAsString(value:Extended):str;
3112: Func RoundFloatToStr(zahl:Extended; decimals:Int):str;
3113: Func RoundFloat(zahl:Extended; decimals:Int):Extended;
3114: Func Round2Dec (zahl:Extended):Extended;
3115: Func GetAngle(x,y:Extended):Double;
3116: Func AddAngle(a1,a2:Double):Double;
3117:
3118: ************************************************************************
3119: unit uPSI_StText;
3120: ************************************************************************
3121: Func TextSeek( var F : TextFile; Target : LongInt) :Bool
3122: Func TextFileSize( var F : TextFile) : LongInt
3123: Func TextPos( var F : TextFile) : LongInt
3124: Func TextFlush( var F : TextFile) :Bool
3125:
3126: ************************************************************************
3127: from  JvVCLUtils;
3128: ************************************************************************
3129: { Windows resources (bitmaps and icons) VCL-oriented routines }
3130: Proc DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Int;Bitmap:TBitmap;TransparentColor:TColor);
3131: Proc DrawBitmapRectTransparent(Dest: TCanvas;DstX,
      DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3132: Proc StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Int; SrcRect: TRect; Bitmap:
      TBitmap; TransparentColor:TColor);
3133: Func MakeBitmap(ResID: PChar): TBitmap;
3134: Func MakeBitmapID(ResID: Word): TBitmap;
3135: Func MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3136: Func CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3137: Func CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3138: Func CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3139:   HighlightColor, ShadowColor: TColor; DrawHighlight:Bool): TBitmap;
3140: Func CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3141: Func ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3142: Proc AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Int);
3143: {$IFDEF WIN32}
3144: Proc ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3145:   X, Y, Index: Int; HighlightColor, GrayColor: TColor; DrawHighlight:Bool);
3146: {$ENDIF}
3147: Func MakeIcon(ResID: PChar): TIcon;
3148: Func MakeIconID(ResID: Word): TIcon;
3149: Func MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
```

```
3150: Func CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3151: {$IFDEF WIN32}
3152: Func CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3153: {$ENDIF}
3154: { Service routines }
3155: Proc NotImplemented;
3156: Proc ResourceNotFound(ResID: PChar);
3157: Func PointInRect(const P: TPoint; const R: TRect):Bool;
3158: Func PointInPolyRgn(const P: TPoint; const Points: array of TPoint):Bool;
3159: Func PaletteColor(Color: TColor): Longint;
3160: Func WidthOf(R: TRect): Int;
3161: Func HeightOf(R: TRect): Int;
3162: Proc PaintInverseRect(const RectOrg, RectEnd: TPoint);
3163: Proc DrawInvertFrame(ScreenRect: TRect; Width: Int);
3164: Proc CopyParentImage(Control: TControl; Dest: TCanvas);
3165: Proc Delay(MSecs: Longint);
3166: Proc DeleteLine(StrList: TStringList; SearchPattern:Str);
3167: Proc CenterControl(Control: TControl);
3168: Func PaletteEntries( Palette : HPALETTE) : Int
3169: Func WindowClassName( Wnd : HWND) :Str
3170: Func ScreenWorkArea : TRect
3171: Proc MoveWindowOrg( DC : HDC; DX, DY : Int)
3172: Proc SwitchToWindow( Wnd : HWND; Restore :Bool)
3173: Proc ActivateWindow( Wnd : HWND)
3174: Proc ShowWinNoAnimate( Handle : HWND; CmdShow : Int)
3175: Proc CenterWindow( Wnd : HWND)
3176: Proc ShadeRect( DC : HDC; const Rect : TRect)
3177: Proc KillMessage( Wnd : HWND; Msg :Card)
3178: Func DialogsToPixelsX(Dlgs:Word):Word
3179: Func DialogsToPixelsY(Dlgs:Word):Word
3180: Func PixelsToDialogsX(Pixs:Word):Word
3181: Func PixelsToDialogsY(Pixs:Word):Word
3182: {$IFDEF WIN32}
3183: Proc ShowMDIClientEdge(ClientHandle: THandle; ShowEdge:Bool);
3184: Func MakeVariant(const Values: array of Variant): Variant;
3185: {$ENDIF}
3186: Func CreateRotatedFont(Font: TFont; Angle: Int): HFONT;
3187: Func MsgBox(const Caption, Text:Str; Flags: Int): Int;
3188: Func MsgDlg(const Msg:str;AType:TMsgDlgType;AButtons:TMsgDlgButtons;HelpCtx:Longint):Word;
3189: {$IFDEF CBUILDER}
3190: Func FindPrevInstance(const MainFormClass: ShortString; const ATitle:Str): HWND;
3191: Func ActivatePrevInstance(const MainFormClass: ShortString; const ATitle:Str):Boolean;
3192: {$ELSE}
3193: Func FindPrevInstance(const MainFormClass, ATitle:Str): HWND;
3194: Func ActivatePrevInstance(const MainFormClass, ATitle:Str):Bool;
3195: {$ENDIF CBUILDER}
3196: Func IsForegroundTask:Bool;
3197: Proc MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show:Bool);
3198: Func GetAveCharSize(Canvas: TCanvas): TPoint;
3199: Func MinimizeText(const Text:Str; Canvas: TCanvas; MaxWidth: Int):Str;
3200: Proc FreeUnusedOle;
3201: Proc Beep;
3202: Func GetWindowsVersionJ:Str;
3203: Func LoadDLL(const LibName:Str): THandle;
3204: Func RegisterServer(const ModuleName:Str):Bool;
3205: {$IFNDEF WIN32}
3206: Func IsLibrary:Bool;
3207: {$ENDIF}
3208: { Gradient filling routine }
3209: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3210: Proc GradientFillRect(Canvas:TCanvas; ARect:TRect;StartColor,
      EndColor:TColor;Direction:TFillDirection;Colors: Byte);
3211: { String routines }
3212: Func GetEnvVar(const VarName:Str):Str;
3213: Func AnsiUpperFirstChar(const S:Str):Str;
3214: Func StringToPChar(var S:Str): PChar;
3215: Func StrPAlloc(const S:Str): PChar;
3216: Proc SplitCommandLine(const CmdLine:Str; var ExeName,Params:str);
3217: Func DropT(const S:Str):Str;
3218: { Memory routines }
3219: Func AllocMemo(Size: Longint): Pointer;
3220: Func ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3221: Proc FreeMemo(var fpBlock: Pointer);
3222: Func GetMemoSize(fpBlock: Pointer): Longint;
3223: Func CompareMem(fpBlock1, fpBlock2: Pointer; Size:Card):Bool;
3224: {$IFNDEF COMPILER5_UP}
3225: Proc FreeAndNil(var Obj);
3226: {$ENDIF}
3227: // from PNGLoader
3228: Func OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Int;TransparentColor:TColor):Int
3229: Proc TransformRGB2LOCO( Image : TLinearBitmap)
3230: Proc TransformLOCO2RGB( Image : TLinearBitmap)
3231: Proc SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3232: Func DrawButtonFace(Canvas:TCanvas;const Client:TRect;BevelWidth:Int;Style:TButtonStyle; IsRounded,IsDown,
      IsFocused:Boolean):TRect  //TButtons
3233:  Func IsAnAllResult( const AModalResult : TModalResult) :Bool
3234:  Func InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam: Longint):Longint
3235:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3236:  //Proc ChangeBiDiModeAlignment( var Alignment : TAlignment)
```

```
3237:  //Func SendAppMessage( Msg :Card; WParam, LParam : Longint) : Longint
3238:  //Proc MoveWindowOrg( DC : HDC; DX, DY : Int)
3239:  Proc SetImeMode( hWnd : HWND; Mode : TImeMode)
3240:  Proc SetImeName( Name : TImeName)
3241:  Func Win32NLSEnableIME( hWnd : HWND; Enable :Bool) :Bool
3242:  Func Imm32GetContext( hWnd : HWND) : HIMC
3243:  Func Imm32ReleaseContext( hWnd: HWND; hImc : HIMC) :Bool
3244:  Func Imm32GetConversionStatus(hImc : HIMC; var Conversion, Sentence : longword):Bool
3245:  Func Imm32SetConversionStatus(hImc : HIMC; Conversion, Sentence : longword):Bool
3246:  Func Imm32SetOpenStatus( hImc : HIMC; fOpen :Bool) :Bool
3247: // Func Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM):Bool
3248:  //Func Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) :Bool
3249:  Func Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:str;dwBufLen:longint):Longint
3250:  Func Imm32IsIME( hKl : longword) :Bool
3251:  Func Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3252:  Proc DragDone( Drop :Bool)
3253:
3254:
3255: //****************************************add  from jvjvclutils
3256: Func CanvasMaxTextHeight(Canvas: TCanvas): Int;
3257: Func ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Bool;
3258: Proc DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Int);
3259: Func IsPositiveResult(Value: TModalResult):Bool;
3260: Func IsNegativeResult(Value: TModalResult):Bool;
3261: Func IsAbortResult(const Value: TModalResult):Bool;
3262: Func StripAllFromResult(const Value: TModalResult): TModalResult;
3263: // returns either BrightColor or DarkColor depending on the luminance of AColor
3264: // This Func gives the same result (AFAIK) as the Func used in Windows to
3265: // calculate the desktop icon text color based on the desktop background color
3266: Func SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3267: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3268: Proc HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3269:   const State: TOwnerDrawState; const Text:Str; var Width: Int;
3270:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Int; var MouseOnLink:Bool;
3271:   var LinkName:Str; Scale: Int = 100); overload;
3272: Proc HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3273:   const State: TOwnerDrawState; const Text:Str; var Width, Height: Int;
3274:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Int; var MouseOnLink:Bool;
3275:   var LinkName:Str; Scale: Int = 100); overload;
3276: Func HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3277:   const State: TOwnerDrawState; const Text:Str; Scale: Int = 100):Str;
3278: Func HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3279:   const State: TOwnerDrawState; const Text:Str; MouseX, MouseY: Int;
3280:   Scale: Int = 100):Str;
3281: Func HTMLPlainText(const Text:Str):Str;
3282: Func HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3283:   const State: TOwnerDrawState; const Text:Str; Scale: Int = 100): TSize;
3284: Func HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3285:   const State: TOwnerDrawState; const Text:Str; Scale: Int = 100): Int;
3286: Func HTMLTextHeight(Canvas: TCanvas; const Text:str;Scale:Int = 100):Int;
3287: Func HTMLPrepareText(const Text:Str):Str;
3288:
3289: ****************************************** uPSI_JvAppUtils;
3290: Func GetDefaultSection( Component : TComponent) :Str
3291: Proc GetDefaultIniData(Control:TControl;var IniFileName,Section:str;UseRegistry:Bool)
3292: Proc GetDefaultIniData( Control : TControl; var IniFileName, Section :Str)
3293: Func GetDefaultIniName :Str
3294:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3295: Func GetDefaultIniRegKey :Str
3296: Func FindForm( FormClass : TFormClass) : TForm
3297: Func FindShowForm( FormClass : TFormClass; const Caption :Str) : TForm
3298: Func ShowDialog( FormClass : TFormClass) :Bool
3299:  //Func InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3300: Proc SaveFormPlacement( Form : TForm; const IniFileName :Str; UseRegistry :Bool)
3301: Proc RestoreFormPlacement(Form : TForm; const IniFileName :Str; UseRegistry :Bool)
3302: Proc SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3303: Proc SaveFormPlacement( Form : TForm; const IniFileName :Str)
3304: Proc RestoreFormPlacement( Form : TForm; const IniFileName :Str)
3305: Func GetUniqueFileNameInDir( const Path, FileNameMask :Str) :Str
3306: Func StrToIniStr( const Str :Str) :Str
3307: Func IniStrToStr( const Str :Str) :Str
3308: Func IniReadString( IniFile : TObject; const Section, Ident, Default:Str):Str
3309: Proc IniWriteString( IniFile : TObject; const Section, Ident, Value:Str)
3310: Func IniReadInt(IniFile:TObject;const Section, Ident :Str; Default:Longint): Longint
3311: Proc IniWriteInt( IniFile : TObject; const Section, Ident :Str; Value : Longint)
3312: Func IniReadBool(IniFile:TObject; const Section,Ident:str; Default:Boolean):Bool
3313: Proc IniWriteBool( IniFile : TObject; const Section, Ident :Str; Value :Bool)
3314: Proc IniReadSections( IniFile : TObject; Strings : TStrings)
3315: Proc IniEraseSection( IniFile : TObject; const Section :Str)
3316: Proc IniDeleteKey( IniFile : TObject; const Section, Ident :Str)
3317: Proc AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3318: Proc AppBroadcast( Msg, wParam : Word; lParam : Longint)
3319: Proc AppTaskbarIcons( AppOnly :Bool)
3320: Proc InternalSaveGridLayout(Grid : TCustomGrid; IniFile : TObject; const Section:str)
3321: Proc InternalRestoreGridLayout(Grid:TCustomGrid;IniFile : TObject; const Section:str)
3322: Proc InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3323: Proc InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3324: ****************************************** uPSI_JvDBUtils;
3325: Func CreateLocate( DataSet : TDataSet) : TJvLocateObject
```

```
3326: Func IsDataSetEmpty( DataSet : TDataSet) :Bool
3327: Proc RefreshQuery( Query : TDataSet)
3328: Func DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:str;CaseInsensitive:Bool):Bool
3329: Func DataSetSectionName( DataSet : TDataSet) :Str
3330: Proc InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section :Str)
3331: Proc InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:str;RestoreVisible:Bool)
3332: Func DataSetLocateThrough(DataSet:TDataSet;const KeyFields:str;const
      KeyValues:Variant;Options:TLocateOptions):Bool
3333: Proc SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3334: Proc RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible :Bool)
3335: Proc AssignRecord( Source, Dest : TDataSet; ByName :Bool)
3336: Func ConfirmDelete :Bool
3337: Proc ConfirmDataSetCancel( DataSet : TDataSet)
3338: Proc CheckRequiredField( Field : TField)
3339: Proc CheckRequiredFields( const Fields : array of TField)
3340: Func DateToSQL( Value : TDateTime) :Str
3341: Func FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName :Str) :Str
3342: Func FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName :Str) :Str
3343: Func FormatSQLNumericRange(const FieldName:str;LowVal,HighVal,LowEmpty,HighEmpty:Double;Inclusive:Bool):str
3344: Func StrMaskSQL( const Value :Str) :Str
3345: Func FormatSQLCondition(const FieldName,Operator,Val:str;FieldType:TFieldType;Exact:Bool):str
3346: Func FormatAnsiSQLCondition(const FieldName,Operator,Val:str;FieldType:TFieldType;Exact:Bool):str
3347: Proc _DBError( const Msg :Str)
3348:  Const('TrueExpr','String '0=0
3349:  Const('sdfStandard16','String '''"''mm''/''dd''/''yyyy''"''
3350:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3351:  Const('sdfOracle','String '"TO_DATE('''dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3352:  Const('sdfInterbase','String '"CAST('''mm"/"dd"/"yyyy"'' AS DATE)"
3353:  Const('sdfMSSQL','String '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3354:  AddTypeS('Largeint', 'Longint
3355:  TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3356:    erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3357:   erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3358:   erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3359: 'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederror);
3360: (*----------------------------------------------------------------------------*)
3361: Proc SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3362: begin
3363:  Func JIniReadBool( const FileName, Section, Line :Str) :Bool
3364:  Func JIniReadInt( const FileName, Section, Line :Str) : Int
3365:  Func JIniReadString( const FileName, Section, Line :Str) :Str
3366:  Proc JIniWriteBool( const FileName, Section, Line :Str; Value :Bool)
3367:  Proc JIniWriteInt( const FileName, Section, Line :Str; Value : Int)
3368:  Proc JIniWriteString( const FileName, Section, Line, Value :Str)
3369:  Proc JIniReadStrings(IniFile:TCustomIniFile; const Section :Str; Strings TStrings)
3370:  Proc JIniWriteStrings(IniFile: TCustomIniFile; const Section:str; Strings:TStrings)
3371: end;
3372:
3373: (* === compile-time registration functions === *)
3374: (*----------------------------------------------------------------------------*)
3375: Proc SIRegister_JclDateTime(CL: TPSPascalCompiler);
3376: begin
3377:  'UnixTimeStart','LongInt'( 25569);
3378:  Func JEncodeDate( const Year : Int; Month, Day : Word) : TDateTime
3379:  Proc JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3380:  Proc DecodeDate1( Date : TDateTime; var Year : Int; var Month, Day : Word);
3381:  Proc DecodeDate2( Date : TDateTime; var Year, Month, Day : Int);
3382:  Func CenturyOfDate( const DateTime : TDateTime) : Int
3383:  Func CenturyBaseYear( const DateTime : TDateTime) : Int
3384:  Func DayOfDate( const DateTime : TDateTime) : Int
3385:  Func MonthOfDate( const DateTime : TDateTime) : Int
3386:  Func YearOfDate( const DateTime : TDateTime) : Int
3387:  Func JDayOfTheYear( const DateTime : TDateTime; var Year : Int) : Int;
3388:  Func DayOfTheYear1( const DateTime : TDateTime) : Int;
3389:  Func DayOfTheYearToDateTime( const Year, Day : Int) : TDateTime
3390:  Func HourOfTime( const DateTime : TDateTime) : Int
3391:  Func MinuteOfTime( const DateTime : TDateTime) : Int
3392:  Func SecondOfTime( const DateTime : TDateTime) : Int
3393:  Func GetISOYearNumberOfDays( const Year : Word) : Word
3394:  Func IsISOLongYear( const Year : Word) :Bool;
3395:  Func IsISOLongYear1( const DateTime : TDateTime) :Bool;
3396:  Func ISODayOfWeek( const DateTime : TDateTime) : Word
3397:  Func JISOWeekNumber(DateTime:TDateTime; var YearOfWeekNumber,WeekDay: Int) : Int;
3398:  Func ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Int) : Int;
3399:  Func ISOWeekNumber2( DateTime : TDateTime) : Int;
3400:  Func ISOWeekToDateTime( const Year, Week, Day : Int) : TDateTime
3401:  Func JIsLeapYear( const Year : Int) :Bool;
3402:  Func IsLeapYear1( const DateTime : TDateTime) :Bool;
3403:  Func JDaysInMonth( const DateTime : TDateTime) : Int
3404:  Func Make4DigitYear( Year, Pivot : Int) : Int
3405:  Func JMakeYear4Digit( Year, WindowsillYear : Int) : Int
3406:  Func JEasterSunday( const Year : Int) : TDateTime // TDosDateTime', 'Int
3407:  Func JFormatDateTime( Form :Str; DateTime : TDateTime) :Str
3408:  Func FATDatesEqual( const FileTime1, FileTime2 : Int64) :Bool;
3409:  Func FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) :Bool;
3410:  Func HoursToMSecs( Hours : Int) : Int
3411:  Func MinutesToMSecs( Minutes : Int) : Int
3412:  Func SecondsToMSecs( Seconds : Int) : Int
3413:  Func TimeOfDateTimeToSeconds( DateTime : TDateTime) : Int
```

```
3414:  Func TimeOfDateTimeToMSecs( DateTime : TDateTime) : Int
3415:  Func DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3416:  Func LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3417:  Func DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3418:  Func JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3419:  Func JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3420:  Proc DateTimeToSystemTime1( DateTime : TDateTime; var SysTime:TSystemTime);
3421:  Func LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3422:  Func DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3423:  Func JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3424:  Proc DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3425:  Func DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3426:  Func DosDateTimeToStr( DateTime : Int) :Str
3427:  Func JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3428:  Func FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3429:  Func JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3430:  Proc FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time: Word);
3431:  Func JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3432:  Proc FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3433:  Func FileTimeToStr( const FileTime : TFileTime) :Str
3434:  Func SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3435:  Func JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3436:  Proc SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3437:  Func SystemTimeToStr( const SystemTime : TSystemTime) :Str
3438:  Func CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3439:  Func LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3440:  Func LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3441:   TJclUnixTime32', 'Longword
3442:  Func JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3443:  Func JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3444:  Func FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3445:  Func UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3446:  Func JNullStamp : TTimeStamp
3447:  Func JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp): Int64
3448:  Func JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) :Bool
3449:  Func JIsNullTimeStamp( const Stamp : TTimeStamp) :Bool
3450:  Func TimeStampDOW( const Stamp : TTimeStamp) : Int
3451:  Func FirstWeekDay( const Year, Month : Int; var DOW : Int): Int;
3452:  Func FirstWeekDay1( const Year, Month : Int) : Int;
3453:  Func LastWeekDay( const Year, Month : Int; var DOW : Int): Int;
3454:  Func LastWeekDay1( const Year, Month : Int) : Int;
3455:  Func IndexedWeekDay( const Year, Month : Int; Index : Int): Int
3456:  Func FirstWeekendDay( const Year, Month : Int; var DOW : Int): Int;
3457:  Func FirstWeekendDay1( const Year, Month : Int) : Int;
3458:  Func LastWeekendDay( const Year, Month : Int; var DOW : Int): Int;
3459:  Func LastWeekendDay1( const Year, Month : Int) : Int;
3460:  Func IndexedWeekendDay( const Year, Month : Int; Index : Int): Int
3461:  Func FirstDayOfWeek( const Year, Month, DayOfWeek : Int) : Int
3462:  Func LastDayOfWeek( const Year, Month, DayOfWeek : Int) : Int
3463:  Func IndexedDayOfWeek( const Year, Month, DayOfWeek, Index:Int):Int
3464:   FindClass('TOBJECT'),'EJclDateTimeError
3465: end;
3466:
3467: Proc SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3468: begin
3469:  Func SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3470:  Func CreateDOSProcessRedirected(const CommandLine, InputFile,OutputFile:str):Bool
3471:  Func CreateDOSProcessRedirected2(const CommandLine, InputFile,OutputFile:str):Bool
3472:  Func CreateDOSProcessRedirected3(const CommandLine, InputFile, OutputFile, ErrMsg:Str):Bool;
3473:  Func WinExec32( const Cmd :Str; const CmdShow : Int) :Bool
3474:  Func WinExec32AndWait( const Cmd :Str; const CmdShow : Int) :Card
3475:  Func WinExec32AndRedirectOutput(const Cmd:str;var Output:str;RawOutput:Bool):Card
3476:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3477:  Func ExitWindows( ExitCode :Card) :Bool
3478:  Func LogOffOS( KillLevel : TJclKillLevel) :Bool
3479:  Func PowerOffOS( KillLevel : TJclKillLevel) :Bool
3480:  Func ShutDownOS( KillLevel : TJclKillLevel) :Bool
3481:  Func RebootOS( KillLevel : TJclKillLevel) :Bool
3482:  Func HibernateOS( Force, DisableWakeEvents :Bool) :Bool
3483:  Func SuspendOS( Force, DisableWakeEvents :Bool) :Bool
3484:  Func ShutDownDialog(const DialogMessage:str;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3485:  Func ShutDownDialog1(const MachineName,DialogMessage:str;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3486:  Func AbortShutDown :Bool;
3487:  Func AbortShutDown1( const MachineName :Str) :Bool;
3488:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3489:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3490:  Func GetAllowedPowerOperations : TJclAllowedPowerOperations
3491:   FindClass('TOBJECT'),'EJclCreateProcessError
3492:  Proc CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine :Str)
3493:  Proc CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:str;const Environment:PChar);
3494:   // with Add(EJclCreateProcessError) do
3495: end;
3496:
3497: Proc SIRegister_JclAnsistrs(CL: TPSPascalCompiler);
3498: begin
3499:   //'AnsiSigns','Set').SetSet(['-', '+']);
3500:  'C1_UPPER','LongWord( $0001);
3501:  'C1_LOWER','LongWord( $0002);
3502:  'C1_DIGIT','LongWord').SetUInt( $0004);
```

```
'C1_SPACE','LongWord').SetUInt( $0008);
'C1_PUNCT','LongWord').SetUInt( $0010);
'C1_CNTRL','LongWord').SetUInt( $0020);
'C1_BLANK','LongWord').SetUInt( $0040);
'C1_XDIGIT','LongWord').SetUInt( $0080);
'C1_ALPHA','LongWord').SetUInt( $0100);
 AnsiChar', 'Char
Func StrIsAlpha( const S : Ansistr) :Bool
Func StrIsAlphaNum( const S : Ansistr) :Bool
Func StrIsAlphaNumUnderscore( const S : Ansistr) :Bool
Func StrContainsChars(const S:Ansistr;Chars:TSysCharSet; CheckAll:Boolean):Bool
Func StrConsistsOfNumberChars( const S : Ansistr) :Bool
Func StrIsDigit( const S : Ansistr) :Bool
Func StrIsSubset( const S : Ansistr; const ValidChars : TSysCharSet): Bool
Func StrSame( const S1, S2 : Ansistr) :Bool
//Func StrCenter( const S : Ansistr; L : Int; C : AnsiChar) : Ansistr
Func StrCharPosLower( const S : Ansistr; CharPos : Int) : Ansistr
Func StrCharPosUpper( const S : Ansistr; CharPos : Int) : Ansistr
Func StrDoubleQuote( const S : Ansistr) : Ansistr
Func StrEnsureNoPrefix( const Prefix, Text : Ansistr) : Ansistr
Func StrEnsureNoSuffix( const Suffix, Text : Ansistr) : Ansistr
Func StrEnsurePrefix( const Prefix, Text : Ansistr) : Ansistr
Func StrEnsureSuffix( const Suffix, Text : Ansistr) : Ansistr
Func StrEscapedToString( const S : Ansistr) : Ansistr
Func JStrLower( const S : Ansistr) : Ansistr
Proc StrLowerInPlace( var S : Ansistr)
///Proc StrLowerBuff( S : PAnsiChar)
Proc JStrMove( var Dest:AnsiStr;const Source:AnsiStr;const ToIndex,FromIndex,Count:Int;
Func StrPadLeft( const S : Ansistr; Len : Int; C : AnsiChar) : Ansistr
Func StrPadRight( const S : Ansistr; Len : Int; C : AnsiChar) : Ansistr
Func StrProper( const S : Ansistr) : Ansistr
//Proc StrProperBuff( S : PAnsiChar)
Func StrQuote( const S : Ansistr; C : AnsiChar) : Ansistr
Func StrRemoveChars( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
Func StrKeepChars( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
Proc JStrReplace(var S:Ansistr;const Search,Replace:Ansistr;Flags: TReplaceFlags)
Func StrReplaceChar( const S : Ansistr; const Source, Replace : AnsiChar) : Ansistr
Func StrReplaceChars(const S:AnsiStr;const Chars:TSysCharSet;Replace:AnsiChar):Ansistr
Func StrReplaceButChars(const S:AnsiStr;const Chars:TSysCharSet;Replace:AnsiChar):AnsiStr;
Func StrRepeat( const S : Ansistr; Count : Int) : Ansistr
Func StrRepeatLength( const S : Ansistr; const L : Int) : Ansistr
Func StrReverse( const S : Ansistr) : Ansistr
Proc StrReverseInPlace( var S : Ansistr)
Func StrSingleQuote( const S : Ansistr) : Ansistr
Func StrSmartCase( const S : Ansistr; Delimiters : TSysCharSet) : Ansistr
Func StrStringToEscaped( const S : Ansistr) : Ansistr
Func StrStripNonNumberChars( const S : Ansistr) : Ansistr
Func StrToHex( const Source : Ansistr) : Ansistr
Func StrTrimCharLeft( const S : Ansistr; C : AnsiChar) : Ansistr
Func StrTrimCharsLeft( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
Func StrTrimCharRight( const S : Ansistr; C : AnsiChar) : Ansistr
Func StrTrimCharsRight( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
Func StrTrimQuotes( const S : Ansistr) : Ansistr
Func JStrUpper( const S : Ansistr) : Ansistr
Proc StrUpperInPlace( var S : Ansistr)
//Proc StrUpperBuff( S : PAnsiChar)
Func StrOemToAnsi( const S : Ansistr) : Ansistr
Func StrAnsiToOem( const S : Ansistr) : Ansistr
Proc StrAddRef( var S : Ansistr)
Func StrAllocSize( const S : Ansistr) : Longint
Proc StrDecRef( var S : Ansistr)
//Func StrLen( S : PAnsiChar) : Int
Func StrLength( const S : Ansistr) : Longint
Func StrRefCount( const S : Ansistr) : Longint
Proc StrResetLength( var S : Ansistr)
Func StrCharCount( const S : Ansistr; C : AnsiChar) : Int
Func StrCharsCount( const S : Ansistr; Chars : TSysCharSet) : Int
Func StrStrCount( const S, SubS : Ansistr) : Int
Func StrCompare( const S1, S2 : Ansistr) : Int
Func StrCompareRange( const S1, S2 : Ansistr; const Index, Count:Int) : Int
//Func StrFillChar( const C : AnsiChar; Count : Int) : Ansistr;
Func StrFillChar1( const C : Char; Count : Int) : Ansistr;
Func StrFillChar(const C: Char; Count: Int):Str);
Func IntFillChar(const I: Int; Count: Int):Str);
Func ByteFillChar(const B: Byte; Count: Int):Str);
Func ArrFillChar(const AC: Char; Count: Int): TCharArray;;
Func ArrByteFillChar(const AB: Char; Count: Int): TByteArray;
Func StrFind( const Substr, S : Ansistr; const Index : Int) : Int
//Func StrHasPrefix( const S : Ansistr; const Prefixes : array of Ansistr) :Bool
Func StrIndex( const S : Ansistr; const List : array of Ansistr) : Int
Func StrILastPos( const SubStr, S : Ansistr) : Int
Func StrIPos( const SubStr, S : Ansistr) : Int
Func StrIsOneOf( const S : Ansistr; const List : array of Ansistr) :Bool
Func StrLastPos( const SubStr, S : Ansistr) : Int
Func StrMatch( const Substr, S : Ansistr; const Index : Int) : Int
Func StrMatches( const Substr, S : Ansistr; const Index : Int) :Bool
Func StrNIPos( const S, SubStr : Ansistr; N : Int) : Int
Func StrNPos( const S, SubStr : Ansistr; N : Int) : Int
Func StrPrefixIndex( const S : Ansistr; const Prefixes : array of Ansistr) : Int
```

```
3592:  Func StrSearch( const Substr, S : Ansistr; const Index : Int) : Int
3593:  //Func StrAfter( const SubStr, S : Ansistr) : Ansistr
3594:  //Func StrBefore( const SubStr, S : Ansistr) : Ansistr
3595:  Func StrBetween( const S : Ansistr; const Start, Stop : AnsiChar) : Ansistr
3596:  Func StrChopRight( const S : Ansistr; N : Int) : Ansistr
3597:  Func StrLeft( const S : Ansistr; Count : Int) : Ansistr
3598:  Func StrMid( const S : Ansistr; Start, Count : Int) : Ansistr
3599:  Func StrRestOf( const S : Ansistr; N : Int) : Ansistr
3600:  Func StrRight( const S : Ansistr; Count : Int) : Ansistr
3601:  Func CharEqualNoCase( const C1, C2 : AnsiChar) :Bool
3602:  Func CharIsAlpha( const C : AnsiChar) :Bool
3603:  Func CharIsAlphaNum( const C : AnsiChar) :Bool
3604:  Func CharIsBlank( const C : AnsiChar) :Bool
3605:  Func CharIsControl( const C : AnsiChar) :Bool
3606:  Func CharIsDelete( const C : AnsiChar) :Bool
3607:  Func CharIsDigit( const C : AnsiChar) :Bool
3608:  Func CharIsLower( const C : AnsiChar) :Bool
3609:  Func CharIsNumberChar( const C : AnsiChar) :Bool
3610:  Func CharIsPrintable( const C : AnsiChar) :Bool
3611:  Func CharIsPunctuation( const C : AnsiChar) :Bool
3612:  Func CharIsReturn( const C : AnsiChar) :Bool
3613:  Func CharIsSpace( const C : AnsiChar) :Bool
3614:  Func CharIsUpper( const C : AnsiChar) :Bool
3615:  Func CharIsWhiteSpace( const C : AnsiChar) :Bool
3616:  Func CharType( const C : AnsiChar) : Word
3617:  Func CharHex( const C : AnsiChar) : Byte
3618:  Func CharLower( const C : AnsiChar) : AnsiChar
3619:  Func CharUpper( const C : AnsiChar) : AnsiChar
3620:  Func CharToggleCase( const C : AnsiChar) : AnsiChar
3621:  Func CharPos( const S : Ansistr; const C : AnsiChar; const Index : Int) : Int
3622:  Func CharLastPos( const S : Ansistr; const C : AnsiChar; const Index : Int) : Int
3623:  Func CharIPos( const S : Ansistr; C : AnsiChar; const Index : Int) : Int
3624:  Func CharReplace( var S : Ansistr; const Search, Replace : AnsiChar) : Int
3625:  Proc StrIToStrings(S,Sep:Ansistr; const List:TStrings;const AllowEmptyString :Bool)
3626:  Proc StrToStrings(S,Sep: Ansistr; const List:TStrings;const AllowEmptyString :Bool)
3627:  Func StringsToStr(const List:TStrings;const Sep:AnsiStr;const AllowEmptyString:Bool):AnsiStr;
3628:  Proc TrimStrings( const List : TStrings; DeleteIfEmpty :Bool)
3629:  Proc TrimStringsRight( const List : TStrings; DeleteIfEmpty :Bool)
3630:  Proc TrimStringsLeft( const List : TStrings; DeleteIfEmpty :Bool)
3631:  Func AddStringToStrings(const S:Ansistr;Strings:TStrings; const Unique:Boolean):Bool
3632:  Func BooleanToStr( B :Bool) : Ansistr
3633:  Func FileToString( const FileName : Ansistr) : Ansistr
3634:  Proc StringToFile( const FileName, Contents: Ansistr; Append:Bool) //binary save!
3635:  Proc StringToFile2( const S, FileName :Str);
3636:  Func StrToken( var S : Ansistr; Separator : AnsiChar) : Ansistr
3637:  Proc StrTokens( const S : Ansistr; const List : TStrings)
3638:  Proc StrTokenToStrings( S : Ansistr; Separator : AnsiChar; const List : TStrings)
3639:  //Func StrWord( var S : PAnsiChar; out Word : Ansistr) :Bool
3640:  Func StrToFloatSafe( const S : Ansistr) : Float
3641:  Func StrToIntSafe( const S : Ansistr) : Int
3642:  Proc StrNormIndex( const StrLen : Int; var Index : Int; var Count : Int);
3643:  Func ArrayOf( List : TStrings) : TDynStringArray;
3644:   EJclStringError', 'EJclError
3645:  Func IsClass(Address: TObject):Bool;
3646:  Func IsObject(Address: TObject):Bool;
3647:  // Console Utilities
3648:  //Func ReadKey: Char;
3649:  Func IntToStrZeroPad(Value, Count: Int): Ansistr;
3650:  Func JclGUIDToString(const GUID: TGUID):Str;
3651:  Func JclStringToGUID(const S:Str): TGUID;
3652: end;
3653:
3654: ********************************************** uPSI_JvDBUtil;
3655: Proc ExecuteSQLScript(Base:TDataBase;const Script:str; const Commit:TCommit;OnProgress:TOnProgress; const
      UserData : Int)
3656: Func GetQueryResult( const DatabaseName, SQL :Str) : Variant
3657: Func GetStoredProcResult(const ADatabaseName,AStoredProcName:str; AParams: array of Variant;const
      AResultName:str):Variant
3658: //Func StrFieldDesc( Field : FLDDesc) :Str
3659: Func Var2Type( V : Variant; const VarType : Int) : Variant
3660: Proc CopyRecord( DataSet : TDataSet)
3661: //Proc AddReference( Tbl : TTable; RefName:Str; RefField: Word; MasterTable :Str; MasterField : Word;
      ModOp, DelOp : RINTQual)
3662: Proc AddMasterPassword( Table : TTable; pswd :Str)
3663: Proc PackTable( Table : TTable)
3664: Proc PackEncryptedTable( Table : TTable; pswd :Str)
3665: Func EncodeQuotes( const S :Str) :Str
3666: Func Cmp( const S1, S2 :Str) :Bool
3667: Func SubStr( const S :Str; const Index : Int; const Separator :Str) :Str
3668: Func SubStrEnd( const S :Str; const Index : Int; const Separator :Str):Str
3669: Func ReplaceString( S :Str; const OldPattern, NewPattern :Str) :Str
3670: Proc GetXYByPos( const S :Str; const Pos : Int; var X, Y : Int)
3671: ********************************************uPSI_JvJvBDEUtils;***************
3672: //JvBDEUtils
3673: Func CreateDbLocate : TJvLocateObject
3674:  //Func CheckOpen( Status : DBIResult) :Bool
3675: Proc FetchAllRecords( DataSet : TBDEDataSet)
3676: Func TransActive( Database : TDatabase) :Bool
3677: Func AsyncQrySupported( Database : TDatabase) :Bool
```

```
3678: Func GetQuoteChar( Database : TDatabase) :Str
3679: Proc ExecuteQuery( const DbName, QueryText :Str)
3680: Proc ExecuteQueryEx( const SessName, DbName, QueryText :Str)
3681: Func FieldLogicMap( FldType : TFieldType) : Int
3682: Func FieldSubtypeMap( FldType : TFieldType) : Int Value :Str; Buffer : Pointer)
3683: Func GetAliasPath( const AliasName :Str) :Str
3684: Func IsDirectory( const DatabaseName :Str) :Bool
3685: Func GetBdeDirectory :Str
3686: Func LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) :Bool
3687: Func DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName :Str) :Bool
3688: Func DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName :Str) :Bool
3689: Func DataSetRecNo( DataSet : TDataSet) : Longint
3690: Func DataSetRecordCount( DataSet : TDataSet) : Longint
3691: Func DataSetPositionStr( DataSet : TDataSet) :Str
3692: Proc DataSetShowDeleted( DataSet : TBDEDataSet; Show :Bool)
3693: Func CurrentRecordDeleted( DataSet : TBDEDataSet) :Bool
3694: Func IsFilterApplicable( DataSet : TDataSet) :Bool
3695: Func IsBookmarkStable( DataSet : TBDEDataSet) :Bool
3696: Proc SetIndex( Table : TTable; const IndexFieldNames :Str)
3697: Proc RestoreIndex( Table : TTable)
3698: Proc DeleteRange(Table: TTable; IndexFields:array of const; FieldValues: array of const)
3699: Proc PackTable( Table : TTable)
3700: Proc ReindexTable( Table : TTable)
3701: Proc BdeFlushBuffers
3702: Func GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Int) : Pointer
3703: Proc ToggleDebugLayer( Active :Bool; const DebugFile :Str)
3704: Proc DbNotSupported
3705: Proc ExportDataSet(Source:TBDEDataSet;DestTable:TTable;TableType:TTableType;const AsciiCharSe
      :str;AsciiDelimited Boolean;MaxRecordCount:Longint)
3706: Proc ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:str;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3707: Proc
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3708: Proc InitRSRUN(Database:TDatabase;const ConName:str;ConType:Int;const ConServer:str);
3709: ************************************************uPSI_JvDateUtil;
3710: Func CurrentYear: Word;
3711: Func IsLeapYear(AYear: Int):Bool;
3712: Func DaysPerMonth(AYear, AMonth: Int): Int;
3713: Func FirstDayOfPrevMonth: TDateTime;
3714: Func LastDayOfPrevMonth: TDateTime;
3715: Func FirstDayOfNextMonth: TDateTime;
3716: Func ExtractDay(ADate: TDateTime): Word;
3717: Func ExtractMonth(ADate: TDateTime): Word;
3718: Func ExtractYear(ADate: TDateTime): Word;
3719: Func IncDate(ADate: TDateTime; Days, Months, Years: Int): TDateTime;
3720: Func IncDay(ADate: TDateTime; Delta: Int): TDateTime;
3721: Func IncMonth(ADate: TDateTime; Delta: Int): TDateTime;
3722: Func IncYear(ADate: TDateTime; Delta: Int): TDateTime;
3723: Func ValidDate(ADate: TDateTime):Bool;
3724: Proc DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3725: Func MonthsBetween(Date1, Date2: TDateTime): Double;
3726: Func DaysInPeriod(Date1, Date2: TDateTime): Longint;
3727: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3728: Func DaysBetween(Date1, Date2: TDateTime): Longint;
3729: { The same as previous but if Date2 < Date1 result = 0 }
3730: Func IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Int): TDateTime;
3731: Func IncHour(ATime: TDateTime; Delta: Int): TDateTime;
3732: Func IncMinute(ATime: TDateTime; Delta: Int): TDateTime;
3733: Func IncSecond(ATime: TDateTime; Delta: Int): TDateTime;
3734: Func IncMSec(ATime: TDateTime; Delta: Int): TDateTime;
3735: Func CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3736: { String to date conversions }
3737: Func GetDateOrder(const DateFormat:Str): TDateOrder;
3738: Func MonthFromName(const S:Str; MaxLen: Byte): Byte;
3739: Func StrToDateDef(const S:Str; Default: TDateTime): TDateTime;
3740: Func StrToDateFmt(const DateFormat, S:Str): TDateTime;
3741: Func StrToDateFmtDef(const DateFormat, S:Str; Default: TDateTime): TDateTime;
3742: Func DefDateFormat(FourDigitYear:Bool):Str;
3743: Func DefDateMask(BlanksChar: Char; FourDigitYear:Bool):Str;
3744: ---------------------------------------------------------------------------------------
3745: ********************************** JvUtils;******************************
3746: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3747: Func GetWordOnPos(const S:Str; const P: Int):Str;
3748: {GetWordOnPosEx worklike GetWordOnPosfunction,also returns Wordposition in iBeg,iEnd variables}
3749: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
3750: { SubStr returns substring from string, S, separated with Separator string}
3751: Func SubStr(const S:Str; const Index: Int; const Separator:Str):Str;
3752: { SubStrEnd same to previous Func but Index numerated from the end of string }
3753: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
3754: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3755: Func SubWord(P: PChar; var P2: PChar):Str;
3756: { NumberByWord returns the text representation of
3757:   the number, N, in normal russian language. Was typed from Monitor magazine }
3758: Func NumberByWord(const N: Longint):Str;
3759: //  Func CurrencyByWord(Value : Currency) :Str; GetLineByPos returns Line number, there
3760:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3761: Func GetLineByPos(const S:Str; const Pos: Int): Int;
3762: { GetXYByPos is same to previous function, but returns X position in line too}
3763: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
```

```
3764: { ReplaceString searches for all substrings,OldPattern,in a string,S, replaces them with NewPattern }
3765: Func ReplaceString(S:Str; const OldPattern, NewPattern:Str):Str;
3766: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3767: Func ConcatSep(const S, S2, Separator:Str):Str;
3768: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3769: Func ConcatLeftSep(const S, S2, Separator:Str):Str;
3770: { MinimizeString truncs long string,S,appends '...' symbols,if Length of S is more than MaxLen}
3771: Func MinimizeString(const S:Str; const MaxLen: Int):Str;
3772: { Next 4 Func for russian chars transliterating.
3773:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3774: Proc Dos2Win(var S:Str);
3775: Proc Win2Dos(var S:Str);
3776: Func Dos2WinRes(const S:Str):Str;
3777: Func Win2DosRes(const S:Str):Str;
3778: Func Win2Koi(const S:Str):Str;
3779: { Spaces returns string consists on N space chars }
3780: Func Spaces(const N: Int):Str;
3781: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3782: Func AddSpaces(const S:Str; const N: Int):Str;
3783: { Func LastDate for russian users only } { returns date relative to current date: '' }
3784: Func LastDate(const Dat: TDateTime):Str;
3785: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3786: Func CurrencyToStr(const Cur: currency):Str;
3787: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3788: Func Cmp(const S1, S2:Str):Bool;
3789: { StringCat add S2 string to S1 and returns this string }
3790: Func StringCat(var S1:Str; S2:Str):Str;
3791: { HasChar returns True, if Char, Ch, contains in string, S }
3792: Func HasChar(const Ch: Char; const S:Str):Bool;
3793: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
3794: Func CharInSet(const Ch: Char; const SetOfChar: TSetOfChar):Bool;
3795: Func CountOfChar(const Ch: Char; const S:Str): Int;
3796: Func DefStr(const S:Str; Default:Str):Str;
3797: {**** files routines}
3798: { GetWinDir returns Windows folder name }
3799: Func GetWinDir: TFileName;
3800: Func GetSysDir:Str;
3801: { GetTempDir returns Windows temporary folder name }
3802: Func GetTempDir:Str;
3803: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3804: Func GenTempFileName(FileName:Str):Str;
3805: { GenTempFileNameExt sameto previous function,but returning filename has given extension, FileExt }
3806: Func GenTempFileNameExt(FileName:Str; const FileExt:Str):Str;
3807: { ClearDir clears folder Dir }
3808: Func ClearDir(const Dir:Str):Bool;
3809: { DeleteDir clears and than delete folder Dir }
3810: Func DeleteDir(const Dir:Str):Bool;
3811: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3812: Func FileEquMask(FileName, Mask: TFileName):Bool;
3813: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3814:   Masks must be separated with comma (';') }
3815: Func FileEquMasks(FileName, Masks: TFileName):Bool;
3816: Proc DeleteFiles(const Folder: TFileName; const Masks:Str);
3817: { LZFileExpand expand file,FileSource into FileDest.File must be compressed,using MS Compress program }
3818: Func LZFileExpand(const FileSource, FileDest:Str):Bool;
3819: { FileGetInfo fills SearchRec record for specified file attributes}
3820: Func FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec):Bool;
3821: { HasSubFolder returns True, if folder APath contains other folders }
3822: Func HasSubFolder(APath: TFileName):Bool;
3823: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3824: Func IsEmptyFolder(APath: TFileName):Bool;
3825: { AddSlash add slash Char to Dir parameter, if needed }
3826: Proc AddSlash(var Dir: TFileName);
3827: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3828: Func AddSlash2(const Dir: TFileName):Str;
3829: { AddPath returns FileName with Path, if FileName not contain any path }
3830: Func AddPath(const FileName, Path: TFileName): TFileName;
3831: Func AddPaths(const PathList, Path:Str):Str;
3832: Func ParentPath(const Path: TFileName): TFileName;
3833: Func FindInPath(const FileName, PathList:Str): TFileName;
3834: Func FindInPaths(const fileName,paths:Str):Str;
3835: {$IFNDEF BCB1}
3836: { BrowseForFolder displays Browse For Folder dialog }
3837: Func BrowseForFolder(const Handle: HWND; const Title:Str; var Folder:Str):Boolean;
3838: {$ENDIF BCB1}
3839: Func BrowseForFolder(const ATitle:Str; AllowCreate :Bool; var ADirectory :str; AHelpContext :
      THelpContext) :Bool
3840: Func BrowseForComputer(const ATitle :Str; AllowCreate :Bool; var ADirectory:str; AHelpContext :
      THelpContext): Bool
3841: Func BrowseDirectory(var AFolderName:str;const DlgText:str;AHelpContext:THelpContext):Bool
3842: Func BrowseComputer(var AComputerName:str;const DlgText:str;AHelpContext:THelpContext):Bool
3843: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3844: Func DeleteReadOnlyFile(const FileName: TFileName):Bool;
3845: { HasParam returns True, if program running with specified parameter, Param }
3846: Func HasParam(const Param:Str):Bool;
3847: Func HasSwitch(const Param:Str):Bool;
3848: Func Switch(const Param:Str):Str;
3849: { ExePath returns ExtractFilePath(ParamStr(0)) }
3850: Func ExePath: TFileName;
```

```
3851: Func CopyDir(const SourceDir, DestDir: TFileName):Bool;
3852: Func FileTimeToDateTime(const FT: TFileTime): TDateTime;
3853: Func MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3854: {**** Graphic routines }
3855: { TTFontSelected returns True, if True Type font is selected in specified device context }
3856: Func TTFontSelected(const DC: HDC):Bool;
3857: { TrueInflateRect inflates rect in other method, than InflateRect API Func }
3858: Func TrueInflateRect(const R: TRect; const I: Int): TRect;
3859: {**** Windows routines }
3860: { SetWindowTop put window to top without recreating window }
3861: Proc SetWindowTop(const Handle: HWND; const Top:Bool);
3862: {**** other routines }
3863: { KeyPressed returns True, if Key VK is now pressed }
3864: Func KeyPressed(VK: Int):Bool;
3865: Proc SwapInt(var Int1, Int2: Int);
3866: Func IntPower(Base, Exponent: Int): Int;
3867: Func ChangeTopException(E: TObject): TObject;
3868: Func StrToBool(const S:Str):Bool;
3869: {$IFNDEF COMPILER3_UP}
3870: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3871:   Length of MaxLen bytes. The compare operation is controlled by the
3872:   current Windows locale. The return value is the same as for CompareStr. }
3873: Func AnsiStrLIComp(S1, S2: PChar; MaxLen:Card): Int;
3874: Func AnsiStrIComp(S1, S2: PChar): Int;
3875: {$ENDIF}
3876: Func Var2Type(V: Variant; const VarType: Int): Variant;
3877: Func VarToInt(V: Variant): Int;
3878: Func VarToFloat(V: Variant): Double;
3879: {following functions are not documented because they are don't work properly,so don't use them}
3880: Func ReplaceSokr1(S:Str; const Word, Frase:Str):Str;
3881: { ReplaceSokr1 is full equal to ReplaceString Func - only for compatibility - don't use }
3882: { GetSubStr is full equal to SubStr Func - only for compatibility - don't use }
3883: Func GetSubStr(const S:Str; const Index: Int; const Separator: Char):Str;
3884: Func GetParameter:Str;
3885: Func GetLongFileName(FileName:Str):Str;
3886: {* from  FileCtrl}
3887: Func DirectoryExists(const Name:Str):Bool;
3888: Proc ForceDirectories(Dir:Str);
3889: {# from  FileCtrl}
3890: Func FileNewExt(const FileName, NewExt: TFileName): TFileName;
3891: Func GetComputerID:Str;
3892: Func GetComputerName:Str;
3893: {**** string routines }
3894: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 Func }
3895: Func ReplaceAllSokr(S:Str; Words, Frases: TStrings):Str;
3896: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3897:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with same
      Index, and then update NewSelStart variable }
3898: Func ReplaceSokr(S:str;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):Str;
3899: { CountOfLines calculates linescount in a string,each line separated from another with CrLf sequence }
3900: Func CountOfLines(const S:Str): Int;
3901: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3902: Proc DeleteEmptyLines(Ss: TStrings);
3903: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3904:   Note:If strings SQL allready contains where-statement,it must be started on begining of any line}
3905: Proc SQLAddWhere(SQL: TStrings; const Where:Str);
3906: {**** files routines - }
3907: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3908:   Resource can be compressed using MS Compress program}
3909: Func ResSaveToFile(const Typ,Name:str;const Compressed:Bool;const FileName:str):Bool;
3910: Func ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:str): Bool
3911: Func ResSaveToString(Instance: HINST; const Typ, Name:Str; var S:Str):Bool;
3912: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3913: Func ExecuteJ(const CommandLine, WorkingDirectory:Str): Int;
3914: { IniReadSection read section, Section, from ini-file,
3915:   IniFileName, into strings, Ss.This Func reads ALL strings from specified section.
3916:   Note: TIninFile.ReadSection Func reads only strings with '=' symbol.}
3917: Func IniReadSection(const IniFileName:TFileName;const Section:str;Ss:TStrings):Bool;
3918: { LoadTextFile load text file, FileName, into string }
3919: Func LoadTextFile(const FileName: TFileName):Str;
3920: Proc SaveTextFile(const FileName: TFileName; const Source:Str);
3921: { ReadFolder reads files list from disk folder,Folder,that are equal Mask,into strings,FileList}
3922: Func ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Int;
3923: Func ReadFolders(const Folder: TFileName; FolderList: TStrings): Int;
3924: {$IFDEF COMPILER3_UP}
3925: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3926: Func TargetFileName(const FileName: TFileName): TFileName;
3927: { return filename ShortCut linked to }
3928: Func ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3929: {$ENDIF COMPILER3_UP}
3930: {**** Graphic routines - }
3931: {LoadIcoToImage loads 2 icons from resource named NameRes,into twoimage lists ALarge and ASmall}
3932: Proc LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes:Str);
3933: { RATextOut same with TCanvas.TextOut procedure,but can clipping drawing with rectangle,RClip.}
3934: Proc RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S:Str);
3935: {RATextOutEx same with RATextOut funct, but can calculate needed height for correct output }
3936: Func RATextOutEx(Canvas:TCanvas;const R,RClip:TRect;const S:str;const CalcHeight:Bool):Int;
3937: {RATextCalcHeight calculate needed height to correct output,using RATextOut or RATextOutEx functions}
```

```
3938: Func RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S:Str): Int;
3939: { Cinema draws some visual effect }
3940: Proc Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3941: { Roughed fills rect with special 3D pattern }
3942: Proc Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert:Bool);
3943: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
      and height,AWidth,AHeight and placed on specified Index,Index in source bitmap}
3944: Func BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Int): TBitmap;
3945: { TextWidth calculate text with for writing using standard desktop font }
3946: Func TextWidth(AStr:Str): Int;
3947: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3948: Func DefineCursor(Identifer: PChar): TCursor;
3949: {**** other routines - }
3950: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3951: Func FindFormByClass(FormClass: TFormClass): TForm;
3952: Func FindFormByClassName(FormClassName:Str): TForm;
3953: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3954:   having Tag property value, equaled to Tag parameter }
3955: Func FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Int):TComponent;
3956: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3957: Func ControlAtPos2(Parent: TWinControl; X, Y: Int): TControl;
3958: { RBTag searches WinControl.Controls for checked RadioButton and returns Tag property value }
3959: Func RBTag(Parent: TWinControl): Int;
3960: { AppMinimized returns True, if Application is minimized }
3961: Func AppMinimized:Bool;
3962: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3963:   if Caption parameter = '', it replaced with Application.Title }
3964: Func MessageBoxJ(const Msg:Str; Caption:Str;const Flags: Int): Int;
3965: Func MsgDlg2(const Msg, ACaption:Str; DlgType: TMsgDlgType;
3966:   Buttons: TMsgDlgButtons; HelpContext: Int; Control: TWinControl): Int;
3967: Func MsgDlgDef(const Msg, ACaption:Str; DlgType: TMsgDlgType;
3968:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Int;Control: TWinControl):Int;
3969: { Delay stop program execution to MSec msec }
3970: Proc Delay(MSec: Longword);
3971: Proc CenterHor(Parent: TControl; MinLeft: Int; Controls: array of TControl);
3972: Proc EnableControls(Control: TWinControl; const Enable:Bool);
3973: Proc EnableMenuItems(MenuItem: TMenuItem; const Tag: Int; const Enable:Bool);
3974: Proc ExpandWidth(Parent: TControl; MinWidth: Int; Controls: array of TControl);
3975: Func PanelBorder(Panel: TCustomPanel): Int;
3976: Func Pixels(Control: TControl; APixels: Int): Int;
3977: Proc SetChildPropOrd(Owner: TComponent; PropName:Str; Value: Longint);
3978: Proc Error(const Msg:Str);
3979: Proc ItemHtDrawEx(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
3980:   const HideSelColor:Bool; var PlainItem:Str; var Width: Int; CalcWidth:Bool);
3981:   {ex.Text parameter:'Item1<b>bold</b><i>italic ITALIC<c:Red>red<c:Green>green<c:blue>blue</i>'}
3982: Func ItemHtDraw(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
3983:   const HideSelColor:Bool):Str;
3984: Func ItemHtWidth(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
3985:   const HideSelColor:Bool): Int;
3986: Func ItemHtPlain(const Text:Str):Str;
3987: { ClearList - clears list of TObject }
3988: Proc ClearList(List: TList);
3989: Proc MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3990: Proc ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3991: { RTTI support }
3992: Func GetPropType(Obj: TObject; const PropName:Str): TTypeKind;
3993: Func GetPropStr(Obj: TObject; const PropName:Str):Str;
3994: Func GetPropOrd(Obj: TObject; const PropName:Str): Int;
3995: Func GetPropMethod(Obj: TObject; const PropName:Str): TMethod;
3996: Proc PrepareIniSection(SS: TStrings);
3997: {following functions are not documented because they are don't work properly,so don't use them}
3998: {$IFDEF COMPILER2}
3999: Func CompareMem(P1, P2: Pointer; Length: Int):Bool; assembler;
4000: {$ENDIF}
4001:
4002: Proc SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
4003: begin
4004:  Proc BoxMoveSelectedItems( SrcList, DstList : TWinControl)
4005:  Proc BoxMoveAllItems( SrcList, DstList : TWinControl)
4006:  Proc BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var Accept:Bool;Sorted:Bool;
4007:  Proc BoxMoveFocusedItem( List : TWinControl; DstIndex : Int)
4008:  Proc BoxMoveSelected( List : TWinControl; Items : TStrings)
4009:  Proc BoxSetItem( List : TWinControl; Index : Int)
4010:  Func BoxGetFirstSelection( List : TWinControl) : Int
4011:  Func BoxCanDropItem( List:TWinControl; X,Y:Int; var DragIndex : Int):Boolean
4012: end;
4013:
4014: Proc SIRegister_JvCsvParse(CL: TPSPascalCompiler);
4015: begin
4016:  Const('MaxInitStrNum','LongInt'( 9);
4017: Func JvAnsiStrSplit( const InString : Ansistr; const SplitChar, QuoteChar:AnsiChar; var OutStrings : array
      of Ansistr; MaxSplit : Int) : Int
4018: Func JvStrSplit(const InString:str;const SplitChar,QuoteChar:Char;var OutStrings:array of
      string;MaxSplit:Int):Int
4019: Func JvAnsiStrSplitStrings(const InStr:Ansistr;const SplitChar,QuoteChar:AnsiChar;OutStrs:TStrings):Int;
4020: Func JvAnsiStrStrip( S : Ansistr) : Ansistr
4021: Func JvStrStrip( S :Str) :Str
4022: Func GetString( var Source : Ansistr; const Separator : Ansistr) : Ansistr
4023: Func PadString( const S : Ansistr; Len : Int; PadChar : AnsiChar) : Ansistr
```

```
4024: Func BuildPathName( const PathName, FileName : Ansistr) : Ansistr
4025: Func StrEatWhiteSpace( const S :Str) :Str
4026: Func HexToAscii( const S : Ansistr) : Ansistr
4027: Func AsciiToHex( const S : Ansistr) : Ansistr
4028: Func StripQuotes( const S1 : Ansistr) : Ansistr
4029: Func ValidNumericLiteral( S1 : PAnsiChar) :Bool
4030: Func ValidIntLiteral( S1 : PAnsiChar) :Bool
4031: Func ValidHexLiteral( S1 : PAnsiChar) :Bool
4032: Func HexPCharToInt( S1 : PAnsiChar) : Int
4033: Func ValidStringLiteral( S1 : PAnsiChar) :Bool
4034: Func StripPCharQuotes( S1 : PAnsiChar) : Ansistr
4035: Func JvValidIdentifierAnsi( S1 : PAnsiChar) :Bool
4036: Func JvValidIdentifier( S1 :Str) :Bool
4037: Func JvEndChar( X : AnsiChar) :Bool
4038: Proc JvGetToken( S1, S2 : PAnsiChar)
4039: Func IsExpressionKeyword( S1 : PAnsiChar) :Bool
4040: Func IsKeyword( S1 : PAnsiChar) :Bool
4041: Func JvValidVarReference( S1 : PAnsiChar) :Bool
4042: Func GetParenthesis( S1, S2 : PAnsiChar) :Bool
4043: Proc JvGetVarReference( S1, S2, SIdx : PAnsiChar)
4044: Proc JvEatWhitespaceChars( S1 : PAnsiChar);
4045: Proc JvEatWhitespaceChars1( S1 : PWideChar);
4046: Func GetTokenCount : Int
4047: Proc ResetTokenCount
4048: end;
4049:
4050: Proc SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
4051: begin
4052:   SIRegister_TJvQueryParamsDialog(CL);
4053:  Func EditQueryParams(DataSet:TDataSet;List:TParams;AHelpContext:THelpContext):Bool
4054: end;
4055:
4056: ********************************** JvStrUtil /  JvStrUtils;******************************
4057: Func FindNotBlankCharPos(const S:Str): Int;
4058: Func AnsiChangeCase(const S:Str):Str;
4059: Func GetWordOnPos(const S:Str; const P: Int):Str;
4060: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
4061: Func Cmp(const S1, S2:Str):Bool;
4062: { Spaces returns string consists on N space chars }
4063: Func Spaces(const N: Int):Str;
4064: { HasChar returns True, if char, Ch, contains in string, S }
4065: Func HasChar(const Ch: Char; const S:Str):Bool;
4066: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
4067: { SubStr returns substring from string, S, separated with Separator string}
4068: Func SubStr(const S:Str; const Index: Int; const Separator:Str):Str;
4069: { SubStrEnd same to previous Func but Index numerated from the end of string }
4070: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
4071: { ReplaceString searches forall substrings,OldPattern,in a string,S,replaces with NewPattern }
4072: Func ReplaceString(S:Str; const OldPattern, NewPattern:Str):Str;
4073: Func CharInSet(const Ch: Char; const SetOfChar: TSetOfChar):Bool;
4074: { GetXYByPos is same to previous function, but returns X position in line too}
4075: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
4076: { AddSlash returns string with added slash char to Dir parameter, if needed }
4077: Func AddSlash2(const Dir: TFileName):Str;
4078: { AddPath returns FileName with Path, if FileName not contain any path }
4079: Func AddPath(const FileName, Path: TFileName): TFileName;
4080: { ExePath returns ExtractFilePath(ParamStr(0)) }
4081: Func ExePath: TFileName;
4082: Func LoadTextFile(const FileName: TFileName):Str;
4083: Proc SaveTextFile(const FileName: TFileName; const Source:Str);
4084: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4085: Func ConcatSep(const S, S2, Separator:Str):Str;
4086: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4087: Func FileEquMask(FileName, Mask: TFileName):Bool;
4088: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4089:   Masks must be separated with comma (';') }
4090: Func FileEquMasks(FileName, Masks: TFileName):Bool;
4091: Func StringEndsWith(const Str, SubStr:Str):Bool;
4092: Func ExtractFilePath2(const FileName:Str):Str;
4093: Func StrToOem(const AnsiStr:Str):Str;
4094: { StrToOem translates a string from the Windows character set into the OEM character set. }
4095: Func OemToAnsiStr(const OemStr:Str):Str;
4096: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4097: Func IsEmptyStr(const S:Str; const EmptyChars: TCharSet):Bool;
4098: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4099: Func ReplaceStr(const S, Srch, Replace:Str):Str;
4100: { Returns string with every occurrence of Srch string replaced with Replace string. }
4101: Func DelSpace(const S:Str):Str;
4102: { DelSpace return a string with all white spaces removed. }
4103: Func DelChars(const S:Str; Chr: Char):Str;
4104: { DelChars return a string with all Chr characters removed. }
4105: Func DelBSpace(const S:Str):Str;
4106: { DelBSpace trims leading spaces from the given string. }
4107: Func DelESpace(const S:Str):Str;
4108: { DelESpace trims trailing spaces from the given string. }
4109: Func DelRSpace(const S:Str):Str;
4110: { DelRSpace trims leading and trailing spaces from the given string. }
4111: Func DelSpace1(const S:Str):Str;
4112: { DelSpace1 return a string with all non-single white spaces removed. }
```

```
4113: Func Tab2Space(const S:Str; Numb: Byte):Str;
4114: { Tab2Space converts any tabulation chars in given string to the Numb spaces characters. }
4115: Func NPos(const C:Str; S:Str; N: Int): Int;
4116: { NPos searches for a N-th position of substring C in a given string. }
4117: Func MakeStr(C: Char; N: Int):Str;
4118: Func MS(C: Char; N: Int):Str;
4119: { MakeStr return a string of length N filled with character C. }
4120: Func AddChar(C: Char; const S:Str; N: Int):Str;
4121: { AddChar return a string left-padded to length N with characters C. }
4122: Func AddCharR(C: Char; const S:Str; N: Int):Str;
4123: { AddCharR return a string right-padded to length N with characters C. }
4124: Func LeftStr(const S:Str; N: Int):Str;
4125: { LeftStr return a string right-padded to length N with blanks. }
4126: Func RightStr(const S:Str; N: Int):Str;
4127: { RightStr return a string left-padded to length N with blanks. }
4128: Func CenterStr(const S:Str; Len: Int):Str;
4129: { CenterStr centers the characters in the string based upon the Len specified. }
4130: Func CompStr(const S1, S2:Str): Int;
4131: {CompStr compares S1 to S2,case-sensitivity.return val is -1 if S1<S2,0 if S1=S2,or 1 if S1>S2.}
4132: Func CompText(const S1, S2:Str): Int;
4133: {CompText compares S1 to S2,without case-sensitivity. return value is the same as for CompStr.}
4134: Func Copy2Symb(const S:Str; Symb: Char):Str;
4135: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4136: Func Copy2SymbDel(var S:Str; Symb: Char):Str;
4137: {Copy2SymbDel returns substr of str S from first character Symb and removes substring from S.}
4138: Func Copy2Space(const S:Str):Str;
4139: { Copy2Symb returns a substring of a string S from begining to first white space. }
4140: Func Copy2SpaceDel(var S:Str):Str;
4141: { Copy2SpaceDel returns a substring of a string S from begining to first
4142:   white space and removes this substring from S. }
4143: Func AnsiProperCase(const S:Str; const WordDelims: TCharSet):Str;
4144: { Returns string, with the first letter of each word in uppercase,
4145:   all other letters in lowercase. Words are delimited by WordDelims. }
4146: Func WordCount(const S:Str; const WordDelims: TCharSet): Int;
4147: { WordCount given a set of word delimiters, returns number of words in S. }
4148: Func WordPosition(const N: Int; const S:Str; const WordDelims: TCharSet): Int;
4149: { Given a set of word delimiters, returns start position of N'th word in S. }
4150: Func ExtractWord(N: Int; const S:Str; const WordDelims: TCharSet):Str;
4151: Func ExtractWordPos(N:Int; const S:str; const WordDelims:TCharSet;var Pos: Int):str;
4152: Func ExtractDelimited(N: Int; const S:Str; const Delims: TCharSet):Str;
4153: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4154:   delimiters, return the N'th word in S. }
4155: Func ExtractSubstr(const S:Str; var Pos: Int; const Delims: TCharSet):Str;
4156: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4157: Func IsWordPresent(const W, S:Str; const WordDelims: TCharSet):Bool;
4158: { IsWordPresent given set of word delimiters, returns True if word W is present in string S. }
4159: Func QuotedString(const S:Str; Quote: Char):Str;
4160: { QuotedString returns given string as a quoted string, using the provided Quote character. }
4161: Func ExtractQuotedString(const S:Str; Quote: Char):Str;
4162: { ExtractQuotedString removes the Quote characters from beginning and end of a quoted string,
4163:    and reduces pairs of Quote characters within the quoted string to a single character. }
4164: Func FindPart(const HelpWilds, InputStr:Str): Int;
4165: {FindPart compares a string with '?' and another,returns position of HelpWilds in InputStr. }
4166: Func IsWild(InputStr, Wilds:Str; IgnoreCase:Bool):Bool;
4167: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4168: Func XorString(const Key, Src: ShortString): ShortString;
4169: Func XorEncode(const Key, Source:Str):Str;
4170: Func XorDecode(const Key, Source:Str):Str;
4171: { ** Command line routines ** }
4172: {$IFNDEF COMPILER4_UP}
4173: Func FindCmdLineSwitch(const Switch:str;SwitchChars TCharSet;IgnoreCase: Bool):Bool;
4174: {$ENDIF}
4175: Func GetCmdLineArg(const Switch:Str; SwitchChars: TCharSet):Str;
4176: { ** Numeric string handling routines ** }
4177: Func Numb2USA(const S:Str):Str;
4178: { Numb2USA converts numeric string S to USA-format. }
4179: Func Dec2Hex(N: Longint; A: Byte):Str;
4180: Func D2H(N: Longint; A: Byte):Str;
4181: { Dec2Hex converts the given value to a hexadecimal string representation
4182:   with the minimum number of digits (A) specified. }
4183: Func Hex2Dec(const S:Str): Longint;
4184: Func H2D(const S:Str): Longint;
4185: { Hex2Dec converts the given hexadecimal string to the corresponding Int value. }
4186: Func Dec2Numb(N: Longint; A, B: Byte):Str;
4187: { Dec2Numb converts the given value to a string representation with the
4188:   base equal to B and with the minimum number of digits (A) specified. }
4189: Func Numb2Dec(S:Str; B: Byte): Longint;
4190: { Numb2Dec converts given B-based numeric string to the corresponding Int value. }
4191: Func IntToBin(Value: Longint; Digits, Spaces: Int):Str;
4192: {IntToBin converts given value to bin str representation with min number of digits specified.}
4193: Func IntToRoman(Value: Longint):Str;
4194: { IntToRoman converts the given value to a roman numeric string representation. }
4195: Func RomanToInt(const S:Str): Longint;
4196: { RomanToInt converts the given string to an Int value. If the string
4197:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4198: Proc I64ToCardinals(I: Int64; var LowPart, HighPart:Card);
4199: Proc CardinalsToI64(var I: Int64; const LowPart, HighPart:Card);
4200: ********************************** JvFileUtil;**********************************
4201: Proc CopyFileJ(const FileName, DestName:Str; ProgressControl: TControl);
```

```
4202: Proc CopyFileEx(const FileName,DestName:str;OverwriteReadOnly,ShellDialog:Bool;ProgressControl:TControl);
4203: Proc MoveFile(const FileName, DestName: TFileName);
4204: Proc MoveFileEx(const FileName, DestName: TFileName; ShellDialog:Bool);
4205: {$IFDEF COMPILER4_UP}
4206: Func GetFileSize(const FileName:Str): Int64;
4207: {$ELSE}
4208: Func GetFileSize(const FileName:Str): Longint;
4209: {$ENDIF}
4210: Func FileDateTime(const FileName:Str): TDateTime;
4211: Func HasAttr(const FileName:Str; Attr: Int):Bool;
4212: Func DeleteFiles(const FileMask:Str):Bool;
4213: Func DeleteFilesEx(const FileMasks: array of string):Bool;
4214: Func ClearDir(const Path:Str; Delete:Bool):Bool;
4215: Func NormalDir(const DirName:Str):Str;
4216: Func RemoveBackSlash(const DirName:Str):Str;
4217: Func ValidFileName(const FileName:Str):Bool;
4218: Func DirExists(Name:Str):Bool;
4219: Proc ForceDirectories(Dir:Str);
4220: Func FileLock(Handle: Int; Offset, LockSize: Longint): Int;
4221:    {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4222: {$IFDEF COMPILER4_UP}
4223: Func FileLock(Handle:Int; Offset, LockSize: Int64): Int; overload;
4224: {$ENDIF}
4225: Func FileUnlock(Handle:Int; Offset, LockSize: Longint): Int;
4226: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4227: {$IFDEF COMPILER4_UP}
4228: Func FileUnlock(Handle:Int; Offset, LockSize: Int64): Int; overload;
4229: {$ENDIF}
4230: Func GetTempDir:Str;
4231: Func GetWindowsDir:Str;
4232: Func GetSystemDir:Str;
4233: Func BrowseDirectory(var AFolderName:str; const DlgText:str; AHelpContext:THelpContext):Boolean;
4234: {$IFDEF WIN32}
4235: Func BrowseComputer(var ComputerName:str;const DlgText:str;AHelpContext:THelpContext):Bool;
4236: Func ShortToLongFileName(const ShortName:Str):Str;
4237: Func ShortToLongPath(const ShortName:Str):Str;
4238: Func LongToShortFileName(const LongName:Str):Str;
4239: Func LongToShortPath(const LongName:Str):Str;
4240: Proc CreateFileLink(const FileName, DisplayName:Str; Folder: Int);
4241: Proc DeleteFileLink(const DisplayName:Str; Folder: Int);
4242: {$ENDIF WIN32}
4243: {$IFNDEF COMPILER3_UP}
4244: Func IsPathDelimiter(const S:Str; Index: Int):Bool;
4245: {$ENDIF}
4246: Func CreateCalculatorForm(AOwner:TComponent;AHelpContext:THelpContext):TJvCalculatorForm
4247: Func CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4248: Func CreatePopupCalculator( AOwner : TComponent) : TWinControl
4249: Proc SetupPopupCalculator(PopupCalc:TWinControl;APrecision Byte; ABeepOnError: Bool)
4250:
4251: //***************************Proc SIRegister_VarHlpr(CL: TPSPascalCompiler);
4252:  Proc VariantClear( var V : Variant)
4253:  Proc VariantArrayRedim( var V : Variant; High : Int)
4254:  Proc VariantCast( const src : Variant; var dst : Variant; vt : Int)
4255:  Proc VariantCpy(const src:Variant;var dst:Variant)
4256:  Proc VariantAdd(const src:Variant;var dst:Variant)
4257:  Proc VariantSub(const src:Variant;var dst:Variant)
4258:  Proc VariantMul(const src:Variant;var dst:Variant)
4259:  Proc VariantDiv(const src:Variant;var dst:Variant)
4260:  Proc VariantMod(const src:Variant;var dst:Variant)
4261:  Proc VariantAnd(const src:Variant;var dst:Variant)
4262:  Proc VariantOr( const src:Variant;var dst:Variant)
4263:  Proc VariantXor(const src:Variant;var dst:Variant)
4264:  Proc VariantShl(const src:Variant;var dst:Variant)
4265:  Proc VariantShr(const src:Variant;var dst:Variant)
4266:  Func VariantAdd2(const V1:Variant; const V2:Variant):Variant
4267:  Func VariantSub2(const V1:Variant; const V2:Variant):Variant
4268:  Func VariantMul2(const V1:Variant; const V2:Variant):Variant
4269:  Func VariantDiv2(const V1:Variant; const V2:Variant):Variant
4270:  Func VariantMod2(const V1:Variant; const V2:Variant):Variant
4271:  Func VariantAnd2(const V1:Variant; const V2:Variant):Variant
4272:  Func VariantOr2( const V1:Variant; const V2:Variant):Variant
4273:  Func VariantXor2(const V1: Variant; const V2: Variant): Variant
4274:  Func VariantShl2(const V1  Variant; const V2: Variant): Variant
4275:  Func VariantShr2(const V1: Variant; const V2: Variant): Variant
4276:  Func VariantNot(const V1: Variant) : Variant
4277:  Func VariantNeg(const V1: Variant) : Variant
4278:  Func VariantGetElement( const V : Variant; i1 : Int) : Variant;
4279:  Func VariantGetElement1(const V : Variant; i1, i2 : Int) : Variant;
4280:  Func VariantGetElement2(const V : Variant; i1, i2, i3 : Int) : Variant;
4281:  Func VariantGetElement3(const V : Variant; i1, i2, i3, i4 : Int) : Variant;
4282:  Func VariantGetElement4(const V : Variant; i1, i2, i3, i4, i5 : Int) : Variant;
4283:  Proc VariantPutElement(var V : Variant; const data : Variant; i1 : Int);
4284:  Proc VariantPutElement1(var V : Variant; const data : Variant; i1, i2 : Int);
4285:  Proc VariantPutElement2(var V : Variant; const data : Variant; i1, i2, i3 : Int);
4286:  Proc VariantPutElement3(var V: Variant; const data : Variant; i1, i2, i3, i4 : Int);
4287:  Proc VariantPutElement4(var V: Variant; const data : Variant; i1, i2, i3, i4, i5 : Int);
4288: end;
4289:
4290: ****************************************unit uPSI_JvgUtils;*************************************
```

```
4291: Func IsEven(I: Int):Bool;
4292: Func InchesToPixels(DC: HDC; Value: Single; IsHorizontal:Bool): Int;
4293: Func CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal:Bool): Int;
4294: Proc SwapInt(var I1, I2: Int);
4295: Func Spaces(Count: Int):Str;
4296: Func DupStr(const Str:Str; Count: Int):Str;
4297: Func DupChar(C: Char; Count: Int):Str;
4298: Proc Msg(const AMsg:Str);
4299: Func RectW(R: TRect): Int;
4300: Func RectH(R: TRect): Int;
4301: Func IncColor(AColor: Longint; AOffset: Byte): Longint;
4302: Func DecColor(AColor: Longint; AOffset: Byte): Longint;
4303: Func IsItAFilledBitmap(Bmp: TBitmap):Bool;
4304: Proc DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text:Str;
4305:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4306: Proc DrawTextInRect(DC:HDC;R:TRect;const Text:str;Style:TglTextStyle;Fnt:TFont;Flags:UINT);
4307: Proc ExtTextOutExt(DC: HDC; X, Y: Int; R: TRect; const Text:Str;
4308:   Style: TglTextStyle; ADelineated, ASupress3D:Bool; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4309: Proc DrawBox(DC:HDC;var R:TRect;Style:TglBoxStyle;BackgrColor:Longint;ATransparent: Bool);
4310: Func DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4311:   BevelInner,BevelOuter:TPanelBevel;Bold:Boolean;BackgrColor:Longint;ATransparent:Bool):TRect;
4312: Proc GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Int);
4313: Proc ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4314: Proc DrawBitmapExt(DC: HDC; { DC - background & result}
4315:   SourceBitmap: TBitmap; R: TRect; X, Y: Int; //...X,Y _in_ rect!
4316:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4317:   ATransparent:Bool; TransparentColor: TColor; DisabledMaskColor: TColor);
4318: Proc CreateBitmapExt(DC: HDC; { DC - background & result}
4319:   SourceBitmap: TBitmap; R: TRect; X, Y: Int; //...X,Y _in_ rect!
4320:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4321:   ATransparent:Bool; TransparentColor: TColor; DisabledMaskColor: TColor);
4322: Proc BringParentWindowToTop(Wnd: TWinControl);
4323: Func GetParentForm(Control: TControl): TForm;
4324: Proc GetWindowImageFrom(Control:TWinContrl;X,Y:Int;ADrawSelf,ADrawChildWindows:Bool;DC:HDC)
4325: Proc GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows:Bool; DC: HDC);
4326: Proc GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4327: Func CreateRotatedFont(F: TFont; Escapement: Int): HFONT;
4328: Func FindMainWindow(const AWndClass, AWndTitle:Str): THandle;
4329: Proc CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4330: Func CalcMathString(AExpression:Str): Single;
4331: Func IIF(AExpression:Bool; IfTrue, IfFalse: Variant): Variant; overload;
4332: Func IIF(AExpression:Bool; const IfTrue, IfFalse:Str):Str; overload;
4333: Func GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4334: Proc TypeStringOnKeyboard(const S:Str);
4335: Func NextStringGridCell( Grid: TStringGrid ):Bool;
4336: Proc DrawTextExtAligned(Canvas:TCanvas;const Text:str;R:TRect;Alignment:TglAlignment;WordWrap:Bool;
4337: Proc LoadComponentFromTextFile(Component: TComponent; const FileName:Str);
4338: Proc SaveComponentToTextFile(Component: TComponent; const FileName:Str);
4339: Func ComponentToString(Component: TComponent):Str;
4340: Proc StringToComponent(Component: TComponent; const Value:Str);
4341: Func PlayWaveResource(const ResName:Str):Bool;
4342: Func UserName:Str;
4343: Func ComputerName:Str;
4344: Func CreateIniFileName:Str;
4345: Func ExpandString(const Str:Str; Len: Int):Str;
4346: Func Transliterate(const Str:Str; RusToLat:Bool):Str;
4347: Func IsSmallFonts:Bool;
4348: Func SystemColorDepth: Int;
4349: Func GetFileTypeJ(const FileName:Str): TglFileType;
4350: Func GetFileType( hFile : THandle) : DWORD;
4351: Func FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass):TControl;
4352: Func StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4353:
4354: { ****************************************Utility routines of unit classes}
4355: Func LineStart(Buffer, BufPos: PChar): PChar
4356: Func ExtractStrings(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strings:TStrings):Int
4357: Func TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4358: Proc RegisterClass( AClass : TPersistentClass)
4359: Proc RegisterClasses( AClasses : array of TPersistentClass)
4360: Proc RegisterClassAlias( AClass : TPersistentClass; const Alias :Str)
4361: Proc UnRegisterClass( AClass : TPersistentClass)
4362: Proc UnRegisterClasses( AClasses : array of TPersistentClass)
4363: Proc UnRegisterModuleClasses( Module : HMODULE)
4364: Func FindGlobalComponent( const Name :Str) : TComponent
4365: Func IsUniqueGlobalComponentName( const Name :Str) :Bool
4366: Func InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) :Bool
4367: Func InitComponentRes( const ResName :Str; Instance : TComponent) :Bool
4368: Func ReadComponentRes( const ResName :Str; Instance : TComponent) : TComponent
4369: Func ReadComponentResEx( HInstance : THandle; const ResName :Str) : TComponent
4370: Func ReadComponentResFile( const FileName :Str; Instance : TComponent) : TComponent
4371: Proc WriteComponentResFile(const FileName:Str; Instance : TComponent)
4372: Proc GlobalFixupReferences
4373: Proc GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4374: Proc GetFixupInstanceNames(Root:TComponent;const ReferenceRootName string;Names:TStrings)
4375: Proc RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName :Str)
4376: Proc RemoveFixupReferences( Root : TComponent; const RootName :Str)
4377: Proc RemoveFixups( Instance : TPersistent)
4378: Func FindNestedComponent( Root : TComponent; const NamePath :Str) : TComponent
```

```
4379: Proc BeginGlobalLoading
4380: Proc NotifyGlobalLoading
4381: Proc EndGlobalLoading
4382: Func GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4383: Func GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4384:  // AddTypeS('TWndMethod', 'Proc (var Message : TMessage)
4385:  //Func MakeObjectInstance( Method : TWndMethod) : Pointer
4386:  Proc FreeObjectInstance( ObjectInstance : Pointer)
4387: // Func AllocateHWnd( Method : TWndMethod) : HWND
4388:  Proc DeallocateHWnd( Wnd : HWND)
4389:  Func AncestorIsValid( Ancestor:TPersistent; Root, RootAncestor: TComponent) :Bool
4390: ******************************unit uPSI_SqlTimSt and DB;************************************
4391: Proc VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4392: Func VarSQLTimeStampCreate3: Variant;
4393: Func VarSQLTimeStampCreate2( const AValue :Str) : Variant;
4394: Func VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4395: Func VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4396: Func VarSQLTimeStamp : TVarType
4397: Func VarIsSQLTimeStamp( const aValue : Variant) :Bool;
4398: Func LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp): TSQLTimeStamp //beta
4399: Func UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp): TSQLTimeStamp //beta
4400: Func VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4401: Func SQLTimeStampToStr( const Format :Str; DateTime : TSQLTimeStamp):Str
4402: Func SQLDayOfWeek( const DateTime : TSQLTimeStamp) : Int
4403: Func DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4404: Func SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4405: Func TryStrToSQLTimeStamp( const S :Str; var TimeStamp : TSQLTimeStamp): Bool
4406: Func StrToSQLTimeStamp( const S :Str) : TSQLTimeStamp
4407: Proc CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4408: Func ExtractFieldName( const Fields :Str; var Pos : Int) :Str;
4409: Func ExtractFieldName( const Fields : WideString; var Pos : Int) : WideString;
4410:  //'Proc RegisterFields( const FieldClasses : array of TFieldClass)
4411: Proc DatabaseError( const Message : WideString; Component : TComponent)
4412: Proc DatabaseErrorFmt(const Message:WIdeString;const Args:array of const;Component:TComponent)
4413: Proc DisposeMem( var Buffer, Size : Int)
4414: Func BuffersEqual( Buf1, Buf2 : Pointer; Size : Int) :Bool
4415: Func GetFieldProperty(DataSet:TDataSet;Control:TComponent;const FieldName:WideStr):TField
4416: Func VarTypeToDataType( VarType : Int) : TFieldType
4417: ******************************unit JvStrings;****************************************
4418: {template functions}
4419: Func ReplaceFirst(const SourceStr, FindStr, ReplaceStr:Str):Str;
4420: Func ReplaceLast(const SourceStr, FindStr, ReplaceStr:Str):Str;
4421: Func InsertLastBlock(var SourceStr:Str; BlockStr:Str):Bool;
4422: Func RemoveMasterBlocks(const SourceStr:Str):Str;
4423: Func RemoveFields(const SourceStr:Str):Str;
4424: {http functions}
4425: Func URLEncode(const Value:Ansistr):Ansistr;// Converts string To A URLEncoded string
4426: Func URLDecode(const Value:Ansistr):Ansistr;// Converts string From A URLEncoded string
4427: {set functions}
4428: Proc SplitSet(AText:Str; AList: TStringList);
4429: Func JoinSet(AList: TStringList):Str;
4430: Func FirstOfSet(const AText:Str):Str;
4431: Func LastOfSet(const AText:Str):Str;
4432: Func CountOfSet(const AText:Str): Int;
4433: Func SetRotateRight(const AText:Str):Str;
4434: Func SetRotateLeft(const AText:Str):Str;
4435: Func SetPick(const AText:Str; AIndex: Int):Str;
4436: Func SetSort(const AText:Str):Str;
4437: Func SetUnion(const Set1, Set2:Str):Str;
4438: Func SetIntersect(const Set1, Set2:Str):Str;
4439: Func SetExclude(const Set1, Set2:Str):Str;
4440: {replace any <,> etc by &lt; &gt;}
4441: Func XMLSafe(const AText:Str):Str;
4442: {simple hash, Result can be used in Encrypt}
4443: Func Hash(const AText:Str): Int;
4444: { Base64 encode and decode a string }
4445: Func B64Encode(const S: Ansistr): Ansistr;
4446: Func B64Decode(const S: Ansistr): Ansistr;
4447: {Basic encryption from a Borland Example}
4448: Func Encrypt(const InString: Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4449: Func Decrypt(const InString: Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4450: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4451: Func EncryptB64(const InString:Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4452: Func DecryptB64(const InString:Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4453: Proc CSVToTags(Src, Dst: TStringList);
4454: // converts a csv list to a tagged string list
4455: Proc TagsToCSV(Src, Dst: TStringList);
4456: // converts a tagged string list to a csv list
4457: // only fieldnames from the first record are scanned ib the other records
4458: Proc ListSelect(Src, Dst: TStringList; const AKey, AValue:Str);
4459: {selects akey=avalue from Src and returns recordset in Dst}
4460: Proc ListFilter(Src: TStringList; const AKey, AValue:Str);
4461: {filters Src for akey=avalue}
4462: Proc ListOrderBy(Src: TStringList; const AKey:Str; Numeric: Bool);
4463: {orders a tagged Src list by akey}
4464: Func PosStr(const FindString, SourceString:Str;
4465:   StartPos: Int = 1): Int;
4466: { PosStr searches the first occurrence of a substring FindString in a string
4467:   given by SourceString with case sensitivity (upper and lower case characters
```

```
4468:   are differed). This Func returns the index value of the first character
4469:   of a specified substring from which it occurs in a given string starting with
4470:   StartPos character index. If a specified substring is not found Q_PosStr
4471:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings from www.torry.ru). }
4472: Func PosStrLast(const FindString, SourceString:Str): Int;
4473: {finds the last occurance}
4474: Func LastPosChar(const FindChar: Char; SourceString:Str): Int;
4475: Func PosText(const FindString, SourceString:Str; StartPos: Int = 1): Int;
4476: { PosText searches the first occurrence of a substring FindString in a string
4477:   given by SourceString without case sensitivity (upper and lower case characters are not differed). This
      Func returns index value of first character of a specified substring from which it occurs in a given
      string starting with Start
4478: Func PosTextLast(const FindString, SourceString:Str): Int;
4479: {finds last occurance}
4480: Func NameValuesToXML(const AText:Str):Str;
4481: {$IFDEF MSWINDOWS}
4482: Proc LoadResourceFile(AFile:Str; MemStream: TMemoryStream);
4483: {$ENDIF MSWINDOWS}
4484: Proc DirFiles(const ADir, AMask:Str; AFileList: TStringList);
4485: Proc RecurseDirFiles(const ADir:Str; var AFileList: TStringList);
4486: Proc RecurseDirProgs(const ADir:Str; var AFileList: TStringList);
4487: Proc SaveString(const AFile, AText:Str);
4488: Proc SaveStringasFile( const AFile, AText :Str)
4489: Func LoadStringJ(const AFile:Str):Str;
4490: Func LoadStringofFile( const AFile :Str) :Str
4491: Proc SaveStringtoFile( const AFile, AText :Str)
4492: Func LoadStringfromFile( const AFile :Str) :Str
4493: Func HexToColor(const AText:Str): TColor;
4494: Func UppercaseHTMLTags(const AText:Str):Str;
4495: Func LowercaseHTMLTags(const AText:Str):Str;
4496: Proc GetHTMLAnchors(const AFile:Str; AList: TStringList);
4497: Func RelativePath(const ASrc, ADst:Str):Str;
4498: Func GetToken(var Start: Int; const SourceText:Str):Str;
4499: Func PosNonSpace(Start: Int; const SourceText:Str): Int;
4500: Func PosEscaped(Start: Int; const SourceText, FindText:Str; EscapeChar: Char): Int;
4501: Func DeleteEscaped(const SourceText:Str; EscapeChar: Char):Str;
4502: Func BeginOfAttribute(Start: Int; const SourceText:Str): Int;
4503: // parses the beginning of an attribute: space + alpha character
4504: Func ParseAttribute(var Start:Int;const SourceText:str; var AName,AValue:str):Bool;
4505: //parses a name="value" attrib from Start;returns 0 when not found or else position behind attribute
4506: Proc ParseAttributes(const SourceText:Str; Attributes: TStrings);
4507: // parses all name=value attributes to the attributes TStringList
4508: Func HasStrValue(const AText, AName:Str; var AValue:Str):Bool;
4509: // checks if a name="value" pair exists and returns any value
4510: Func GetStrValue(const AText, AName, ADefault:Str):Str;
4511: // retrieves string value from a line like:
4512: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4513: // returns ADefault when not found
4514: Func GetHTMLColorValue(const AText, AName:Str; ADefault: TColor):TColor;
4515: // same for a color
4516: Func GetIntValue(const AText, AName:Str; ADefault: Int): Int;
4517: // same for an Int
4518: Func GetFloatValue(const AText, AName:Str; ADefault: Extended):Extended;
4519: // same for a float
4520: Func GetBoolValue(const AText, AName:Str):Bool;
4521: // same for Boolean but without default
4522: Func GetValue(const AText, AName:Str):Str;
4523: //retrieves string value from a line like:name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4524: Proc SetValue(var AText:Str; const AName, AValue:Str);
4525: // sets a string value in a line
4526: Proc DeleteValue(var AText:Str; const AName:Str);
4527: // deletes a AName="value" pair from AText
4528: Proc GetNames(AText:Str; AList: TStringList);
4529: // get a list of names from a string with name="value" pairs
4530: Func GetHTMLColor(AColor: TColor):Str;
4531: // converts a color value to the HTML hex value
4532: Func BackPosStr(Start: Int; const FindString, SourceString:Str): Int;
4533: // finds a string backward case sensitive
4534: Func BackPosText(Start: Int; const FindString, SourceString:Str): Int;
4535: // finds a string backward case insensitive
4536: Func PosRangeStr(Start: Int; const HeadString, TailString, SourceString:Str;
4537:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4538: // finds a text range, e.g. <TD>....</TD> case sensitive
4539: Func PosRangeText(Start: Int; const HeadString, TailString, SourceString:Str;
4540:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4541: // finds a text range, e.g. <TD>....</td> case insensitive
4542: Func BackPosRangeStr(Start: Int; const HeadString, TailString, SourceString:Str;
4543:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4544: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4545: Func BackPosRangeText(Start: Int; const HeadString, TailString, SourceString:str;
4546:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4547: // finds a text range backward, e.g. <TD>....</td> case insensitive
4548: Func PosTag(Start: Int; SourceString:Str; var RangeBegin: Int;
4549:   var RangeEnd: Int):Bool;
4550: // finds a HTML or XML tag:  <....>
4551: Func InnerTag(Start: Int; const HeadString, TailString, SourceString:Str;
4552:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4553: // finds the innertext between opening and closing tags
4554: Func Easter(NYear: Int): TDateTime;
```

```
4555: // returns the easter date of a year.
4556: Func GetWeekNumber(Today: TDateTime):Str;
4557: //gets a datecode. Returns year and weeknumber in format: YYWW
4558: Func ParseNumber(const S:Str): Int;
4559: // parse number returns the last position, starting from 1
4560: Func ParseDate(const S:Str): Int;
4561: // parse a SQL style data string from positions 1, // starts and ends with #
4562:
4563: **************************unit JvJCLUtils;***********************************
4564: PathDelim','String').SetString( '\;
4565:  DriveDelim','String').SetString( ':;
4566:  PathSep','String').SetString( ';;
4567:  AllFilesMask','String').SetString( '*.*;
4568:  PathDelim','String').SetString( '/;
4569:  AllFilesMask','String').SetString( '*;
4570:  NullHandle','LongInt').SetInt( 0);
4571:  USDecimalSeparator','String').SetString( '.;
4572:   ClassN(CL.FindClass('TOBJECT'),'EJvConvertError;
4573:   TypeS('TFileTime', 'Int;
4574:  // TypeS('TFormatSettings', 'record DecimalSeparator : Char; end;
4575:  Func SendRectMessage(Handle: THandle; Msg : Int; wParam : longint; var R : TRect): Int;
4576:  //Func SendStructMessage(Handle:THandle;Msg:Int;wParam:WPARAM;var Data): Int;
4577:  Func ReadCharsFromStream(Stream: TStream; var Buf:array of PChar;BufSize:Int):Int;
4578:  Func WriteStringToStream(Stream: TStream; const Buf: Ansistr;BufSize:Int): Int;
4579:  Func UTF8ToString( const S : UTF8String) :Str;
4580:  //ConstantN('DefaultDateOrder','').SetString('doDMY;
4581:  ConstantN('CenturyOffset','Byte').SetUInt( 60);
4582:
4583: Func VarIsInt(Value: Variant):Bool;
4584:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4585: {PosIdx returns index of first appearance of SubStr in Str.search starts at index "Index".}
4586: Func PosIdx(const SubStr, S:Str; Index: Int = 0): Int;
4587: Func PosIdxW(const SubStr, S: WideString; Index: Int = 0): Int;
4588: Func PosLastCharIdx(Ch: Char; const S:Str; Index: Int = 0): Int;
4589: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4590: Func GetWordOnPos(const S:Str; const P: Int):Str;
4591: Func GetWordOnPosW(const S: WideString; const P: Int): WideString;
4592: Func GetWordOnPos2(const S:Str; P: Int; var iBeg, iEnd: Int):Str;
4593: Func GetWordOnPos2W(const S: WideString; P: Int; var iBeg, iEnd: Int): WideString;
4594: { GetWordOnPosEx working like GetWordOnPos function, but
4595:   also returns Word position in iBeg, iEnd variables }
4596: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
4597: Func GetWordOnPosExW(const S: WideString; const P: Int; var iBeg, iEnd: Int): WideString;
4598: Func GetNextWordPosEx(const Text:Str; StartIndex: Int; var iBeg, iEnd: Int):Str;
4599: Func GetNextWordPosExW(const Text:WideString;StartIndex:Int; var iBeg,iEnd:Int):WideString;
4600: Proc GetEndPosCaret(const Text:Str; CaretX, CaretY: Int; var X, Y: Int);
4601: { GetEndPosCaret returns the caret position of the last char. For the position
4602:   after the last char of Text you must add 1 to the returned X value. }
4603: Proc GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Int;var X,Y:Int);
4604: { GetEndPosCaret returns the caret position of the last char. For the position
4605:   after the last char of Text you must add 1 to the returned X value. }
4606: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4607: Func SubStrBySeparator(const S:str;const Index:Int;const Separatr:str;StartIndex:Int=1):str;
4608: Func SubStrBySeparatorW(const S:WideString;const Index:Int;const
      Separator:WideString;StartIndex:Int:WideString;
4609: { SubStrEnd same to previous Func but Index numerated from the end of string }
4610: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
4611: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4612: Func SubWord(P: PChar; var P2: PChar):Str;
4613: Func CurrencyByWord(Value: Currency):Str;
4614: { GetLineByPos returns Line number,there symbol Pos is pointed.Lines separated with #13 symbol}
4615: Func GetLineByPos(const S:Str; const Pos: Int): Int;
4616: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4617: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
4618: Proc GetXYByPosW(const S: WideString; const Pos: Int; var X, Y: Int);
4619: { ReplaceString searches for all substrings, OldPattern,
4620:   in a string, S, and replaces them with NewPattern }
4621: Func ReplaceString(S:Str;const OldPattern,NewPattern:str;StartIndex:Int= 1):str;
4622: Func ReplaceStringW(S:WideString;const OldPattern,NewPattern:WideString;StartIndex:Int=1):WideString;
4623: { ConcatSep concatenate S1 and S2 strings with Separator. if S ='' then separator not included}
4624: Func ConcatSep(const S1, S2, Separator:Str):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4625: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4626: Func ConcatLeftSep(const S1,S2, Separator:str):str;{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF SUPPORTS_INLINE}
4627: { Next 4 Func for russian chars transliterating.
4628:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4629: Proc Dos2Win(var S: Ansistr);
4630: Proc Win2Dos(var S: Ansistr);
4631: Func Dos2WinRes(const S: Ansistr): Ansistr; inline; {$ENDIF SUPPORTS_INLINE}
4632: Func Win2DosRes(const S: Ansistr): Ansistr; inline; {$ENDIF SUPPORTS_INLINE}
4633: Func Win2Koi(const S: Ansistr): Ansistr;
4634: { FillString fills the string Buffer with Count Chars }
4635: Proc FillString(var Buffer:Str; Count: Int; const Value: Char); overload;
4636: Proc FillString1(var Buffer:Str; StartIndex,Count:Int; const Value: Char);overload;
4637: { MoveString copies Count Chars from Source to Dest }
4638: Proc MoveString(const Source:Str; var Dest:Str; Count: Int); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE} overload;
4639: Proc MoveString1(const Source:Str; SrcStartIdx: Int; var Dest:Str;
4640:   DstStartIdx: Int;Count:Int);{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}overload;
4641: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
```

```
4642: Proc FillWideChar(var Buffer; Count: Int; const Value: WideChar);
4643: { MoveWideChar copies Count WideChars from Source to Dest }
4644: Proc MoveWideChar(const Source;var Dest;Count:Int);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF SUPPORTS_INLINE}
4645: { FillNativeChar fills Buffer with Count NativeChars }
4646: Proc FillNativeChar(var Buffer; Count: Int; const Value: Char);// D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4647: { MoveWideChar copies Count WideChars from Source to Dest }
4648: Proc MoveNativeChar(const Source; var Dest; Count: Int);// D2009 internal error {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE}
4649: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4650: Func IsSubString(const S:Str; StartIndex: Int; const SubStr:Str):Bool;
4651: { Spaces returns string consists on N space chars }
4652: Func Spaces(const N: Int):Str;
4653: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4654: Func AddSpaces(const S:Str; const N: Int):Str;
4655: Func SpacesW(const N: Int): WideString;
4656: Func AddSpacesW(const S: WideString; const N: Int): WideString;
4657: { Func LastDateRUS for russian users only }
4658: { returns date relative to current date: 'ãâà äíÿ íàçàä' }
4659: Func LastDateRUS(const Dat: TDateTime):Str;
4660: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4661: Func CurrencyToStr(const Cur: Currency):Str;
4662: { HasChar returns True, if Char, Ch, contains in string, S }
4663: Func HasChar(const Ch: Char; const S:Str):Bool;
4664: Func HasCharW(const Ch: WideChar; const S: WideString):Bool; inline; {$ENDIF SUPPORTS_INLINE}
4665: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
4666: {$IFNDEF COMPILER12_UP}
4667: Func CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4668: {$ENDIF ~COMPILER12_UP}
4669: Func CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF SUPPORTS_INLINE}
4670: Func CountOfChar(const Ch: Char; const S:Str): Int;
4671: Func DefStr(const S:str; Default:str):str; {$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4672: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4673: Func StrLICompW2(S1, S2: PWideChar; MaxLen: Int): Int;
4674: Func StrPosW(S, SubStr: PWideChar): PWideChar;
4675: Func StrLenW(S: PWideChar): Int;
4676: Func TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4677: Func TrimLeftW(const S:WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4678: Func TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4679: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4680: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4681:  Func GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4682: Func GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4683: Proc SetBitmapPixelFormat(Bitmap: TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod)
4684: Func BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4685: Proc GrayscaleBitmap( Bitmap : TBitmap)
4686: Func BitmapToMemory( Bitmap : TBitmap; Colors : Int) : TStream
4687: Proc SaveBitmapToFile( const Filename :Str; Bitmap : TBitmap; Colors : Int)
4688: Func ScreenPixelFormat : TPixelFormat
4689: Func ScreenColorCount : Int
4690: Proc TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4691: Func ZoomImage( ImageW, ImageH, MaxW, MaxH : Int; Stretch :Bool): TPoint
4692: //  SIRegister_TJvGradient(CL);
4693:
4694: {***************************************** files routines}
4695: Proc SetDelimitedText(List: TStrings; const Text:Str; Delimiter: Char);
4696: const
4697:   {$IFDEF MSWINDOWS}
4698:   DefaultCaseSensitivity = False;
4699:   {$ENDIF MSWINDOWS}
4700:   {$IFDEF UNIX}
4701:   DefaultCaseSensitivity = True;
4702:   {$ENDIF UNIX}
4703: { GenTempFileName returns temporary file name on
4704:   drive, there FileName is placed }
4705: Func GenTempFileName(FileName:Str):Str;
4706: { GenTempFileNameExt same to previous function, but
4707:   returning filename has given extension, FileExt }
4708: Func GenTempFileNameExt(FileName:Str; const FileExt:Str):Str;
4709: { ClearDir clears folder Dir }
4710: Func ClearDir(const Dir:Str):Bool;
4711: { DeleteDir clears and than delete folder Dir }
4712: Func DeleteDir(const Dir:Str):Bool;
4713: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4714: Func FileEquMask(FileName,Mask:TFileName;CaseSensitive:Boolean=DefaultCaseSensitivity):Bool;
4715: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4716:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4717: Func FileEquMasks(FileName, Masks: TFileName;
4718:   CaseSensitive:Bool = DefaultCaseSensitivity):Bool;
4719: Func DeleteFiles(const Folder: TFileName; const Masks:Str):Bool;
4720: {$IFDEF MSWINDOWS}
4721: { LZFileExpand expand file, FileSource,
4722:   into FileDest. Given file must be compressed, using MS Compress program }
4723: Func LZFileExpand(const FileSource, FileDest:Str):Bool;
4724: {$ENDIF MSWINDOWS}
4725: { FileGetInfo fills SearchRec record for specified file attributes}
4726: Func FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec):Bool;
4727: { HasSubFolder returns True, if folder APath contains other folders }
4728: Func HasSubFolder(APath: TFileName):Bool;
```

```
4729: { IsEmptyFolder returns True, if there are no files or
4730:   folders in given folder, APath}
4731: Func IsEmptyFolder(APath: TFileName):Bool;
4732: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4733: Func AddSlash(const Dir: TFileName):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4734: { AddPath returns FileName with Path, if FileName not contain any path }
4735: Func AddPath(const FileName, Path: TFileName): TFileName;
4736: Func AddPaths(const PathList, Path:Str):Str;
4737: Func ParentPath(const Path: TFileName): TFileName;
4738: Func FindInPath(const FileName, PathList:Str): TFileName;
4739: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4740: Func DeleteReadOnlyFile(const FileName: TFileName):Bool;
4741: { HasParam returns True, if program running with specified parameter, Param }
4742: Func HasParam(const Param:Str):Bool;
4743: Func HasSwitch(const Param:Str):Bool;
4744: Func Switch(const Param:Str):Str;
4745: { ExePath returns ExtractFilePath(ParamStr(0)) }
4746: Func ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4747: Func CopyDir(const SourceDir, DestDir: TFileName):Bool;
4748: //Func FileTimeToDateTime(const FT: TFileTime): TDateTime;
4749: Proc FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4750: Func MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4751: {**** Graphic routines }
4752: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4753: Func IsTTFontSelected(const DC: HDC):Bool;
4754: Func KeyPressed(VK: Int):Bool;
4755: Func isKeypressed:Bool;  //true if key on memo2 (shell output) is pressed
4756: { TrueInflateRect inflates rect in other method, than InflateRect API Func }
4757: Func TrueInflateRect(const R: TRect; const I: Int): TRect;
4758: {**** Color routines }
4759: Proc RGBToHSV(R, G, B: Int; var H, S, V: Int);
4760: Func RGBToBGR(Value:Card):Card;
4761: //Func ColorToPrettyName(Value: TColor):Str;
4762: //Func PrettyNameToColor(const Value:Str): TColor;
4763: {**** other routines }
4764: Proc SwapInt(var Int1, Int2:Int);{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4765: Func IntPower(Base, Exponent:Int): Int;
4766: Func ChangeTopException(E: TObject):TObject;//Linux version writes error message to ErrOutput
4767: Func StrToBool(const S:Str):Bool;
4768: Func Var2Type(V: Variant; const DestVarType: Int): Variant;
4769: Func VarToInt(V: Variant): Int;
4770: Func VarToFloat(V: Variant): Double;
4771: {following functions are not documented cause not work properly sometimes,so do not use them}
4772: // (rom) ReplaceStrings1, GetSubStr removed
4773: Func GetLongFileName(const FileName:Str):Str;
4774: Func FileNewExt(const FileName, NewExt: TFileName): TFileName;
4775: Func GetParameter:Str;
4776: Func GetComputerID:Str;
4777: Func GetComputerName:Str;
4778: {**** string routines }
4779: { ReplaceAllStrings searches for all substrings, Words,
4780:   in a string, S, and replaces them with Frases with the same Index. }
4781: Func ReplaceAllStrings(const S:Str; Words, Frases: TStrings):Str;
4782: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4783:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with same
    Index, and then update NewSelStart variable }
4784: Func ReplaceStrings(const S:str;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):str;
4785: { CountOfLines calculates the lines count in a string, S,
4786:   each line must be separated from another with CrLf sequence }
4787: Func CountOfLines(const S:Str): Int;
4788: { DeleteLines deletes all lines from strings which in the words,  words.
4789:   The word of will be deleted from strings. }
4790: Proc DeleteOfLines(Ss: TStrings; const Words: array of string);
4791: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4792:   Lines contained only spaces also deletes. }
4793: Proc DeleteEmptyLines(Ss: TStrings);
4794: { SQLAddWhere addes or modifies existing where-statement, where,
4795:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4796:   it must be started on the begining of any line }
4797: Proc SQLAddWhere(SQL: TStrings; const Where:Str);
4798: {**** files routines - }
4799: {$IFDEF MSWINDOWS}
4800: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4801:   Resource can be compressed using MS Compress program}
4802: Func ResSaveToFile(const Typ,Name:str;const Compressed:Bool;const FileName:str):Bool;
4803: Func ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:str):Bool;
4804: Func ResSaveToString(Instance: HINST;const Typ,Name:str; var S:str):Boolean;
4805: {$ENDIF MSWINDOWS}
4806: { IniReadSection read section, Section, from ini-file,
4807:   IniFileName, into strings, Ss.This Func reads ALL strings from specified section.
4808:   Note: TIninFile.ReadSection Func reads only strings with '=' symbol.}
4809: Func IniReadSection(const IniFileName:TFileName;const Section:str; Ss: TStrings): Bool;
4810: { LoadTextFile load text file, FileName, into string }
4811: Func LoadTextFile(const FileName: TFileName):Str;
4812: Proc SaveTextFile(const FileName: TFileName; const Source:Str);
4813: {ReadFolder reads fileslist from diskfolder,Folder,that are equal to mask,Mask,into strings, FileList}
4814: Func ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Int;
4815: Func ReadFolders(const Folder: TFileName; FolderList: TStrings): Int;
4816: { RATextOut same with TCanvas.TextOut procedure,but can clipping drawing with rectangle,RClip.}
```

```
4817: Proc RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S:Str);
4818: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output}
4819: Func RATextOutEx(Canvas:TCanvas;const R,RClip:TRect;const S:str;const CalcHeight:Bool):Int;
4820: { RATextCalcHeight calculate needed height for
4821:   correct output, using RATextOut or RATextOutEx functions }
4822: Func RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S:Str): Int;
4823: { Cinema draws some visual effect }
4824: Proc Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4825: { Roughed fills rect with special 3D pattern }
4826: Proc Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert:Bool);
4827: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
      height,AWidth,AHeight and placed on a specified Index,Index in source bitmap }
4828: Func BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Int): TBitmap;
4829: { TextWidth calculate text with for writing using standard desktop font }
4830: Func TextWidth(const AStr:Str): Int;
4831: { TextHeight calculate text height for writing using standard desktop font }
4832: Func TextHeight(const AStr:Str): Int;
4833: Proc SetChildPropOrd(Owner: TComponent; const PropName:Str; Value: Longint);
4834: Proc Error(const Msg:Str);
4835: Proc ItemHtDrawEx(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
4836:   const HideSelColor:Bool; var PlainItem:Str; var Width: Int; CalcWidth:Bool);
4837: {ex.Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>'}
4838: Func ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4839:   const State: TOwnerDrawState; const Text:Str; const HideSelColor:Bool):Str;
4840: Func ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4841:   const State: TOwnerDrawState; const Text:Str; const HideSelColor:Bool): Int;
4842: Func ItemHtPlain(const Text:Str):Str;
4843: { ClearList - clears list of TObject }
4844: Proc ClearList(List: TList);
4845: Proc MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4846: Proc ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4847: { RTTI support }
4848: Func GetPropType(Obj: TObject; const PropName:Str): TTypeKind;
4849: Func GetPropStr(Obj: TObject; const PropName:Str):Str;
4850: Func GetPropOrd(Obj: TObject; const PropName:Str): Int;
4851: Func GetPropMethod(Obj: TObject; const PropName:Str): TMethod;
4852: Proc PrepareIniSection(Ss: TStrings);
4853: {following functions are not documented because they are don't work properly,so don't use them}
4854: // (rom) from JvBandWindows to make it obsolete
4855: Func PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4856: // (rom) from JvBandUtils to make it obsolete
4857: Func iif(const Test:Bool; const ATrue, AFalse: Variant): Variant;
4858: Proc CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4859: Func CreateIconFromClipboard: TIcon;
4860: { begin JvIconClipboardUtils } { Icon clipboard routines }
4861: Func CF_ICON: Word;
4862: Proc AssignClipboardIcon(Icon: TIcon);
4863: { Real-size icons support routines (32-bit only) }
4864: Proc GetIconSize(Icon: HICON; var W, H: Int);
4865: Func CreateRealSizeIcon(Icon: TIcon): HICON;
4866: Proc DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Int);
4867: {end JvIconClipboardUtils }
4868: Func CreateScreenCompatibleDC: HDC;
4869: Func InvalidateRect(hWnd: HWND; const lpRect:TRect; bErase:BOOL):BOOL; overload;{$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF}
4870: Func InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4871: { begin JvRLE } // (rom) changed API for inclusion in JCL
4872: Proc RleCompressTo(InStream, OutStream: TStream);
4873: Proc RleDecompressTo(InStream, OutStream: TStream);
4874: Proc RleCompress(Stream: TStream);
4875: Proc RleDecompress(Stream: TStream);
4876: { end JvRLE } { begin JvDateUtil }
4877: Func CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4878: Func IsLeapYear(AYear: Int):Bool;
4879: Func DaysInAMonth(const AYear, AMonth: Word): Word;
4880: Func DaysPerMonth(AYear, AMonth: Int): Int;
4881: Func FirstDayOfPrevMonth: TDateTime;
4882: Func LastDayOfPrevMonth: TDateTime;
4883: Func FirstDayOfNextMonth: TDateTime;
4884: Func ExtractDay(ADate: TDateTime): Word;
4885: Func ExtractMonth(ADate: TDateTime): Word;
4886: Func ExtractYear(ADate: TDateTime): Word;
4887: Func IncDate(ADate: TDateTime; Days, Months, Years: Int): TDateTime;
4888: Func IncDay(ADate: TDateTime;Delta:Int):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4889: Func IncMonth(ADate: TDateTime; Delta: Int): TDateTime;
4890: Func IncYear(ADate: TDateTime; Delta: Int): TDateTime;
4891: Func ValidDate(ADate: TDateTime):Bool;
4892: Proc DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4893: Func MonthsBetween(Date1, Date2: TDateTime): Double;
4894: Func DaysInPeriod(Date1, Date2: TDateTime): Longint;
4895: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4896: Func DaysBetween(Date1, Date2: TDateTime): Longint;
4897: { The same as previous but if Date2 < Date1 result = 0 }
4898: Func IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Int): TDateTime;
4899: Func IncHour(ATime: TDateTime; Delta: Int): TDateTime;
4900: Func IncMinute(ATime: TDateTime; Delta: Int): TDateTime;
4901: Func IncSecond(ATime: TDateTime; Delta: Int): TDateTime;
4902: Func IncMSec(ATime: TDateTime; Delta: Int): TDateTime;
4903: Func CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
```

```
4904: { String to date conversions }
4905: Func GetDateOrder(const DateFormat:Str): TDateOrder;
4906: Func MonthFromName(const S:Str; MaxLen: Byte): Byte;
4907: Func StrToDateDef(const S:Str; Default: TDateTime): TDateTime;
4908: Func StrToDateFmt(const DateFormat, S:Str): TDateTime;
4909: Func StrToDateFmtDef(const DateFormat, S:Str; Default: TDateTime): TDateTime;
4910: //Func DefDateFormat(AFourDigitYear:Bool):Str;
4911: //Func DefDateMask(BlanksChar: Char; AFourDigitYear:Bool):Str;
4912: Func FormatLongDate(Value: TDateTime):Str;
4913: Func FormatLongDateTime(Value: TDateTime):Str;
4914: { end JvDateUtil }
4915: Func BufToBinStr(Buf: Pointer; BufSize: Int):Str;
4916: Func BinStrToBuf(Value:Str; Buf: Pointer; BufSize: Int): Int;
4917: { begin JvStrUtils } { ** Common string handling routines ** }
4918: {$IFDEF UNIX}
4919: Func iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes:Card;
4920:   const ToCode, FromCode: Ansistr):Bool;
4921: Func iconvString(const S, ToCode, FromCode: Ansistr):Str;
4922: Func iconvWideString(const S: WideString; const ToCode, FromCode:Ansistr):WideString;
4923: Func OemStrToAnsi(const S: Ansistr): Ansistr;
4924: Func AnsiStrToOem(const S: Ansistr): Ansistr;
4925: {$ENDIF UNIX}
4926: Func StrToOem(const AnsiStr: Ansistr): Ansistr;
4927: { StrToOem translates a string from the Windows character set into the OEM character set. }
4928: Func OemToAnsiStr(const OemStr: Ansistr): Ansistr;
4929: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4930: Func IsEmptyStr(const S:Str; const EmptyChars: TSysCharSet):Bool;
4931: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4932: Func ReplaceStr(const S, Srch, Replace:Str):Str;
4933: { Returns string with every occurrence of Srch string replaced with Replace string. }
4934: Func DelSpace(const S:Str):Str;
4935: { DelSpace return a string with all white spaces removed. }
4936: Func DelChars(const S:Str; Chr: Char):Str;
4937: { DelChars return a string with all Chr characters removed. }
4938: Func DelBSpace(const S:Str):Str;
4939: { DelBSpace trims leading spaces from given string. }
4940: Func DelESpace(const S:Str):Str;
4941: { DelESpace trims trailing spaces from  given string. }
4942: Func DelRSpace(const S:Str):Str;
4943: { DelRSpace trims leading and trailing spaces from the given string. }
4944: Func DelSpace1(const S:Str):Str;
4945: { DelSpace1 return a string with all non-single white spaces removed. }
4946: Func Tab2Space(const S:Str; Numb: Byte):Str;
4947: { Tab2Space converts any tabulation character in given string to the
4948:   Numb spaces characters. }
4949: Func NPos(const C:Str; S:Str; N: Int): Int;
4950: { NPos searches for a N-th position of substring C in a given string. }
4951: Func MakeStr(C: Char; N: Int):Str; overload;
4952: {$IFNDEF COMPILER12_UP}
4953: Func MakeStr(C: WideChar; N: Int): WideString; overload;
4954: {$ENDIF !COMPILER12_UP}
4955: Func MS(C: Char; N: Int):Str; {$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4956: { MakeStr return a string of length N filled with character C. }
4957: Func AddChar(C: Char; const S:Str; N: Int):Str;
4958: { AddChar return a string left-padded to length N with characters C. }
4959: Func AddCharR(C: Char; const S:Str; N: Int):Str;
4960: { AddCharR return a string right-padded to length N with characters C. }
4961: Func LeftStr(const S:Str; N: Int):Str;
4962: { LeftStr return a string right-padded to length N with blanks. }
4963: Func RightStr(const S:Str; N: Int):Str;
4964: { RightStr return a string left-padded to length N with blanks. }
4965: Func CenterStr(const S:Str; Len: Int):Str;
4966: { CenterStr centers the characters in the string based upon the Len specified. }
4967: Func CompStr(const S1, S2:Str): Int; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4968: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4969:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4970: Func CompText(const S1, S2:Str):Int; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4971: { CompText compares S1 to S2,without case-sensitivity. return value is same as for CompStr. }
4972: Func Copy2Symb(const S:Str; Symb: Char):Str;
4973: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4974: Func Copy2SymbDel(var S:Str; Symb: Char):Str;
4975: { Copy2SymbDel returns a substring of a string S from begining to first
4976:   character Symb and removes this substring from S. }
4977: Func Copy2Space(const S:Str):Str;
4978: { Copy2Symb returns a substring of a string S from begining to first white space. }
4979: Func Copy2SpaceDel(var S:Str):Str;
4980: { Copy2SpaceDel returns a substring of a string S from begining to first
4981:   white space and removes this substring from S. }
4982: Func AnsiProperCase(const S:Str; const WordDelims: TSysCharSet):Str;
4983: { Returns string, with the first letter of each word in uppercase,
4984:   all other letters in lowercase. Words are delimited by WordDelims. }
4985: Func WordCount(const S:Str; const WordDelims: TSysCharSet): Int;
4986: { WordCount given a set of word delimiters, returns number of words in S. }
4987: Func WordPosition(const N: Int; const S:Str; const WordDelims: TSysCharSet):Int;
4988: { Given a set of word delimiters, returns start position of N'th word in S. }
4989: Func ExtractWord(N: Int; const S:Str; const WordDelims: TSysCharSet):Str;
4990: Func ExtractWordPos(N:Int;const S:str;const WordDelims:TSysCharSet;var Pos:Int):str;
4991: Func ExtractDelimited(N: Int; const S:Str;const Delims: TSysCharSet):Str;
4992: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
```

```
4993:   delimiters, return the N'th word in S. }
4994: Func ExtractSubstr(const S:Str; var Pos: Int;const Delims: TSysCharSet):Str;
4995: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4996:   that started from position Pos. }
4997: Func IsWordPresent(const W, S:Str; const WordDelims: TSysCharSet):Bool;
4998: {IsWordPresent given a set of word delimiters, returns True if word W is present in string S.}
4999: Func QuotedString(const S:Str; Quote: Char):Str;
5000: {QuotedString returns the given string as a quoted string, using provided Quote character. }
5001: Func ExtractQuotedString(const S:Str; Quote: Char):Str;
5002: { ExtractQuotedString removes the Quote characters from beginning and
5003:   end of a quoted string,and reduces pairs of Quotecharacters within quoted string to single character.}
5004: Func FindPart(const HelpWilds, InputStr:Str): Int;
5005: { FindPart compares a string with '?' and another, returns position of HelpWilds in InputStr.}
5006: Func IsWild(InputStr, Wilds:Str; IgnoreCase:Bool):Bool;
5007: { IsWild compares InputString with WildCard string and returns True if corresponds. }
5008: Func XorString(const Key, Src: ShortString): ShortString;
5009: Func XorEncode(const Key, Source:Str):Str;
5010: Func XorDecode(const Key, Source:Str):Str;
5011: { ** Command line routines ** }
5012: Func GetCmdLineArg(const Switch:Str; ASwitchChars: TSysCharSet):Str;
5013: { ** Numeric string handling routines ** }
5014: Func Numb2USA(const S:Str):Str;
5015: { Numb2USA converts numeric string S to USA-format. }
5016: Func Dec2Hex(N: Longint; A: Byte):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
5017: { Dec2Hex converts the given value to a hexadecimal string representation
5018:   with the minimum number of digits (A) specified. }
5019: Func Hex2Dec(const S:Str): Longint;
5020: { Hex2Dec converts the given hexadecimal string to the corresponding Int value. }
5021: Func Dec2Numb(N: Int64; A, B: Byte):Str;
5022: { Dec2Numb converts the given value to a string representation with the
5023:   base equal to B and with the minimum number of digits (A) specified. }
5024: Func Numb2Dec(S:Str; B: Byte): Int64;
5025: { Numb2Dec converts the given B-based numeric string to corresponding Int value. }
5026: Func IntToBin(Value: Longint; Digits, Spaces: Int):Str;
5027: { IntToBin converts the given value to a binary string representation
5028:   with the minimum number of digits specified. }
5029: Func IntToRoman(Value: Longint):Str;
5030: { IntToRoman converts the given value to a roman numeric string representation. }
5031: Func RomanToInt(const S:Str): Longint;
5032: { RomanToInt converts the given string to an Int value. If the string
5033:   doesn't contain a valid roman numeric value, the 0 value is returned. }
5034: Func FindNotBlankCharPos(const S:Str): Int;
5035: Func FindNotBlankCharPosW(const S: WideString): Int;
5036: Func AnsiChangeCase(const S:Str):Str;
5037: Func WideChangeCase(const S:Str):Str;
5038: Func StartsText(const SubStr, S:Str):Bool;
5039: Func EndsText(const SubStr, S:Str):Bool;
5040: Func DequotedStr(const S:Str; QuoteChar: Char = '''''):Str;
5041: Func AnsiDequotedStr(const S:Str; AQuote:Char):str;//follow Delphi 2009's "Ansi" prefix
5042: {end JvStrUtils}
5043: {$IFDEF UNIX}
5044: Func GetTempFileName(const Prefix: Ansistr):Ansistr;
5045: {$ENDIF UNIX}
5046: { begin JvFileUtil }
5047: Func FileDateTime(const FileName:Str): TDateTime;
5048: Func HasAttr(const FileName:Str; Attr: Int):Bool;
5049: Func DeleteFilesEx(const FileMasks: array of string):Bool;
5050: Func NormalDir(const DirName:Str):Str;
5051: Func RemoveBackSlash(const DirName:Str):Str; //only for Windows/DOS Paths
5052: Func ValidFileName(const FileName:Str):Bool;
5053: {$IFDEF MSWINDOWS}
5054: Func FileLock(Handle: Int; Offset, LockSize: Longint): Int; overload;
5055: Func FileLock(Handle: Int; Offset, LockSize: Int64): Int; overload;
5056: Func FileUnlock(Handle: Int; Offset, LockSize: Longint): Int; overload;
5057: Func FileUnlock(Handle: Int; Offset, LockSize: Int64): Int;overload;
5058: {$ENDIF MSWINDOWS}
5059: Func GetWindowsDir:Str;
5060: Func GetSystemDir:Str;
5061: Func ShortToLongFileName(const ShortName:Str):Str;
5062: Func LongToShortFileName(const LongName:Str):Str;
5063: Func ShortToLongPath(const ShortName:Str):Str;
5064: Func LongToShortPath(const LongName:Str):Str;
5065: {$IFDEF MSWINDOWS}
5066: Proc CreateFileLink(const FileName, DisplayName:Str; Folder: Int);
5067: Proc DeleteFileLink(const DisplayName:Str; Folder: Int);
5068: {$ENDIF MSWINDOWS}
5069: { end JvFileUtil }
5070: // Works like PtInRect but includes all edges in comparision
5071: Func PtInRectInclusive(R: TRect; Pt: TPoint):Bool;
5072: // Works like PtInRect but excludes all edges from comparision
5073: Func PtInRectExclusive(R: TRect; Pt: TPoint):Bool;
5074: Func FourDigitYear:Bool; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5075: Func IsFourDigitYear:Bool;
5076: { moved from JvJVCLUTils }
5077: //Open an object with the shell (url or something like that)
5078: Func OpenObject(const Value:Str):Bool; overload;
5079: Func OpenObject(Value: PChar):Bool; overload;
5080: {$IFDEF MSWINDOWS}
5081: //Raise the last Exception
```

```
5082: Proc RaiseLastWin32; overload;
5083: Proc RaiseLastWin32(const Text:Str); overload;
5084: //Raise last Exception with a small comment from your part {GetFileVersion returns most significant 32
      bits of a file's binary version number. Typically, this includes major/minor version placed together in
      one 32-bit Int.
5085: Func GetFileVersion(const AFileName:Str):Card;
5086: {$EXTERNALSYM GetFileVersion}
5087: //Get version of Shell.dll
5088: Func GetShellVersion:Card;
5089: {$EXTERNALSYM GetShellVersion}
5090: // CD functions  on HW
5091: Proc OpenCdDrive;
5092: Proc CloseCdDrive;
5093: // returns True if Drive is accessible
5094: Func DiskInDrive(Drive: Char):Bool;
5095: {$ENDIF MSWINDOWS}
5096: //Same as linux Func ;)
5097: Proc PError(const Text:Str);
5098: // execute a program without waiting
5099: Proc Exec(const FileName, Parameters, Directory:Str);
5100: // execute a program and wait for it to finish
5101: Func ExecuteAndWait(CmdLine:str;const WorkingDirectory:str;Visibility:Int=SW_SHOW):Int;
5102: // returns True if this is the first instance of the program that is running
5103: Func FirstInstance(const ATitle:Str):Bool;
5104: // restores a window based on it's classname and Caption. Either can be left empty
5105: // to widen the search
5106: Proc RestoreOtherInstance(const MainFormClassName, MainFormCaption:Str);
5107: // manipulate the traybar and start button
5108: Proc HideTraybar;
5109: Proc ShowTraybar;
5110: Proc ShowStartButton(Visible:Bool = True);
5111: //(rom) SC_MONITORPOWER documented as Win 95 only(rom) do some testingset monitor functions
5112: Proc MonitorOn;
5113: Proc MonitorOff;
5114: Proc LowPower;
5115: // send a key to the window named AppName
5116: Func SendKey(const AppName:Str; Key: Char):Bool;
5117: {$IFDEF MSWINDOWS}
5118: // returns list of all win currently visible, Objects property is filled with window handle
5119: Proc GetVisibleWindows(List: TStrings);
5120: Func GetVisibleWindowsF( List : TStrings):TStrings;
5121: // associates an extension to a specific program
5122: Proc AssociateExtension(const IconPath, ProgramName, Path, Extension:Str);
5123: Proc AddToRecentDocs(const FileName:Str);
5124: Func GetRecentDocs: TStringList;
5125: {$ENDIF MSWINDOWS}
5126: Func CharIsMoney(const Ch: Char):Bool;
5127: //Func StrToCurrDef(const Str:Str; Def: Currency): Currency;
5128: Func IntToExtended(I: Int): Extended;
5129: { GetChangedText works out new text given the current cursor pos & the key pressed
5130:  is not very useful in other contexts,but in thisunit as itis needed in both MemoEx and TypedEdit}
5131: Func GetChangedText(const Text:Str; SelStart, SelLength: Int; Key: Char):Str;
5132: Func MakeYear4Digit(Year, Pivot: Int): Int;
5133: //Func StrIsInt(const S:Str):Bool;
5134: Func StrIsFloatMoney(const Ps:Str):Bool;
5135: Func StrIsDateTime(const Ps:Str):Bool;
5136: Func PreformatDateString(Ps:Str):Str;
5137: Func BooleanToInt(const B:Bool): Int;
5138: Func StringToBoolean(const Ps:Str):Bool;
5139: Func SafeStrToDateTime(const Ps:Str): TDateTime;
5140: Func SafeStrToDate(const Ps:Str): TDateTime;
5141: Func SafeStrToTime(const Ps:Str): TDateTime;
5142: Func StrDelete(const psSub, psMain:Str):Str;
5143:   { returns the fractional value of pcValue}
5144: Func TimeOnly(pcValue: TDateTime): TTime;
5145: { returns the integral value of pcValue }
5146: Func DateOnly(pcValue: TDateTime): TDate;
5147: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5148: const { TDateTime value used to signify Null value}
5149:   NullEquivalentDate: TDateTime = 0.0;
5150: Func DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind):Bool;
5151: // Replacement for Win32Check to avoid platform specific warnings in D6
5152: Func OSCheck(RetVal:Bool):Bool;
5153: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5154:   Same as FileCtrl.MinimizeName in functionality (but not implementation).Included here to
5155:   not be forced to use FileCtrl unnecessarily }
5156: Func MinimizeFileName(const FileName:Str; Canvas: TCanvas; MaxLen: Int):Str;
5157: Func MinimizeText(const Text:Str; Canvas: TCanvas; MaxWidth: Int):Str;
5158: {MinimizeString trunactes longstring,S,and appends'...'symbols,if Length ofS is morethan MaxLen}
5159: Func MinimizeString(const S:Str; const MaxLen: Int):Str;
5160: Proc RunDll32Internal(Wnd:THandle;const DLLName,FuncName,CmdLine:str;CmdShow:Int= SW_SHOWDEFAULT);
5161: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion Func and calls it, returning major and minor
      version values from function. Returns False if DLL not loaded or if GetDLLVersion couldn't be found.}
5162: Func GetDLLVersion(const DLLName:Str; var pdwMajor, pdwMinor: Int):Boolean;
5163: {$ENDIF MSWINDOWS}
5164: Proc ResourceNotFound(ResID: PChar);
5165: Func EmptyRect: TRect;
5166: Func RectWidth(R: TRect): Int;
5167: Func RectHeight(R: TRect): Int;
```

```
5168: Func CompareRect(const R1, R2: TRect):Bool;
5169: Proc RectNormalize(var R: TRect);
5170: Func RectIsSquare(const R: TRect):Bool;
5171: Func RectSquare(var ARect: TRect; AMaxSize: Int = -1):Bool;
5172: //If AMaxSize = -1 ,then auto calc Square's max size
5173: {$IFDEF MSWINDOWS}
5174: Proc FreeUnusedOle;
5175: Func GetWindowsVersion:Str;
5176: Func LoadDLL(const LibName:Str): THandle;
5177: Func RegisterServer(const ModuleName:Str): Bool;
5178: Func UnregisterServer(const ModuleName:Str): Bool;
5179: {$ENDIF MSWINDOWS}
5180: { String routines }
5181: Func GetEnvVar(const VarName:Str):Str;
5182: Func AnsiUpperFirstChar(const S:str):str;//follow Delphi2009's ex. with "Ansi" prefix
5183: Func StringToPChar(var S:Str): PChar;
5184: Func StrPAlloc(const S:Str): PChar;
5185: Proc SplitCommandLine(const CmdLine:Str; var ExeName, Params:Str);
5186: Func DropT(const S:Str):Str;
5187: { Memory routines }
5188: Func AllocMemo(Size: Longint): Pointer;
5189: Func ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5190: Proc FreeMemo(var fpBlock: Pointer);
5191: Func GetMemoSize(fpBlock: Pointer): Longint;
5192: Func CompareMem(fpBlock1, fpBlock2: Pointer; Size:Card):Bool;
5193: { Manipulate huge pointers routines }
5194: Proc HugeInc(var HugePtr: Pointer; Amount: Longint);
5195: Proc HugeDec(var HugePtr: Pointer; Amount: Longint);
5196: Func HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5197: Proc HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5198: Proc HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5199: Func WindowClassName(Wnd: THandle):Str;
5200: Proc SwitchToWindow(Wnd: THandle; Restore:Bool);
5201: Proc ActivateWindow(Wnd: THandle);
5202: Proc ShowWinNoAnimate(Handle: THandle; CmdShow: Int);
5203: Proc KillMessage(Wnd: THandle; Msg:Card);
5204: { SetWindowTop put window to top without recreating window }
5205: Proc SetWindowTop(const Handle: THandle; const Top:Bool);
5206: Proc CenterWindow(Wnd: THandle);
5207: Func MakeVariant(const Values: array of Variant): Variant;
5208: { Convert dialog units to pixels and backwards }
5209: {$IFDEF MSWINDOWS}
5210: Func DialogUnitsToPixelsX(DlgUnits:Word):Word;
5211: Func DialogUnitsToPixelsY(DlgUnits:Word):Word;
5212: Func PixelsToDialogUnitsX(PixUnits:Word):Word;
5213: Func PixelsToDialogUnitsY(PixUnits:Word):Word;
5214: {$ENDIF MSWINDOWS}
5215: Func GetUniqueFileNameInDir(const Path, FileNameMask:Str):Str;
5216: {$IFDEF BCB}
5217: Func FindPrevInstance(const MainFormClass: ShortString;const ATitle:Str): THandle;
5218: Func ActivatePrevInstance(const MainFormClass: ShortString;const ATitle:str):Bool;
5219: {$ELSE}
5220: Func FindPrevInstance(const MainFormClass, ATitle:Str): THandle;
5221: Func ActivatePrevInstance(const MainFormClass, ATitle:Str):Bool;
5222: {$ENDIF BCB}
5223: {$IFDEF MSWINDOWS}
5224: { BrowseForFolderNative displays Browse For Folder dialog }
5225: Func BrowseForFolderNative(const Handle:THandle;const Title:str;var Folder:str):Bool;
5226: {$ENDIF MSWINDOWS}
5227: Proc AntiAlias(Clip: TBitmap);
5228: Proc AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Int);
5229: Proc CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5230:   ABitmap: TBitmap; const SourceRect: TRect);
5231: Func IsTrueType(const FontName:Str):Bool;
5232: // Removes all non-numeric characters from AValue and returns the resulting string
5233: Func TextToValText(const AValue:Str):Str;
5234: Func ExecRegExpr( const ARegExpr, AInputStr : RegExprString) :Bool
5235: Proc SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5236: Func ReplaceRegExpr(const ARegExpr,AInputStr,
      AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5237: Func QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5238: Func RegExprSubExpressions(const ARegExpr:str;ASubExprs:TStrings;AExtendedSyntax:bool):
5239:
5240: ****************************************************unit uPSI_JvTFUtils;
5241:  Func JExtractYear( ADate : TDateTime) : Word
5242:  Func JExtractMonth( ADate : TDateTime) : Word
5243:  Func JExtractDay( ADate : TDateTime) : Word
5244:  Func ExtractHours( ATime : TDateTime) : Word
5245:  Func ExtractMins( ATime : TDateTime) : Word
5246:  Func ExtractSecs( ATime : TDateTime) : Word
5247:  Func ExtractMSecs( ATime : TDateTime) : Word
5248:  Func FirstOfMonth( ADate : TDateTime) : TDateTime
5249:  Func GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5250:  Func GetWeeksInMonth( Year, Month : Word; StartOfWeek : Int) : Word
5251:  Proc IncBorlDOW( var BorlDOW : Int; N : Int)
5252:  Proc IncDOW( var DOW : TTFDayOfWeek; N : Int)
5253:  Proc IncDays( var ADate : TDateTime; N : Int)
5254:  Proc IncWeeks( var ADate : TDateTime; N : Int)
5255:  Proc IncMonths( var ADate : TDateTime; N : Int)
```

```
5256:  Proc IncYears( var ADate : TDateTime; N : Int)
5257:  Func EndOfMonth( ADate : TDateTime) : TDateTime
5258:  Func IsFirstOfMonth( ADate : TDateTime) :Bool
5259:  Func IsEndOfMonth( ADate : TDateTime) :Bool
5260:  Proc EnsureMonth( Month : Word)
5261:  Proc EnsureDOW( DOW : Word)
5262:  Func EqualDates( D1, D2 : TDateTime) :Bool
5263:  Func Lesser( N1, N2 : Int) : Int
5264:  Func Greater( N1, N2 : Int) : Int
5265:  Func GetDivLength( TotalLength, DivCount, DivNum : Int) : Int
5266:  Func GetDivNum( TotalLength, DivCount, X : Int) : Int
5267:  Func GetDivStart( TotalLength, DivCount, DivNum : Int) : Int
5268:  Func DOWToBorl( ADOW : TTFDayOfWeek) : Int
5269:  Func BorlToDOW( BorlDOW : Int) : TTFDayOfWeek
5270:  Func DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5271:  Proc CalcTextPos(HostRect:TRect;var TextLeft,TextTop:Int;var
       TextBounds:TRect;AFont:TFont;AAngle:Int;HAlign:TAlignment;VAlign:TJvTFVAlignment;ATxt:str)
5272:  Proc DrawAngleText(ACanvas:TCanvas;HostRect:TRect;var TextBounds:TRect;AAngle:Int;HAlign:
       TAlignment;VAlign:TJvTFVAlignment;ATxt:str)
5273:  Func JRectWidth( ARect : TRect) : Int
5274:  Func JRectHeight( ARect : TRect): Int
5275:  Func JEmptyRect : TRect
5276:  Func IsClassByName( Obj : TObject; ClassName:str):Boolean
5277:
5278: Proc SIRegister_MSysUtils(CL: TPSPascalCompiler);
5279: begin
5280:  Proc HideTaskBarButton( hWindow : HWND)
5281:  Func msLoadStr( ID : Int) :Str
5282:  Func msFormat( fmt :Str; params : array of const) :Str
5283:  Func msFileExists( const FileName :Str) :Bool
5284:  Func msIntToStr( Int : Int64) :Str
5285:  Func msStrPas( const Str : PChar) :Str
5286:  Func msRenameFile( const OldName, NewName :Str) :Bool
5287:  Func CutFileName( s :Str) :Str
5288:  Func GetVersionInfo( var VersionString :Str) : DWORD
5289:  Func FormatTime( t :Card) :Str
5290:  Func msCreateDir( const Dir :Str) :Bool
5291:  Func SetAutoRun( NeedAutoRun :Bool; AppName :Str) :Bool
5292:  Func SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5293:  Func msStrLen( Str : PChar) : Int
5294:  Func msDirectoryExists( const Directory :Str) :Bool
5295:  Func GetFolder( hWnd : hWnd; RootDir : Int; Caption :Str) :Str
5296:  Func SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5297:  Func EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT
5298:  Proc SetEditWndProc( hWnd : HWND; ptr : TObject)
5299:  Func GetTextFromFile( Filename :Str) :Str
5300:  Func IsTopMost(hWnd:HWND):Bool //'LWA_ALPHA','LongWord').SetUInt( $00000002);
5301:  Func msStrToIntDef( const s :Str; const i : Int) : Int
5302:  Func msStrToInt( s :Str) : Int
5303:  Func GetItemText( hDlg : THandle; ID : DWORD):str
5304: end;
5305:
5306: Proc SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5307: begin
5308:   //TDynFloatArray', 'array of Extended
5309:   TDynLWordArray', 'array of LongWord
5310:   TDynLIntArray', 'array of LongInt
5311:   TDynFloatMatrix', 'array of TDynFloatArray
5312:   TDynLWordMatrix', 'array of TDynLWordArray
5313:   TDynLIntMatrix', 'array of TDynLIntArray
5314:  Func SquareAll( const X : TDynFloatArray) : TDynFloatArray
5315:  Func InverseAll( const X : TDynFloatArray) : TDynFloatArray
5316:  Func LnAll( const X : TDynFloatArray) : TDynFloatArray
5317:  Func Log10All( const X : TDynFloatArray) : TDynFloatArray
5318:  Func LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5319:  Func AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5320:  Func SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5321:  Func MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5322:  Func DotProduct( const X, Y : TDynFloatArray) : Extended
5323:  Func MNorm( const X : TDynFloatArray) : Extended
5324:  Func MatrixIsRectangular( const X : TDynFloatMatrix) :Bool
5325:  Proc MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Bool;
5326:  Func MatrixIsSquare( const X : TDynFloatMatrix) :Bool
5327:  Func MatricesSameDimensions( const X, Y : TDynFloatMatrix) :Bool
5328:  Func AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5329:  Proc AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5330:  Func SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5331:  Proc SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5332:  Func MultiplyMatrixByConst( const X: TDynFloatMatrix; const K: Extended):TDynFloatMatrix
5333:  Proc MultiplyMatrixByConst2( var X: TDynFloatMatrix; const K: Extended);
5334:  Func MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5335:  Func TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5336:  Func GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5337: end;
5338:
5339: Proc SIRegister_ESBMaths(CL: TPSPascalCompiler);
5340: begin
5341:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5342:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
```

```
5343:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5344:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5345:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5346:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5347:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5348:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5349:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5350:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5351:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5352:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5353:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5354:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5355:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5356:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5357:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5358:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5359:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5360:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5361:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5362:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5363:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5364:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5365:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5366:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5367:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5368:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5369:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5370:  'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5371:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5372:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5373:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5374:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5375:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5376:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5377:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5378:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5379:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5380:  'ESBThreePi','Extended').setExtended( 9.42477796076937971 53);
5381:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5382:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5383:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5384:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5385:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5386:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5387:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5388:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5389:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5390:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5391:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5392:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5393:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5394:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5395:   //LongWord', 'Cardinal
5396:   TBitList', 'Word
5397:  Func UMul( const Num1, Num2 : LongWord) : LongWord
5398:  Func UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5399:  Func UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5400:  Func UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5401:  Func SameFloat( const X1, X2 : Extended) :Bool
5402:  Func FloatIsZero( const X : Extended) :Bool
5403:  Func FloatIsPositive( const X : Extended) :Bool
5404:  Func FloatIsNegative( const X : Extended) :Bool
5405:  Proc IncLim( var B : Byte; const Limit : Byte)
5406:  Proc IncLimSI( var B : ShortInt; const Limit : ShortInt)
5407:  Proc IncLimW( var B : Word; const Limit : Word)
5408:  Proc IncLimI( var B : Int; const Limit : Int)
5409:  Proc IncLimL( var B : LongInt; const Limit : LongInt)
5410:  Proc DecLim( var B : Byte; const Limit : Byte)
5411:  Proc DecLimSI( var B : ShortInt; const Limit : ShortInt)
5412:  Proc DecLimW( var B : Word; const Limit : Word)
5413:  Proc DecLimI( var B : Int; const Limit : Int)
5414:  Proc DecLimL( var B : LongInt; const Limit : LongInt)
5415:  Func MaxB( const B1, B2 : Byte) : Byte
5416:  Func MinB( const B1, B2 : Byte) : Byte
5417:  Func MaxSI(const B1, B2 : ShortInt) : ShortInt
5418:  Func MinSI(const B1, B2 : ShortInt) : ShortInt
5419:  Func MaxW(const B1, B2 : Word) : Word
5420:  Func MinW(const B1, B2 : Word) : Word
5421:  Func esbMaxI( const B1, B2 : Int) : Int
5422:  Func esbMinI( const B1, B2 : Int) : Int
5423:  Func MaxL(const B1, B2 : LongInt) : LongInt
5424:  Func MinL(const B1, B2 : LongInt) : LongInt
5425:  Proc SwapB( var B1, B2 : Byte)
5426:  Proc SwapSI( var B1, B2 : ShortInt)
5427:  Proc SwapW( var B1, B2 : Word)
5428:  Proc SwapI( var B1, B2 : SmallInt)
5429:  Proc SwapL( var B1, B2 : LongInt)
5430:  Proc SwapI32( var B1, B2 : Int)
5431:  Proc SwapC( var B1, B2 : LongWord)
```

```
Proc SwapInt64( var X, Y : Int64)
Func esbSign( const B : LongInt) : ShortInt
Func Max4Word(const X1, X2, X3, X4 : Word) : Word
Func Min4Word(const X1, X2, X3, X4 : Word) : Word
Func Max3Word(const X1, X2, X3 : Word) : Word
Func Min3Word(const X1, X2, X3 : Word) : Word
Func MaxBArray(const B : array of Byte) : Byte
Func MaxWArray(const B : array of Word) : Word
Func MaxSIArray(const B : array of ShortInt) : ShortInt
Func MaxIArray(const B :array of Int) : Int
Func MaxLArray(const B :array of LongInt) : LongInt
Func MinBArray(const B :array of Byte) : Byte
Func MinWArray(const B :array of Word) : Word
Func MinSIArray( const B: array of ShortInt) : ShortInt
Func MinIArray(const B :array of Int) : Int
Func MinLArray(const B :array of LongInt) : LongInt
Func SumBArray(const B :array of Byte) : Byte
Func SumBArray2( const B :array of Byte) : Word
Func SumSIArray( const B :array of ShortInt) : ShortInt
Func SumSIArray2( const B: array of ShortInt) : Int
Func SumWArray( const B :array of Word) : Word
Func SumWArray2(const B :array of Word) : LongInt
Func SumIArray(const B : array of Int) : Int
Func SumLArray(const B : array of LongInt) : LongInt
Func SumLWArray(const B :array of LongWord) : LongWord
Func ESBDigits( const X :LongWord) : Byte
Func BitsHighest( const X : LongWord) : Int
Func ESBBitsNeeded( const X : LongWord) : Int
Func esbGCD( const X, Y : LongWord) : LongWord
Func esbLCM( const X, Y : LongInt) : Int64
//Func esbLCM( const X, Y : LongInt) : LongInt
Func RelativePrime( const X, Y : LongWord) :Bool
Func Get87ControlWord : TBitList
Proc Set87ControlWord( const CWord : TBitList)
Proc SwapExt( var X, Y : Extended)
Proc SwapDbl( var X, Y : Double)
Proc SwapSing( var X, Y : Single)
Func esbSgn( const X : Extended) : ShortInt
Func Distance( const X1, Y1, X2, Y2: Extended) : Extended
Func ExtMod( const X, Y : Extended): Extended
Func ExtRem( const X, Y : Extended): Extended
Func CompMOD( const X, Y : Comp) : Comp
Proc Polar2XY( const Rho, Theta : Extended; var X, Y: Extended)
Proc XY2Polar( const X, Y : Extended; var Rho, Theta: Extended)
Func DMS2Extended( const Degs, Mins, Secs : Extended): Extended
Proc Extended2DMS( const X : Extended; var Degs, Mins, Secs: Extended)
Func MaxExt( const X, Y : Extended) : Extended
Func MinExt( const X, Y : Extended) : Extended
Func MaxEArray( const B : array of Extended) : Extended
Func MinEArray( const B : array of Extended) : Extended
Func MaxSArray( const B : array of Single) : Single
Func MinSArray( const B : array of Single) : Single
Func MaxCArray( const B : array of Comp) : Comp
Func MinCArray( const B : array of Comp) : Comp
Func SumSArray( const B : array of Single) : Single
Func SumEArray( const B : array of Extended) : Extended
Func SumSqEArray( const B : array of Extended) : Extended
Func SumSqDiffEArray( const B : array of Extended; Diff : Extended):Extended
Func SumXYEArray( const X, Y : array of Extended) : Extended
Func SumCArray( const B : array of Comp) : Comp
Func FactorialX( A : LongWord) : Extended
Func PermutationX( N, R : LongWord) : Extended
Func esbBinomialCoeff( N, R : LongWord) : Extended
Func IsPositiveEArray( const X : array of Extended) :Bool
Func esbGeometricMean( const X : array of Extended) : Extended
Func esbHarmonicMean( const X : array of Extended) : Extended
Func ESBMean( const X : array of Extended) : Extended
Func esbSampleVariance( const X : array of Extended) : Extended
Func esbPopulationVariance( const X : array of Extended) : Extended
Proc esbSampleVarianceAndMean(const X: array of Extended; var Variance, Mean :Extended)
Proc esbPopulationVarianceAndMean(const X:array of Extended;var Variance,Mean:Extended)
Func GetMedian( const SortedX : array of Extended) : Extended
Func GetMode( const SortedX : array of Extended; var Mode : Extended) :Bool
Proc GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
Func ESBMagnitude( const X : Extended) : Int
Func ESBTan( Angle : Extended) : Extended
Func ESBCot( Angle : Extended) : Extended
Func ESBCosec( const Angle : Extended) : Extended
Func ESBSec( const Angle : Extended) : Extended
Func ESBArcTan( X, Y : Extended) : Extended
Proc ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
Func ESBArcCos( const X : Extended) : Extended
Func ESBArcSin( const X : Extended) : Extended
Func ESBArcSec( const X : Extended) : Extended
Func ESBArcCosec( const X : Extended) : Extended
Func ESBLog10( const X : Extended) : Extended
Func ESBLog2( const X : Extended) : Extended
Func ESBLogBase( const X, Base : Extended) : Extended
Func Pow2( const X : Extended) : Extended
```

```
5521:  Func IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5522:  Func ESBIntPower( const X : Extended; const N : LongInt) : Extended
5523:  Func XtoY( const X, Y : Extended) : Extended
5524:  Func esbTenToY( const Y : Extended) : Extended
5525:  Func esbTwoToY( const Y : Extended) : Extended
5526:  Func LogXtoBaseY( const X, Y : Extended) : Extended
5527:  Func esbISqrt( const I : LongWord) : Longword
5528:  Func ILog2( const I : LongWord) : LongWord
5529:  Func IGreatestPowerOf2( const N : LongWord) : LongWord
5530:  Func ESBArCosh(X:Extended):Extended
5531:  Func ESBArSinh(X:Extended):Extended
5532:  Func ESBArTanh(X:Extended):Extended
5533:  Func ESBCosh(X:Extended):Extended
5534:  Func ESBSinh(X:Extended):Extended
5535:  Func ESBTanh(X:Extended):Extended
5536:  Func InverseGamma(const X: Extended) : Extended
5537:  Func esbGamma(const X : Extended): Extended
5538:  Func esbLnGamma(const X: Extended): Extended
5539:  Func esbBeta(const X, Y: Extended): Extended
5540:  Func IncompleteBeta(X :Extended; P, Q: Extended): Extended
5541: end;
5542:
5543: ****************************Int Huge Cardinal Utils***************************
5544: Func Add_uint64_WithCarry( x, y : uint64; var Carry :Bool) : uint64
5545: Func Add_uint32_WithCarry( x, y : uint32; var Carry :Bool) : uint32
5546: Func Subtract_uint64_WithBorrow( x, y : uint64; var Borrow :Bool) : uint64
5547: Func Subtract_uint32_WithBorrow( x, y : uint32; var Borrow :Bool) : uint32
5548: Func BitCount_8( Value : byte) : Int
5549: Func BitCount_16( Value : uint16) : Int
5550: Func BitCount_32( Value : uint32) : Int
5551: Func BitCount_64( Value : uint64) : Int
5552: Func CountSetBits_64( Value : uint64) : Int TPrimalityTestNoticeProc',
5553: Proc ( CountPrimalityTests : Int)
5554: Func gcdhuge( a, b : THugeCardinal) : THugeCardinal
5555: Func lcmhuge( a, b : THugeCardinal) : THugeCardinal
5556: Func isCoPrime( a, b : THugeCardinal) :Bool
5557: Func isProbablyPrime(p:THugeCardinal;OnProgress:TProgress;var wasAborted:bool):boolean
5558: Func hasSmallFactor( p : THugeCardinal) :Bool
5559: Func GeneratePrime(NumBits:
      Int;OnProgress:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;PassCount:Int;Pool1:TMemoryStreamPool;var
      Prime:THugeCardinal;var NumbersTested:Int):bool
5560:  Func Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5561:  Const('StandardExponent','LongInt'( 65537);
5562:  Proc Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:Int;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPool;var
      Numbers
5563:  Func Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal:Bool')
5564:  Func AnsiBytesOf(const S:Str): TBytes;
5565:  Func GetBytes(value:Str): TBytes;
5566:  Func GetBytesInt(value: Smallint): TBytes;
5567:  Func HugeToBase10(const AHuge: THugeCardinal):Str;
5568:
5569: (*----------------------------------------------------------------------------*)
5570: Proc SIRegister_uTPLb_StreamUtils(CL: TPSPascalCompiler);
5571: begin
5572:  SIRegister_TDesalinationWriteStream(CL);
5573:  // CL.AddTypeS('TByteDynArray', 'array of Byte
5574:  Func CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
5575:  Proc CopyMemoryStream( Source, Destination : TMemoryStream)
5576:  Proc BurnMemoryStream( Destructo : TMemoryStream)
5577:  Proc BurnMemory( var Buff, BuffLen : integer)
5578:  Proc BurnMemoryString( var Buff:Str; BuffLen : integer)
5579:  Proc BurnMemoryInteger( var Buff: Integer; BuffLen : integer)
5580:  Proc BurnMemoryString2( Buff:Str; BuffLen : integer)
5581:  Proc BurnMemoryInteger2( Buff: Integer; BuffLen : integer)
5582:  (Proc BurnMemoryByteArray( var array of Byte; BuffLen : integer)
5583:  Proc BurnMemoryByteArray2( var buff: TByteDynArray; BuffLen : integer)
5584:  Proc BurnMemoryByteArray3( buff: TByteDynArray; BuffLen : integer)
5585:  Proc ZeroFillStream( Stream : TMemoryStream)
5586:  Proc RandomFillStream( Stream : TMemoryStream)
5587:  Proc XOR_Streams2( Dest, Srce : TMemoryStream)
5588:  Proc XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
5589:  Func CompareMemoryStreams( S1, S2 : TMemoryStream) :Bool
5590:  Func Stream_to_Base64( Source : TStream) : Ansistr
5591:  Func Stream_to_Base642( Source : TStream) : Ansistr
5592:  Proc Base64_to_stream2( const Base64 : Ansistr; Destin : TStream)
5593:  Func Stream_to_Ansistr( Source : TStream) : Ansistr
5594:  Func StreamtoAnsistr( Source : TStream) : Ansistr
5595:  Func StreamToString( Source : TStream) :Str
5596:  Func StreamToString2( Source : TStream) :Str
5597:  Func StreamToString3( Source : TStream) :Str
5598:  Func StreamToString4( Source : TStream) :Str
5599:   Ansistr_to_stream( const Value : Ansistr; Destin : TStream)
5600:  Func CompareFiles(const FN1,FN2:str;Breathe:TNotifyEvent;BreathingSender:TObject):bool;
5601:  Func FileSize( const FileName :Str) : int64
5602:  Func Stream_to_decimalbytes( Source : TStream) :Str
5603:  Func StreamToDecimalbytes( Source : TStream) :Str
5604:  Func StreamtoOrd( Source : TStream) :Str
5605:  Func StreamToByte( Source : TStream) :Str
```

```
5606:  Func DisplayStream( Stream : TStream) :Str
5607:  Proc Base64_to_streamBytes( const Base64: TBytes; Destin: TStream);
5608:  Func Stream_to_Base64Bytes(ASource: TStream; const ATransform: TBytes): TBytes;
5609:  Func Stream_To_Hex(ASource: TStream): Ansistr;
5610:  // from unit uPSI_uTPLb_PointerArithmetic;
5611:  Proc ClearMemory( Stream : TMemoryStream; Offset, CountBytes : integer)
5612:  Func ReadMem(Source:TStream;Destin:TMemoryStream;DestinOffset,CountBytes:integer):int);
5613:  Func WriteMem(Source:TMemoryStream;SourceOffset:int;Destin:TStream;CountBytes:int):int;
5614: end;
5615:
5616: Proc SIRegister_EwbCoreTools(CL: TPSPascalCompiler);
5617: begin
5618:  //CL.AddDelphiFunction('Func IsWinXPSP2OrLater( ) :Bool
5619:  Func ColorToHTML2( const Color : TColor) :Str
5620:  //Func WideStringToLPOLESTR( const Source : Widestring) : POleStr
5621:  //Func XPath4Node( node : IHTMLElement) :Str
5622:  //Func TaskAllocWideString( const S :Str) : PWChar
5623:  Func AnsiIndexStr2( const AText :Str; const AValues : array of string) : Integer
5624:  Func GetPos( const SubSt, Text :Str; StartPos : Integer) : Integer
5625:  Func _CharPos( const C : Char; const S :Str) : Integer
5626:  Func CutString(var Text:str; const Delimiter:str; const Remove:Boolean) :Str
5627:  Proc FormatPath( Path :Str)
5628:  Func GetWinText( WinHandle : THandle) :Str
5629:  Func GetWinClass( Handle : Hwnd) : WideString
5630:  Func GetParentWinByClass( ChildHandle : HWND; const ClassName :Str) : HWND
5631:  Func DirectoryExists2( const Directory :Str) :Bool
5632:  //CL.AddDelphiFunction('Func VarSupports(const V:Variant;const IID:TGUID; out Intf): Bool);
5633:  Func CharInSet3( C : Char; const CharSet : TSysCharSet) :Bool
5634:  Func AddBackSlash2( const S :Str) :Str
5635:  CL.AddConstantN('WM_SETWBFOCUS','LongWord').SetUInt( $0400 + $44);
5636: end;
5637:
5638: Proc SIRegister_EwbUrl(CL: TPSPascalCompiler);
5639: begin
5640:  'TEMP_SIZE_EWB','LongInt').SetInt( 1024);
5641:  'MAX_BUFFER_EWB','LongInt').SetInt( 256);
5642:  'WebDelim_EWB','String').SetString( '/');
5643:  'ProtocolDelim_EWB','String').SetString( '://');
5644:  'QueryDelim_EWB','String').SetString( '?');
5645:  'BookmarkDelim_EWB','String').SetString( '#');
5646:  'EqualDelim_EWB','String').SetString( '=');
5647:  'DriveDelim_EWB','String').SetString( ':');
5648:   CL.AddTypeS('TQueryOption_EWB', 'ULONG');
5649:   TOnError_EWB','Proc (Sender:TObject;ErrorCode:integer;ErrMessage:str);
5650:   SIRegister_TUrl(CL);
5651: end;
5652:
5653: Func IS_INTRESOURCE(lpszType: PChar): BOOL;  //as of windows.pas
5654: begin
5655:   Result:= ULONG_PTR(lpszType) shr 16 = 0;
5656: end;
5657:
5658: Func ResourceNameToString(lpszName: PChar):Str;
5659: Func ResourceTypeToString(lpszType: PChar):Str;
5660: Proc storeRCDATAResourcetofile(aresname, afilename:Str);
5661: Func InternalfindNextUrlToDownload(aNode: TPageNotYetDownloadedBinTreeNode; '+
5662:          alowDeepLevel: Integer): TPageNotYetDownloadedBinTreeNode;'); //ALWebSpider
5663:
5664: Proc SIRegister_xrtl_math_Int(CL: TPSPascalCompiler);
5665: begin
5666:   AddTypeS('TXRTLInt', 'array of Int
5667:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5668:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5669:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5670:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5671:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5672:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5673:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5674:   'BitsPerByte','LongInt'( 8);
5675:   BitsPerDigit','LongInt'( 32);
5676:   SignBitMask','LongWord( $80000000);
5677:  Func XRTLAdjustBits( const ABits : Int) : Int
5678:  Func XRTLLength( const AInt : TXRTLInt) : Int
5679:  Func XRTLDataBits( const AInt : TXRTLInt) : Int
5680:  Proc XRTLBitPosition( const BitIndex : Int; var Index, Mask : Int)
5681:  Proc XRTLBitSet( var AInt : TXRTLInt; const BitIndex : Int)
5682:  Proc XRTLBitReset( var AInt : TXRTLInt; const BitIndex : Int)
5683:  Func XRTLBitGet( const AInt : TXRTLInt; const BitIndex : Int) : Int
5684:  Func XRTLBitGetBool( const AInt : TXRTLInt; const BitIndex : Int) :Bool
5685:  Func XRTLExtend(const AInt:TXRTLInt;ADataBits:Int;Sign:Int;var AResult:TXRTLInt):Int;
5686:  Func XRTLZeroExtend(const AInt:TXRTLInt;ADataBits:Int; var AResult:TXRTLInt):Int;
5687:  Func XRTLSignExtend(const AInt:TXRTLInt; ADataBits:Int;var AResult:TXRTLInt):Int;
5688:  Func XRTLSignStrip(const AInt:TXRTLInt;var AResult:TXRTLInt;const AMinDataBits:Int):Int;
5689:  Proc XRTLNot( const AInt : TXRTLInt; var AResult : TXRTLInt)
5690:  Proc XRTLOr( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5691:  Proc XRTLAnd( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5692:  Proc XRTLXor( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5693:  Func XRTLSign( const AInt : TXRTLInt) : Int
5694:  Proc XRTLZero( var AInt : TXRTLInt)
```

```
5695:  Proc XRTLOne( var AInt : TXRTLInt)
5696:  Proc XRTLMOne( var AInt : TXRTLInt)
5697:  Proc XRTLTwo( var AInt : TXRTLInt)
5698:  Func XRTLNeg( const AInt : TXRTLInt; var AResult : TXRTLInt) : Int
5699:  Func XRTLAbs( const AInt : TXRTLInt; var AResult : TXRTLInt) : Int
5700:  Proc XRTLFullSum( const A, B, C : Int; var Sum, Carry : Int)
5701:  Func XRTLAdd( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int;
5702:  Func XRTLAdd1(const AInt1:TXRTLInt;const AInt2:Int64;var AResult:TXRTLInt):Int;
5703:  Func XRTLSub( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int;
5704:  Func XRTLSub1(const AInt1:TXRTLInt;const AInt2:Int64;var AResult:TXRTLInt):Int;
5705:  Func XRTLCompare( const AInt1, AInt2 : TXRTLInt) : Int;
5706:  Func XRTLCompare1( const AInt1 : TXRTLInt; const AInt2 : Int64) : Int;
5707:  Func XRTLUMul( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int
5708:  Func XRTLMulAdd(const AInt1,AInt2,AInt3:TXRTLInt; var AResult:TXRTLInt):Int
5709:  Func XRTLMul( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int
5710:  Proc XRTLDivMod( const AInt1, AInt2 : TXRTLInt; var QResult, RResult : TXRTLInt)
5711:  Proc XRTLSqr( const AInt : TXRTLInt; var AResult : TXRTLInt)
5712:  Proc XRTLSqrt( const AInt : TXRTLInt; var AResult : TXRTLInt)
5713:  Proc XRTLRoot( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5714:  Proc XRTLRootApprox(const AInt1,AInt2:TXRTLInt;var ALowApproxResult,AHighApproxResult:TXRTLInt)
5715:  Proc XRTLURootApprox(const AInt1,AInt2:TXRTLInt;var ALowApproxResult,AHighApproxResult:TXRTLInt);
5716:  Proc XRTLExp( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5717:  Proc XRTLExpMod( const AInt1, AInt2, AInt3 : TXRTLInt;var AResult: TXRTLInt)
5718:  Proc XRTLSLBL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5719:  Proc XRTLSABL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5720:  Proc XRTLRCBL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5721:  Proc XRTLSLDL(const AInt:TXRTLInt;const DigitCount:Int; var AResult:TXRTLInt)
5722:  Proc XRTLSADL(const AInt: TXRTLInt; const DigitCount:Int;var AResult: TXRTLInt)
5723:  Proc XRTLRCDL(const AInt:TXRTLInt; const DigitCount:Int; var AResult: TXRTLInt)
5724:  Proc XRTLSLBR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5725:  Proc XRTLSABR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5726:  Proc XRTLRCBR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5727:  Proc XRTLSLDR(const AInt:TXRTLInt; const DigitCount:Int; var AResult: TXRTLInt)
5728:  Proc XRTLSADR(const AInt: TXRTLInt; const DigitCount:Int;var AResult: TXRTLInt)
5729:  Proc XRTLRCDR(const AInt: TXRTLInt;const DigitCount:Int;var AResult: TXRTLInt)
5730:  Func XRTLToHex( const AInt : TXRTLInt; Digits : Int) :Str
5731:  Func XRTLToBin( const AInt : TXRTLInt; Digits : Int) :Str
5732:  Func XRTLToString(const AInt : TXRTLInt; Radix : Int; Digits : Int) :Str
5733:  Proc XRTLFromHex(const Value :Str; var AResult : TXRTLInt)
5734:  Proc XRTLFromBin(const Value :Str; var AResult : TXRTLInt)
5735:  Proc XRTLFromString(const Value :Str; var AResult : TXRTLInt; Radix : Int)
5736:  Proc XRTLAssign( const AInt : TXRTLInt; var AResult : TXRTLInt);
5737:  Proc XRTLAssign1(const Value : Int; var AResult : TXRTLInt);
5738:  Proc XRTLAssign2(const Value : Int64; var AResult : TXRTLInt);
5739:  Proc XRTLAppend( const ALow, AHigh : TXRTLInt; var AResult : TXRTLInt)
5740:  Proc XRTLSplit(const AInt: TXRTLInt; var ALow,AHigh: TXRTLInt;LowDigits: Int)
5741:  Func XRTLGetMSBitIndex( const AInt : TXRTLInt) : Int
5742:  Proc XRTLMinMax(const AInt1, AInt2:TXRTLInt;var AMinResult,AMaxResult: TXRTLInt)
5743:  Proc XRTLMin( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt);
5744:  Proc XRTLMin1(const AInt1: TXRTLInt;const AInt2:Int;var AResult : TXRTLInt);
5745:  Proc XRTLMax( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt);
5746:  Proc XRTLMax1(const AInt1:TXRTLInt; const AInt2:Int; var AResult:TXRTLInt);
5747:  Proc XRTLGCD( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5748:  Proc XRTLSwap( var AInt1, AInt2 : TXRTLInt)
5749:  Proc XRTLFactorial( const AInt : TXRTLInt; var AResult : TXRTLInt)
5750:  Proc XRTLFactorialMod( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5751: end;
5752:
5753: Proc SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5754: begin
5755:  Func JvXPMethodsEqual( const Method1, Method2 : TMethod) :Bool
5756:  Proc JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Int)
5757:  Proc JvXPCreateGradientRect( const AWidth, AHeight : Int; const StartColor, EndColor : TColor;const
      Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Bool;var Bitmap:TBitmap);
5758:  Proc JvXPAdjustBoundRect(const BorderWidth:Byte;const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect: TRect)
5759:  Proc JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const AColor:TColor;Rect:TRect;
5760:  Proc JvXPConvertToGray2( Bitmap : TBitmap)
5761:  Proc JvXPRenderText(const AParent:TControl;const ACanvas:TCanvas;ACaption:TCaption;const
      AFont:TFont;const AEnabled,AShowAccelChar:Bool;var ARect:TRect;AFlags:Int)
5762:  Proc JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const Swapped:Bool;
5763:  Proc JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5764:  Proc JvXPSetDrawFlags(const AAlignment:TAlignment;const AWordWrap:Bool;var Flags: Int)
5765:  Proc 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)
5766: end;
5767:
5768: Proc SIRegister_uwinstr(CL: TPSPascalCompiler);
5769: begin
5770:  Func StrDec(S :Str) :Str
5771:  Func uIsNumeric( var S :Str; var X : Float) :Bool
5772:  Func ReadNumFromEdit( Edit : TEdit) : Float
5773:  Proc WriteNumToFile( var F : Text; X : Float)
5774: end;
5775:
5776: Proc SIRegister_utexplot(CL: TPSPascalCompiler);
5777: begin
5778:  Func TeX_InitGraphics(FileName:str;PgWidth,PgHeight:Int; Header:Boolean):Boolean
```

```
5779:  Proc TeX_SetWindow( X1, X2, Y1, Y2 : Int; GraphBorder :Bool)
5780:  Proc TeX_LeaveGraphics( Footer :Bool)
5781:  Proc TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5782:  Proc TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5783:  Proc TeX_SetGraphTitle( Title :Str)
5784:  Proc TeX_SetOxTitle( Title :Str)
5785:  Proc TeX_SetOyTitle( Title :Str)
5786:  Proc TeX_PlotOxAxis
5787:  Proc TeX_PlotOyAxis
5788:  Proc TeX_PlotGrid( Grid : TGrid)
5789:  Proc TeX_WriteGraphTitle
5790:  Func TeX_SetMaxCurv( NCurv : Byte) :Bool
5791:  Proc TeX_SetPointParam( CurvIndex, Symbol, Size : Int)
5792:  Proc TeX_SetLineParam( CurvIndex, Style : Int; Width : Float; Smooth :Bool)
5793:  Proc TeX_SetCurvLegend( CurvIndex : Int; Legend :Str)
5794:  Proc TeX_SetCurvStep( CurvIndex, Step : Int)
5795:  Proc TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Int)
5796:  Proc TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Int)
5797:  Proc TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Int; CurvIndex : Int)
5798:  Proc TeX_WriteLegend( NCurv : Int; ShowPoints, ShowLines :Bool)
5799:  Proc TeX_ConRec( Nx, Ny, Nc : Int; X, Y, Z : TVector; F : TMatrix)
5800:  Func Xcm(X : Float) : Float
5801:  Func Ycm(Y : Float) : Float
5802: end;
5803:
5804: *----------------------------------------------------------------------------*)
5805: Proc SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5806: begin
5807:   TConstArray', 'array of TVarRec
5808:  Func CopyVarRec( const Item : TVarRec) : TVarRec
5809:  Func CreateConstArray( const Elements : array of const) : TConstArray
5810:  Proc FinalizeVarRec( var Item : TVarRec)
5811:  Proc FinalizeConstArray( var Arr : TConstArray)
5812: end;
5813:
5814: Proc SIRegister_StStrS(CL: TPSPascalCompiler);
5815: begin
5816:  Func HexBS( B : Byte) : ShortString
5817:  Func HexWS( W : Word) : ShortString
5818:  Func HexLS( L : LongInt) : ShortString
5819:  Func HexPtrS( P : Pointer) : ShortString
5820:  Func BinaryBS( B : Byte) : ShortString
5821:  Func BinaryWS( W : Word) : ShortString
5822:  Func BinaryLS( L : LongInt) : ShortString
5823:  Func OctalBS( B : Byte) : ShortString
5824:  Func OctalWS( W : Word) : ShortString
5825:  Func OctalLS( L : LongInt) : ShortString
5826:  Func Str2Int16S( const S : ShortString; var I : SmallInt) :Bool
5827:  Func Str2WordS(const S: ShortString; var I : Word) :Bool
5828:  Func Str2LongS(const S: ShortString; var I : LongInt) :Bool
5829:  Func Str2RealS(const S: ShortString; var R : Double) :Bool
5830:  Func Str2RealS(const S: ShortString; var R : Real) :Bool
5831:  Func Str2ExtS(const S : ShortString; var R : Extended) :Bool
5832:  Func Long2StrS( L : LongInt) : ShortString
5833:  Func Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5834:  Func Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5835:  Func ValPrepS( const S : ShortString) : ShortString
5836:  Func CharStrS( C : AnsiChar; Len :Card) : ShortString
5837:  Func PadChS( const S : ShortString; C : AnsiChar; Len :Card):ShortString
5838:  Func PadS( const S : ShortString; Len :Card) : ShortString
5839:  Func LeftPadChS( const S : ShortString; C : AnsiChar; Len :Card):ShortString
5840:  Func LeftPadS( const S : ShortString; Len :Card) : ShortString
5841:  Func TrimLeadS( const S : ShortString) : ShortString
5842:  Func TrimTrailS( const S : ShortString) : ShortString
5843:  Func TrimS( const S : ShortString) : ShortString
5844:  Func TrimSpacesS( const S : ShortString) : ShortString
5845:  Func CenterChS( const S : ShortString; C : AnsiChar; Len :Card): ShortString
5846:  Func CenterS( const S : ShortString; Len :Card) : ShortString
5847:  Func EntabS( const S : ShortString; TabSize : Byte) : ShortString
5848:  Func DetabS( const S : ShortString; TabSize : Byte) : ShortString
5849:  Func ScrambleS( const S, Key : ShortString) : ShortString
5850:  Func SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5851:  Func FilterS( const S, Filters : ShortString) : ShortString
5852:  Func CharExistsS( const S : ShortString; C : AnsiChar) :Bool
5853:  Func CharCountS( const S : ShortString; C : AnsiChar) : Byte
5854:  Func WordCountS( const S, WordDelims : ShortString) :Card
5855:  Func WordPositionS(N:Card;const S, WordDelims:ShortString;var Pos:Card):Bool
5856:  Func ExtractWordS( N :Card; const S, WordDelims : ShortString) : ShortString
5857:  Func AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) :Card
5858:  Func AsciiPositionS(N:Card;const S,WordDelims:ShortStr;Quote:AnsiChar;var Pos:Card):Bool
5859:  Func ExtractAsciiS(N:Card;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5860:  Proc WordWrapS(const InSt:ShortString;var OutSt,Overlap:ShortString; Margin:Card;PadToMargin:Bool)
5861:  Func CompStringS( const S1, S2 : ShortString) : Int
5862:  Func CompUCStringS( const S1, S2 : ShortString) : Int
5863:  Func SoundexS( const S : ShortString) : ShortString
5864:  Func MakeLetterSetS( const S : ShortString) : Longint
5865:  Proc BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5866:  Func BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var Pos:Card):Bool;
5867:  Func BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortStr;var Pos:Card):Bool;
```

```
5868:  Func DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5869:  Func ForceExtensionS( const Name, Ext : ShortString) : ShortString
5870:  Func JustFilenameS( const PathName : ShortString) : ShortString
5871:  Func JustNameS( const PathName : ShortString) : ShortString
5872:  Func JustExtensionS( const Name : ShortString) : ShortString
5873:  Func JustPathnameS( const PathName : ShortString) : ShortString
5874:  Func AddBackSlashS( const DirName : ShortString) : ShortString
5875:  Func CleanPathNameS( const PathName : ShortString) : ShortString
5876:  Func HasExtensionS( const Name : ShortString; var DotPos :Card):Bool
5877:  Func CommaizeS( L : LongInt) : ShortString
5878:  Func CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5879:  Func FloatFormS(const Mask:ShortStr;R:TstFloat;const LtCurr,RtCurr:SString;Sep,DecPt:Char):ShortStr;
5880:  Func LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5881:  Func StrChPosS( const P : ShortString; C : AnsiChar; var Pos :Card) :Bool
5882:  Func StrStPosS( const P, S : ShortString; var Pos :Card) :Bool
5883:  Func StrStCopyS( const S : ShortString; Pos, Count :Card) : ShortString
5884:  Func StrChInsertS(const S : ShortString; C : AnsiChar; Pos :Card):ShortString
5885:  Func StrStInsertS(const S1, S2 : ShortString; Pos :Card) : ShortString
5886:  Func StrChDeleteS(const S: ShortString; Pos :Card) : ShortString
5887:  Func StrStDeleteS(const S: ShortString; Pos, Count :Card) : ShortString
5888:  Func ContainsOnlyS(const S, Chars : ShortString; var BadPos :Card) : Bool
5889:  Func ContainsOtherThanS( const S, Chars : ShortString; var BadPos :Card): Bool
5890:  Func CopyLeftS( const S : ShortString; Len :Card) : ShortString
5891:  Func CopyMidS( const S : ShortString; First, Len :Card) : ShortString
5892:  Func CopyRightS( const S : ShortString; First :Card) : ShortString
5893:  Func CopyRightAbsS( const S : ShortString; NumChars :Card) : ShortString
5894:  Func CopyFromNthWordS(const S,WordDelims:str;const AWord:str;N:Card;var SubString:ShortString):Bool;
5895:  Func DeleteFromNthWordS(const S,WordDelims:str;AWord:ShortString;N:Card;var SubStr:ShortString):Bool;
5896:  Func CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5897:  Func DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5898:  Func CopyWithinS( const S, Delimiter : ShortString; Strip :Bool) : ShortString
5899:  Func DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5900:  Func ExtractTokensS(const S,Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Card
5901:  Func IsChAlphaS( C : Char) :Bool
5902:  Func IsChNumericS( C : Char; const Numbers : ShortString) :Bool
5903:  Func IsChAlphaNumericS( C : Char; const Numbers : ShortString) :Bool
5904:  Func IsStrAlphaS( const S : Shortstring) :Bool
5905:  Func IsStrNumericS( const S, Numbers : ShortString) :Bool
5906:  Func IsStrAlphaNumericS( const S, Numbers : ShortString) :Bool
5907:  Func LastWordS(const S,WordDelims, AWord:ShortString; var Position :Card) :Bool
5908:  Func LastWordAbsS( const S, WordDelims : ShortString; var Position :Card) :Bool
5909:  Func LastStringS( const S, AString : ShortString; var Position :Card) :Bool
5910:  Func LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5911:  Func KeepCharsS( const S, Chars : ShortString) : ShortString
5912:  Func RepeatStringS(const RepeatString:ShortStr;var Repetitions:Card;MaxLen:Card):ShortStr;
5913:  Func ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Card;var Replacements:Card):ShortString;
5914:  Func ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Card):ShortString;
5915:  Func ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Card;var Replacements:Card):ShortString
5916:  Func ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var Replacements:Card):ShortString
5917:  Func RightTrimCharsS( const S, Chars : ShortString) : ShortString
5918:  Func StrWithinS(const S,SearchStr: ShortString;Start:Card;var Position:Card):bool
5919:  Func TrimCharsS( const S, Chars : ShortString) : ShortString
5920:  Func WordPosS(const S,WordDelims,AWord:ShortString;N:Card;var Position:Card):Bool
5921: end;
5922:
5923: ********unit uPSI_StUtils; from Systools4****************************************
5924: Func SignL( L : LongInt) : Int
5925: Func SignF( F : Extended) : Int
5926: Func MinWord(A, B : Word) : Word
5927: Func MidWord(W1, W2, W3 : Word) : Word
5928: Func MaxWord(A, B : Word) : Word
5929: Func MinLong(A, B : LongInt) : LongInt
5930: Func MidLong(L1, L2, L3 : LongInt) : LongInt
5931: Func MaxLong(A, B : LongInt) : LongInt
5932: Func MinFloat(F1, F2 : Extended) : Extended
5933: Func MidFloat(F1, F2, F3 : Extended) : Extended
5934: Func MaxFloat(F1, F2 : Extended) : Extended
5935: Func MakeInt16( H, L : Byte) : SmallInt
5936: Func MakeWordS( H, L : Byte) : Word
5937: Func SwapNibble( B : Byte) : Byte
5938: Func SwapWord( L : LongInt) : LongInt
5939: Proc SetFlag( var Flags : Word; FlagMask : Word)
5940: Proc ClearFlag( var Flags : Word; FlagMask : Word)
5941: Func FlagIsSet( Flags, FlagMask : Word) :Bool
5942: Proc SetByteFlag( var Flags : Byte; FlagMask : Byte)
5943: Proc ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5944: Func ByteFlagIsSet( Flags, FlagMask : Byte) :Bool
5945: Proc SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5946: Proc ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5947: Func LongFlagIsSet( Flags, FlagMask : LongInt) :Bool
5948: Proc ExchangeBytes( var I, J : Byte)
5949: Proc ExchangeWords( var I, J : Word)
5950: Proc ExchangeLongInts( var I, J : LongInt)
5951: Proc ExchangeStructs( var I, J, Size :Card)
5952: Proc FillWord( var Dest, Count :Card; Filler : Word)
5953: Proc FillStruct( var Dest, Count :Card; var Filler, FillerSize :Card)
```

```
5954: Func AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5955: //*****************uPSI_StFIN;***********************************************
5956: Func AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis:TStBasis): Extended
5957: Func AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,Par:Extended;Frequency:TStFrequency;
      Basis : TStBasis): Extended
5958: Func BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5959: Func BondPrice(Settlement,Maturity:TStDate;Rate,Yield,
      Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):Extended
5960: Func CumulativeInterest(Rate:Extended;NPeriods:Int;PV:Extended;StartPeriod,
      EndPeriod:Int;Frequency:TStFrequency; Timing:TStPaymentTime):Extended
5961: Func CumulativePrincipal(Rate: Extended;NPeriods:Int;PV:Extended;StartPeriod,
      EndPeriod:Int;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
5962: Func DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5963: Func DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Int) : Extended
5964: Func DiscountRate(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis): Extended;
5965: Func DollarToDecimal( FracDollar : Extended; Fraction : Int) : Extended
5966: Func DollarToDecimalText( DecDollar : Extended) :Str
5967: Func DollarToFraction( DecDollar : Extended; Fraction : Int) : Extended
5968: Func DollarToFractionStr( FracDollar : Extended; Fraction : Int) :Str
5969: Func EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5970: Func FutureValueS(Rate:Extended;NPeriods:Int;Pmt,PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5971: Func FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5972: Func FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Int) : Extended
5973: Func InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5974: Func InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5975: Func InternalRateOfReturn16( const Values, NValues : Int; Guess : Extended) : Extended
5976: Func IsCardValid( const S :Str) :Bool
5977: Func ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extend;Freq:TStFrequency;Basis:TStBasis):Extended;
5978: Func ModifiedIRR(const Values:array of Double;FinanceRate,ReinvestRate:Extended): Extended
5979: Func ModifiedIRR16(const Values,NValues:Int; FinanceRate,ReinvestRate: Extended): Extended
5980: Func NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5981: Func NetPresentValue16( Rate : Extended; const Values, NValues : Int) : Extended
5982: Func NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5983: Func NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5984: Func NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5985: Func Payment(Rate:Extended;NPeriods:Int;PV
      FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
5986: Func Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Int;
5987: Func PresentValueS(Rate:Extended;NPeriods:Int;Pmt,
      FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
5988: Func ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5989: Func RoundToDecimal( Value : Extended; Places : Int; Bankers :Bool) : Extended
5990: Func TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5991: Func TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5992: Func TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5993: Func VariableDecliningBalance(Cost,Salvage:Extended;Life:Int;StartPeriod,EndPeriod,
      Factor:Extended;NoSwitch:bool):Extended
5994: Func YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Exted;Basis:TStBasis):Extended;
5995: Func YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5996: Func YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5997:
5998: //*********************************************unit uPSI_StAstroP;
5999: Proc PlanetsPos( JD : Double; var PA : TStPlanetsArray)
6000: //*****unit unit uPSI_StStat; Statistic Package of SysTools*******************
6001: Func AveDev( const Data : array of Double) : Double
6002: Func AveDev16( const Data, NData : Int) : Double
6003: Func Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
6004: Func Correlation( const Data1, Data2 : array of Double) : Double
6005: Func Correlation16( const Data1, Data2, NData : Int) : Double
6006: Func Covariance( const Data1, Data2 : array of Double) : Double
6007: Func Covariance16( const Data1, Data2, NData : Int) : Double
6008: Func DevSq( const Data : array of Double) : Double
6009: Func DevSq16( const Data, NData : Int) : Double
6010: Proc Frequency(const Data:array of Double;const Bins:array of Double;var Counts:array of LongInt);
6011: //Proc Frequency16( const Data, NData : Int; const Bins, NBins : Int; var Counts)
6012: Func GeometricMeanS( const Data : array of Double) : Double
6013: Func GeometricMean16( const Data, NData : Int) : Double
6014: Func HarmonicMeanS( const Data : array of Double) : Double
6015: Func HarmonicMean16( const Data, NData : Int) : Double
6016: Func Largest( const Data : array of Double; K : Int) : Double
6017: Func Largest16( const Data, NData : Int; K : Int) : Double
6018: Func MedianS( const Data : array of Double) : Double
6019: Func Median16( const Data, NData : Int) : Double
6020: Func Mode( const Data : array of Double) : Double
6021: Func Mode16( const Data, NData : Int) : Double
6022: Func Percentile( const Data : array of Double; K : Double) : Double
6023: Func Percentile16( const Data, NData : Int; K : Double) : Double
6024: Func PercentRank( const Data : array of Double; X : Double) : Double
6025: Func PercentRank16( const Data, NData : Int; X : Double) : Double
6026: Func Permutations( Number, NumberChosen : Int) : Extended
6027: Func Combinations( Number, NumberChosen : Int) : Extended
6028: Func FactorialS( N : Int) : Extended
6029: Func Rank( Number : Double; const Data : array of Double; Ascending: Bool) : Int
6030: Func Rank16( Number : Double; const Data, NData : Int; Ascending : Bool) : Int
6031: Func Smallest( const Data : array of Double; K : Int) : Double
```

```
6032: Func Smallest16( const Data, NData : Int; K : Int) : Double
6033: Func TrimMean( const Data : array of Double; Percent : Double) : Double
6034: Func TrimMean16( const Data, NData : Int; Percent : Double) : Double
6035:   AddTypeS('TStLinEst', 'record B0:Double; B1 : double; seB0 : double; seB'
6036:    +1 : Double; R2: Double;sigma :Double;SSr:double;SSe: Double; F0:Double;df:Int;end
6037: Proc LinEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TStLinEst;ErrorStats:Bool;
6038: Proc LogEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TStLinEst;ErrorStats:Bool;
6039: Func Forecast(X: Double;const KnownY: array of Double;const KnownX:array of Double):Double
6040: Func ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double
6041: Func Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
6042: Func RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
6043: Func Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
6044: Func StandardErrorY(const KnownY: array of Double;const KnownX:array of Double):Double
6045: Func BetaDist( X, Alpha, Beta, A, B : Single) : Single
6046: Func BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
6047: Func BinomDist(NumberS,Trials: Int; ProbabilityS : Single;Cumulative:Bool):Single
6048: Func CritBinom( Trials : Int; ProbabilityS, Alpha : Single) : Int
6049: Func ChiDist( X : Single; DegreesFreedom : Int) : Single
6050: Func ChiInv( Probability : Single; DegreesFreedom : Int) : Single
6051: Func ExponDist( X, Lambda : Single; Cumulative :Bool) : Single
6052: Func FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Int) : Single
6053: Func FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Int) : Single
6054: Func LogNormDist( X, Mean, StandardDev : Single) : Single
6055: Func LogInv( Probability, Mean, StandardDev : Single) : Single
6056: Func NormDist( X, Mean, StandardDev : Single; Cumulative :Bool) : Single
6057: Func NormInv( Probability, Mean, StandardDev : Single) : Single
6058: Func NormSDist( Z : Single) : Single
6059: Func NormSInv( Probability : Single) : Single
6060: Func Poisson( X : Int; Mean : Single; Cumulative :Bool) : Single
6061: Func TDist( X : Single; DegreesFreedom : Int; TwoTails :Bool) : Single
6062: Func TInv( Probability : Single; DegreesFreedom : Int) : Single
6063: Func Erfc( X : Single) : Single
6064: Func GammaLn( X : Single) : Single
6065: Func LargestSort( const Data : array of Double; K : Int) : Double
6066: Func SmallestSort( const Data: array of double; K : Int) : Double
6067:
6068: Proc SIRegister_TStSorter(CL: TPSPascalCompiler);
6069:  Func OptimumHeapToUse( RecLen :Card; NumRecs : LongInt) : LongInt
6070:  Func MinimumHeapToUse( RecLen :Card) : LongInt
6071:  Func MergeInfo( MaxHeap : LongInt; RecLen :Card; NumRecs : LongInt) : TMergeInfo
6072:  Func DefaultMergeName( MergeNum : Int) :Str
6073:  Proc ArraySort( var A, RecLen, NumRecs :Card; Compare:TUntypedCompareFunc)
6074: Proc SIRegister_StAstro(CL: TPSPascalCompiler);
6075: Func AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
6076:  Func FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
6077:  Func SunPos( UT : TStDateTimeRec) : TStPosRec
6078:  Func SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
6079:  Func SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
6080:  Func Twilight(LD:TStDate; Longitude,Latitude:Double;TwiType:TStTwilight):TStRiseSetRec
6081:  Func LunarPhase( UT : TStDateTimeRec) : Double
6082:  Func MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
6083:  Func MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
6084:  Func FirstQuarter( D : TStDate) : TStLunarRecord
6085:  Func FullMoon( D : TStDate) : TStLunarRecord
6086:  Func LastQuarter( D : TStDate) : TStLunarRecord
6087:  Func NewMoon( D : TStDate) : TStLunarRecord
6088:  Func NextFirstQuarter( D : TStDate) : TStDateTimeRec
6089:  Func NextFullMoon( D : TStDate) : TStDateTimeRec
6090:  Func NextLastQuarter( D : TStDate) : TStDateTimeRec
6091:  Func NextNewMoon( D : TStDate) : TStDateTimeRec
6092:  Func PrevFirstQuarter( D : TStDate) : TStDateTimeRec
6093:  Func PrevFullMoon( D : TStDate) : TStDateTimeRec
6094:  Func PrevLastQuarter( D : TStDate) : TStDateTimeRec
6095:  Func PrevNewMoon( D : TStDate) : TStDateTimeRec
6096:  Func SiderealTime( UT : TStDateTimeRec) : Double
6097:  Func Solstice( Y, Epoch : Int; Summer :Bool) : TStDateTimeRec
6098:  Func Equinox( Y, Epoch : Int; Vernal :Bool) : TStDateTimeRec
6099:  Func SEaster( Y, Epoch : Int) : TStDate
6100:  Func DateTimeToAJD( D : TDateTime) : Double
6101:  Func HoursMin( RA : Double) : ShortString
6102:  Func DegsMin( DC : Double) : ShortString
6103:  Func AJDToDateTime( D : Double) : TDateTime
6104:
6105: Proc SIRegister_StDate(CL: TPSPascalCompiler);
6106: Func CurrentDate : TStDate
6107: Func StValidDate( Day, Month, Year, Epoch : Int) :Bool
6108: Func DMYtoStDate( Day, Month, Year, Epoch : Int) : TStDate
6109: Proc StDateToDMY( Julian : TStDate; var Day, Month, Year : Int)
6110: Func StIncDate( Julian : TStDate; Days, Months, Years : Int) : TStDate
6111: Func IncDateTrunc( Julian : TStDate; Months, Years : Int) : TStDate
6112: Proc StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Int)
6113: Func BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
6114: Func WeekOfYear( Julian : TStDate) : Byte
6115: Func AstJulianDate( Julian : TStDate) : Double
6116: Func AstJulianDatetoStDate( AstJulian : Double; Truncate :Bool) : TStDate
6117: Func AstJulianDatePrim( Year, Month, Date : Int; UT : TStTime) : Double
6118: Func StDayOfWeek( Julian : TStDate) : TStDayType
6119: Func DayOfWeekDMY( Day, Month, Year, Epoch : Int) : TStDayType
6120: Func StIsLeapYear( Year : Int) :Bool
```

```
6121: Func StDaysInMonth( Month : Int; Year, Epoch : Int) : Int
6122: Func ResolveEpoch( Year, Epoch : Int) : Int
6123: Func ValidTime( Hours, Minutes, Seconds : Int) :Bool
6124: Proc StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
6125: Func HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
6126: Func CurrentTime : TStTime
6127: Proc TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
6128: Func StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
6129: Func DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
6130: Func RoundToNearestHour( T : TStTime; Truncate :Bool) : TStTime
6131: Func RoundToNearestMinute( const T : TStTime; Truncate :Bool) : TStTime
6132: Proc DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
6133: Proc IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Int;Secs:LongInt)
6134: Func DateTimeToStDate(DT : TDateTime) : TStDate
6135: Func DateTimeToStTime(DT : TDateTime) : TStTime
6136: Func StDateToDateTime(D : TStDate) : TDateTime
6137: Func StTimeToDateTime(T : TStTime) : TDateTime
6138: Func Convert2ByteDate(TwoByteDate : Word) : TStDate
6139: Func Convert4ByteDate(FourByteDate : TStDate) : Word
6140:
6141: Proc SIRegister_StDateSt(CL: TPSPascalCompiler);
6142: Func DateStringHMStoAstJD( const Picture, DS :Str; H, M, S, Epoch : Int) : Double
6143: Func MonthToString( const Month : Int) :Str
6144: Func DateStringToStDate( const Picture, S :Str; Epoch : Int) : TStDate
6145: Func DateStringToDMY(const Picture,S:str; Epoch:Int; var D, M, Y : Int):Boolean
6146: Func StDateToDateString( const Picture:str;const Julian:TStDate;Pack:Bool):str
6147: Func DayOfWeekToString( const WeekDay : TStDayType) :Str
6148: Func DMYtoDateString(const Picture:str;Day,Month,Year,Epoch:Int;Pack:Boolean):str);
6149: Func CurrentDateString( const Picture :Str; Pack :Bool) :Str
6150: Func CurrentTimeString( const Picture :Str; Pack :Bool) :Str
6151: Func TimeStringToHMS( const Picture, St :Str; var H, M, S : Int) :Bool
6152: Func TimeStringToStTime( const Picture, S :Str) : TStTime
6153: Func StTimeToAmPmString(const Picture:str; const T:TStTime; Pack:Bool) :Str
6154: Func StTimeToTimeString(const Picture:Str; const T:TStTime;Pack :Bool) :Str
6155: Func DateStringIsBlank( const Picture, S :Str) :Bool
6156: Func InternationalDate( ForceCentury :Bool) :Str
6157: Func InternationalLongDate( ShortNames :Bool; ExcludeDOW :Bool) :Str
6158: Func InternationalTime( ShowSeconds :Bool) :Str
6159: Proc ResetInternationalInfo
6160:
6161: Proc SIRegister_StBase(CL: TPSPascalCompiler);
6162:  Func DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) :Bool
6163:  Func AnsiUpperCaseShort32( const S :Str) :Str
6164:  Func AnsiCompareTextShort32( const S1, S2 :Str) : Int
6165:  Func AnsiCompareStrShort32( const S1, S2 :Str) : Int
6166:  Func HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
6167:  Func HugeDecompressRLE(const InBuffer,InLen:Longint;var OutBuffer,OutLen:LongInt):Longint
6168:  Proc HugeFillChar( var Dest, Count : Longint; Value : Byte)
6169:  Proc HugeFillStruc( var Dest, Count : Longint; const Value, ValSize :Card)
6170:  Func Upcase( C : AnsiChar) : AnsiChar
6171:  Func LoCase( C : AnsiChar) : AnsiChar
6172:  Func CompareLetterSets( Set1, Set2 : LongInt) :Card
6173:  Func CompStruct( const S1, S2, Size :Card) : Int
6174:  Func Search(const Buffer,BufLength:Card;const Match,MatLength:Card;var Pos:Cardi):Bool;
6175:  Func StSearch(const Buff,BufLength:Card;const Match,MatLength:Card;var Pos:Cardi):Bool
6176:  Func SearchUC(const Buff,BufLength:Card;const Match,MatLength:Card;var Pos:Cardi):Bool
6177:  Func IsOrInheritsFrom( Root, Candidate : TClass) :Bool
6178:  Proc RaiseContainerError( Code : longint)
6179:  Proc RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6180:  Func ProductOverflow( A, B : LongInt) :Bool
6181:  Func StNewStr( S :Str) : PShortString
6182:  Proc StDisposeStr( PS : PShortString)
6183:  Proc ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : Int)
6184:  Proc ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : Int)
6185:  Proc ValWord( const S : ShortString; var Wd : word; var ErrorCode : Int)
6186:  Proc RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6187:  Proc RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6188:  Proc RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info :Str)
6189:
6190: Proc SIRegister_usvd(CL: TPSPascalCompiler);
6191: begin
6192: Proc SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Int; S : TVector; V : TMatrix)
6193: Proc SV_SetZero( S : TVector; Lb, Ub : Int; Tol : Float)
6194: Proc SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Int;X:TVector);
6195: Proc SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Int; A : TMatrix)
6196: Proc RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6197: end;
6198:
6199: //**********unit unit ; StMath Package of SysTools*****************************
6200: Func IntPowerS( Base : Extended; Exponent : Int) : Extended
6201: Func PowerS( Base, Exponent : Extended) : Extended
6202: Func StInvCos( X : Double) : Double
6203: Func StInvSin( Y : Double) : Double
6204: Func StInvTan2( X, Y : Double) : Double
6205: Func StTan( A : Double) : Double
6206: Proc DumpException;  //unit StExpEng;
6207: Func HexifyBlock( var Buffer, BufferSize : Int):Str
6208:
6209: //**********unit unit ; StCRC Package of SysTools*****************************
```

```
6210: Func Adler32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
6211: Func Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6212: Func Adler32OfFile( FileName : Ansistr) : LongInt
6213: Func Crc16Prim( var Data, DataSize, CurCrc :Card) :Card
6214: Func Crc16OfStream( Stream : TStream; CurCrc :Card) :Card
6215: Func Crc16OfFile( FileName : Ansistr) :Card
6216: Func Crc32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
6217: Func Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6218: Func Crc32OfFile( FileName : Ansistr) : LongInt
6219: Func InternetSumPrim( var Data, DataSize, CurCrc :Card) :Card
6220: Func InternetSumOfStream( Stream : TStream; CurCrc :Card) :Card
6221: Func InternetSumOfFile( FileName : Ansistr) :Card
6222: Func Kermit16Prim( var Data, DataSize, CurCrc :Card) :Card
6223: Func Kermit16OfStream( Stream : TStream; CurCrc :Card) :Card
6224: Func Kermit16OfFile( FileName : Ansistr) :Card
6225:
6226: //**********unit unit ; StBCD Package of SysTools*****************************************
6227: Func AddBcd( const B1, B2 : TbcdS) : TbcdS
6228: Func SubBcd( const B1, B2 : TbcdS) : TbcdS
6229: Func MulBcd( const B1, B2 : TbcdS) : TbcdS
6230: Func DivBcd( const B1, B2 : TbcdS) : TbcdS
6231: Func ModBcd( const B1, B2 : TbcdS) : TbcdS
6232: Func NegBcd( const B : TbcdS) : TbcdS
6233: Func AbsBcd( const B : TbcdS) : TbcdS
6234: Func FracBcd( const B : TbcdS) : TbcdS
6235: Func IntBcd( const B : TbcdS) : TbcdS
6236: Func RoundDigitsBcd( const B : TbcdS; Digits :Card): TbcdS
6237: Func RoundPlacesBcd( const B : TbcdS; Places :Card : TbcdS
6238: Func ValBcd( const S :Str) : TbcdS
6239: Func LongBcd( L : LongInt) : TbcdS
6240: Func ExtBcd( E : Extended) : TbcdS
6241: Func ExpBcd( const B : TbcdS) : TbcdS
6242: Func LnBcd( const B : TbcdS) : TbcdS
6243: Func IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6244: Func PowBcd( const B, E : TbcdS) : TbcdS
6245: Func SqrtBcd( const B : TbcdS) : TbcdS
6246: Func CmpBcd( const B1, B2 : TbcdS) : Int
6247: Func EqDigitsBcd( const B1, B2 : TbcdS; Digits :Card) :Bool
6248: Func EqPlacesBcd( const B1, B2 : TbcdS; Digits :Card) :Bool
6249: Func IsIntBcd( const B : TbcdS) :Bool
6250: Func TruncBcd( const B : TbcdS) : LongInt
6251: Func BcdExt( const B : TbcdS) : Extended
6252: Func RoundBcd( const B : TbcdS) : LongInt
6253: Func StrBcd( const B : TbcdS; Width, Places :Card) :Str
6254: Func StrExpBcd( const B : TbcdS; Width :Card) :Str
6255: Func FormatBcd( const Format :Str; const B : TbcdS) :Str
6256: Func StrGeneralBcd( const B : TbcdS) :Str
6257: Func FloatFormBcd(const Mask:str;B:TbcdS;const LtCurr,RtCurr:str;Sep,DecPt:AnsiChar):str
6258: Proc ConvertBcd( const SrcB, SrcSize: Byte; var DestB, DestSize : Byte)
6259:
6260: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6261: Proc StParseLine( const Data : Ansistr; Schema : TStTextDataSchema; Result : TStrings)
6262: Func StFieldTypeToStr( FieldType : TStSchemaFieldType) : Ansistr
6263: Func StStrToFieldType( const S : Ansistr) : TStSchemaFieldType
6264: Func StDeEscape( const EscStr : Ansistr) : Char
6265: Func StDoEscape( Delim : Char) : Ansistr
6266: Func StTrimTrailingChars( const S : Ansistr; Trailer : Char) : Ansistr
6267: Func AnsiHashText( const S :Str; Size : Int) : Int
6268: Func AnsiHashStr( const S :Str; Size : Int) : Int
6269: Func AnsiELFHashText( const S :Str; Size : Int) : Int
6270: Func AnsiELFHashStr( const S :Str; Size : Int) : Int
6271:
6272: //**********unit unit ; StNetCon Package of SysTools*****************************************
6273: with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6274: Constructor Create( AOwner : TComponent)
6275: Func Connect : DWord
6276: Func Disconnect : DWord
6277: RegisterProperty('Password', 'String', iptrw);
6278: Property(UserName', 'String', iptrw);
6279: Property(ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6280: Property(DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6281: Property(LocalDevice', 'String', iptrw);
6282: Property(ServerName', 'String', iptrw);
6283: Property(ShareName', 'String', iptrw);
6284: Property(OnConnect', 'TNotifyEvent', iptrw);
6285: Property(OnConnectFail', 'TOnConnectFailEvent', iptrw);
6286: Property(OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6287: Property(OnDisconnect', 'TNotifyEvent', iptrw);
6288: Property(OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6289: Property(OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6290: end;
6291: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6292: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6293: Proc InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6294: Proc EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6295: Proc LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6296: Func InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6297: Func SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
```

```
6298: Func TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6299: Proc DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6300: Func GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6301: Func SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6302: Func SuspendThread( hThread : THandle) : DWORD
6303: Func ResumeThread( hThread : THandle) : DWORD
6304: Func CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6305: Func GetCurrentThread : THandle
6306: Proc ExitThread( dwExitCode : DWORD)
6307: Func TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6308: Func GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6309: Proc EndThread(ExitCode: Int);
6310: Func WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6311: Func MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6312: Proc FreeProcInstance( Proc : FARPROC)
6313: Proc FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6314: Func DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6315: Proc ParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Pointer; ASafeSection:Bool);
6316: Proc ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection :Bool);
6317: Proc ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6318: Proc ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: bool;
6319: Func CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6320: Func CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection :Bool) : TParallelJob;
6321: Func CurrentParallelJobInfo : TParallelJobInfo
6322: Func ObtainParallelJobInfo : TParallelJobInfo
6323: Proc GetSystemInfo( var lpSystemInfo : TSystemInfo);
6324: Func IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL;
6325: Func SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL;
6326: Func
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL;
6327: Func SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL;
6328: Func DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL;dwOptions:DWORD):BOOL;
6329: Func GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL;
6330: Func SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL;
6331:
6332: ****************************************************unit uPSI_JclMime;
6333:  Func MimeEncodeString( const S : Ansistr) : Ansistr
6334:  Func MimeDecodeString( const S : Ansistr) : Ansistr
6335:  Proc MimeEncodeStream(const InputStream:TStream; const OutputStream : TStream)
6336:  Proc MimeDecodeStream(const InputStream:TStream; const OutputStream : TStream)
6337:  Func MimeEncodedSize( const I :Card) :Card
6338:  Func MimeDecodedSize( const I :Card) :Card
6339:  Proc MimeEncode(var InputBuffer:Str;const InputByteCount:Card;var OutputBuffer)
6340:  Func MimeDecode(var InputBuffer:str;const InputBytesCount:Card;var OutputBuffer):Cardi;
6341:  Func MimeDecodePartial(var InputBuffer:str;const InputBytesCount:Card;var OutputBuffer:str;var
      ByteBuffer:Card;var ByteBufferSpace:Card):Card
6342:  Func MimeDecodePartialEnd(var OutputBuf:str;const ByteBuf:Card;const ByteBufferSpace:Card):Cardi;
6343:
6344: ****************************************************unit uPSI_JclPrint;
6345:  Proc DirectPrint( const Printer, Data :Str)
6346:  Proc SetPrinterPixelsPerInch
6347:  Func GetPrinterResolution : TPoint
6348:  Func CharFitsWithinDots( const Text :Str; const Dots : Int) : Int
6349:  Proc PrintMemo( const Memo : TMemo; const Rect : TRect)
6350:
6351: //*******************************unit uPSI_ShLwApi;*************************************
6352:  Func StrChr( lpStart : PChar; wMatch : WORD) : PChar
6353:  Func StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6354:  Func StrCmpN( lpStr1, lpStr2 : PChar; nChar : Int) : Int
6355:  Func StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Int) : Int
6356:  Func StrCSpn( lpStr_, lpSet : PChar) : Int
6357:  Func StrCSpnI( lpStr1, lpSet : PChar) : Int
6358:  Func StrDup( lpSrch : PChar) : PChar
6359:  Func StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6360:  Func StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6361:  Func StrFromTimeInterval(pszOut:PChar cchMax:UINT;dwTimeMS:DWORD;digits:Int):Int
6362:  Func StrIsIntlEqual(fCaseSens: BOOL;lpString1,lpString2:PChar; nChar:Int): BOOL
6363:  Func StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
6364:  Func StrPBrk( psz, pszSet : PChar) : PChar
6365:  Func StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6366:  Func StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6367:  Func StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6368:  Func StrSpn( psz, pszSet : PChar) : Int
6369:  Func StrStr( lpFirst, lpSrch : PChar) : PChar
6370:  Func StrStrI( lpFirst, lpSrch : PChar) : PChar
6371:  Func StrToInt( lpSrch : PChar) : Int
6372:  Func StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Int) : BOOL
6373:  Func StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6374:  Func ChrCmpI( w1, w2 : WORD) : BOOL
6375:  Func ChrCmpIA( w1, w2 : WORD) : BOOL
6376:  Func ChrCmpIW( w1, w2 : WORD) : BOOL
6377:  Func StrIntlEqN( s1, s2 : PChar; nChar : Int) : BOOL
6378:  Func StrIntlEqNI( s1, s2 : PChar; nChar : Int) : BOOL
6379:  Func StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Int) : PChar
6380:  Func StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
6381:  Func IntlStrEqWorker(fCaseSens:BOOL;lpString1,lpString2:PChar;nChar:Int):BOOL
6382:  Func IntlStrEqN( s1, s2 : PChar; nChar : Int) : BOOL
```

```
SZ_CONTENTTYPE_HTMLA','String 'text/html
SZ_CONTENTTYPE_HTMLW','String 'text/html
SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
Func PathIsHTMLFile( pszPath : PChar) : BOOL
STIF_DEFAULT','LongWord( $00000000);
STIF_SUPPORT_HEX','LongWord( $00000001);
Func StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Int) : Int
Func StrNCpy( psz1, psz2 : PChar; cchMax : Int) : PChar
Func StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
Func PathAddBackslash( pszPath : PChar) : PChar
Func PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
Func PathAppend( pszPath : PChar; pMore : PChar) : BOOL
Func PathBuildRoot( szRoot : PChar; iDrive : Int) : PChar
Func PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
Func PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
Func PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
Func PathCompactPathEx(pszOut:PChar;pszSrc:PChar;cchMax:UINT;dwFlags:DWORD):BOOL
Func PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Int
Func PathFileExists( pszPath : PChar) : BOOL
Func PathFindExtension( pszPath : PChar) : PChar
Func PathFindFileName( pszPath : PChar) : PChar
Func PathFindNextComponent( pszPath : PChar) : PChar
Func PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
Func PathGetArgs( pszPath : PChar) : PChar
Func PathFindSuffixArray(pszPath:PChar;const apszSuffix:PChar;iArraySize:Int): PChar
Func PathIsLFNFileSpec( lpName : PChar) : BOOL
Func PathGetCharType( ch : Char) : UINT
GCT_INVALID','LongWord( $0000);
GCT_LFNCHAR','LongWord( $0001);
GCT_SHORTCHAR','LongWord( $0002);
GCT_WILD','LongWord( $0004);
GCT_SEPARATOR','LongWord( $0008);
Func PathGetDriveNumber( pszPath : PChar) : Int
Func PathIsDirectory( pszPath : PChar) : BOOL
Func PathIsDirectoryEmpty( pszPath : PChar) : BOOL
Func PathIsFileSpec( pszPath : PChar) : BOOL
Func PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
Func PathIsRelative( pszPath : PChar) : BOOL
Func PathIsRoot( pszPath : PChar) : BOOL
Func PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
Func PathIsUNC( pszPath : PChar) : BOOL
Func PathIsNetworkPath( pszPath : PChar) : BOOL
Func PathIsUNCServer( pszPath : PChar) : BOOL
Func PathIsUNCServerShare( pszPath : PChar) : BOOL
Func PathIsContentType( pszPath, pszContentType : PChar) : BOOL
Func PathIsURL( pszPath : PChar) : BOOL
Func PathMakePretty( pszPath : PChar) : BOOL
Func PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
Func PathParseIconLocation( pszIconFile : PChar) : Int
Proc PathQuoteSpaces( lpsz : PChar)
Func PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
Proc PathRemoveArgs( pszPath : PChar)
Func PathRemoveBackslash( pszPath : PChar) : PChar
Proc PathRemoveBlanks( pszPath : PChar)
Proc PathRemoveExtension( pszPath : PChar)
Func PathRemoveFileSpec( pszPath : PChar) : BOOL
Func PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
Func PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
Proc PathSetDlgItemPath( hDlg : HWND; id : Int; pszPath : PChar)
Func PathSkipRoot( pszPath : PChar) : PChar
Proc PathStripPath( pszPath : PChar)
Func PathStripToRoot( pszPath : PChar) : BOOL
Proc PathUnquoteSpaces( lpsz : PChar)
Func PathMakeSystemFolder( pszPath : PChar) : BOOL
Func PathUnmakeSystemFolder( pszPath : PChar) : BOOL
Func PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
Proc PathUndecorate( pszPath : PChar)
Func PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
URL_SCHEME_INVALID','LongInt'( - 1);
URL_SCHEME_UNKNOWN','LongInt'( 0);
URL_SCHEME_FTP','LongInt'( 1);
URL_SCHEME_HTTP','LongInt'( 2);
URL_SCHEME_GOPHER','LongInt'( 3);
URL_SCHEME_MAILTO','LongInt'( 4);
URL_SCHEME_NEWS','LongInt'( 5);
URL_SCHEME_NNTP','LongInt'( 6);
URL_SCHEME_TELNET','LongInt'( 7);
URL_SCHEME_WAIS','LongInt'( 8);
URL_SCHEME_FILE','LongInt'( 9);
URL_SCHEME_MK','LongInt'( 10);
URL_SCHEME_HTTPS','LongInt'( 11);
URL_SCHEME_SHELL','LongInt'( 12);
URL_SCHEME_SNEWS','LongInt'( 13);
URL_SCHEME_LOCAL','LongInt'( 14);
URL_SCHEME_JAVASCRIPT','LongInt'( 15);
URL_SCHEME_VBSCRIPT','LongInt'( 16);
```

```
6472:  URL_SCHEME_ABOUT','LongInt'( 17);
6473:  URL_SCHEME_RES','LongInt'( 18);
6474:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6475:  URL_SCHEME', 'Int
6476:  URL_PART_NONE','LongInt'( 0);
6477:  URL_PART_SCHEME','LongInt'( 1);
6478:  URL_PART_HOSTNAME','LongInt'( 2);
6479:  URL_PART_USERNAME','LongInt'( 3);
6480:  URL_PART_PASSWORD','LongInt'( 4);
6481:  URL_PART_PORT','LongInt'( 5);
6482:  URL_PART_QUERY','LongInt'( 6);
6483:  URL_PART', 'DWORD
6484:  URLIS_URL','LongInt'( 0);
6485:  URLIS_OPAQUE','LongInt'( 1);
6486:  URLIS_NOHISTORY','LongInt'( 2);
6487:  URLIS_FILEURL','LongInt'( 3);
6488:  URLIS_APPLIABLE','LongInt'( 4);
6489:  URLIS_DIRECTORY','LongInt'( 5);
6490:  URLIS_HASQUERY','LongInt'( 6);
6491:  TUrlIs', 'DWORD
6492:  URL_UNESCAPE','LongWord( $10000000);
6493:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6494:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6495:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6496:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6497:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6498:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6499:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6500:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6501:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6502:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6503:  URL_INTERNAL_PATH','LongWord( $00800000);
6504:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6505:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6506:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6507:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6508:  URL_APPLY_DEFAULT','LongWord( $00000001);
6509:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6510:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6511:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6512:  Func UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Int
6513:  Func UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6514:  Func UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6515:  Func UrlIsOpaque( pszURL : PChar) : BOOL
6516:  Func UrlIsNoHistory( pszURL : PChar) : BOOL
6517:  Func UrlIsFileUrl( pszURL : PChar) : BOOL
6518:  Func UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6519:  Func UrlGetLocation( psz1 : PChar) : PChar
6520:  Func UrlUnescape(pszUrl,pszUnescaped:PChar;pcchUnescaped:DWORD; dwFlags: DWORD): HRESULT
6521:  Func UrlEscape(pszUrl:PChar; pszEscaped:PChar; pcchEscaped:DWORD;dwFlags:DWORD): HRESULT
6522:  Func UrlCreateFromPath(pszPath:PChar;pszUrl:PChar;pcchUrl: DWORD;dwFlags:DWORD): HRESULT
6523:  Func PathCreateFromUrl(pszUrl:PChar; pszPath:PChar;pcchPath:DWORD; dwFlags:DWORD):HRESULT
6524:  Func UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6525:  Func UrlGetPart(pszIn:PChar; pszOut:PChar;pcchOut:DWORD; dwPart,dwFlags: DWORD): HRESULT
6526:  Func UrlApplyScheme(pszIn:PChar; pszOut:PChar; pcchOut : DWORD; dwFlags: DWORD): HRESULT
6527:  Func HashData(pbData:BYTE;cbData:DWORD; pbHash:BYTE; cbHash: DWORD) : HRESULT
6528:  Func UrlEscapeSpaces(pszUrl:PChar;pszEscaped:PChar;pcchEscaped: WORD): HRESULT
6529:  Func UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6530:  Func SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6531:  Func SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6532:  Func SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6533:  Func SHEnumKeyEx(hKey:HKEY; dwIndex : DWORD; pszName:PChar;var pcchName:DWORD) : Longint
6534:  Func SHEnumValue(hKey:HKEY;dwIndex:DWORD;pszValueName:PChar;var pcchValueName:DWORD;pdwType:DWORD;pvData:
       ___Pointer; pcbData:DWORD):Longint
6535:  Func SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6536:  Func SHCopyKey(hkeySrc:HKEY;szSrcSubKey:PChar; hkeyDest:HKEY; fReserved:DWORD) : DWORD
6537:  Func SHRegGetPath(hKey:HKEY;pcszSubKey,pcszValue:PChar;pszPath:PChar;dwFlags:DWORD):DWORD
6538:  Func SHRegSetPath(hKey:HKEY;pcszSubKey,pcszValue,pcszPath:PChar;dwFlags:DWORD): DWORD
6539:  SHREGDEL_DEFAULT','LongWord( $00000000);
6540:  SHREGDEL_HKCU','LongWord( $00000001);
6541:  SHREGDEL_HKLM','LongWord( $00000010);
6542:  SHREGDEL_BOTH','LongWord( $00000011);
6543:  SHREGENUM_DEFAULT','LongWord( $00000000);
6544:  SHREGENUM_HKCU','LongWord( $00000001);
6545:  SHREGENUM_HKLM','LongWord( $00000010);
6546:  SHREGENUM_BOTH','LongWord( $00000011);
6547:  SHREGSET_HKCU','LongWord( $00000001);
6548:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6549:  SHREGSET_HKLM','LongWord( $00000004);
6550:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6551:  TSHRegDelFlags', 'DWORD
6552:  TSHRegEnumFlags', 'DWORD
6553:  HUSKEY', 'THandle
6554:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6555:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6556:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6557:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6558:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6559:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
```

```
ASSOCF_NOTRUNCATE','LongWord( $00000020);
ASSOCF_VERIFY','LongWord( $00000040);
ASSOCF_REMAPRUNDLL','LongWord( $00000080);
ASSOCF_NOFIXUPS','LongWord( $00000100);
ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
ASSOCF', 'DWORD
ASSOCSTR_COMMAND','LongInt'( 1);
ASSOCSTR_EXECUTABLE','LongInt'( 2);
ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
ASSOCSTR_NOOPEN','LongInt'( 5);
ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
ASSOCSTR_DDECOMMAND','LongInt'( 7);
ASSOCSTR_DDEIFEXEC','LongInt'( 8);
ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
ASSOCSTR_DDETOPIC','LongInt'( 10);
ASSOCSTR_INFOTIP','LongInt'( 11);
ASSOCSTR_MAX','LongInt'( 12);
ASSOCSTR', 'DWORD
ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
ASSOCKEY_APP','LongInt'( 2);
ASSOCKEY_CLASS','LongInt'( 3);
ASSOCKEY_BASECLASS','LongInt'( 4);
ASSOCKEY_MAX','LongInt'( 5);
ASSOCKEY', 'DWORD
ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
ASSOCDATA_MAX','LongInt'( 5);
ASSOCDATA', 'DWORD
ASSOCENUM_NONE','LongInt'( 0);
ASSOCENUM', 'DWORD
SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
SHACF_DEFAULT $00000000);
SHACF_FILESYSTEM','LongWord( $00000001);
SHACF_URLHISTORY','LongWord( $00000002);
SHACF_URLMRU','LongWord( $00000004);
SHACF_USETAB','LongWord( $00000008);
SHACF_FILESYS_ONLY','LongWord( $00000010);
SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
Func SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
Proc SHSetThreadRef( punk : IUnknown)
Proc SHGetThreadRef( out ppunk : IUnknown)
CTF_INSIST','LongWord( $00000001);
CTF_THREAD_REF','LongWord( $00000002);
CTF_PROCESS_REF','LongWord( $00000004);
CTF_COINIT','LongWord( $00000008);
Func SHCreateShellPalette( hdc : HDC) : HPALETTE
Proc ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
Func ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
Func ColorAdjustLuma( clrRGB : TColorRef; n : Int; fScale :Bool) : TColorRef
Func GetSysColorBrush( nIndex : Int) : HBRUSH
Func DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
Func FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Int
Func FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Int
Func InvertRect( hDC : HDC; const lprc : TRect) : BOOL
Func SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Int) : BOOL
Func SetRectEmpty( var lprc : TRect) : BOOL
Func CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
Func InflateRect( var lprc : TRect; dx, dy : Int) : BOOL
Func IntersectRect2(var lprcDst: TRect; const lprcSrc1, lprcSrc2 : TRect):BOOL
Func SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect):BOOL
Func InitializeFlatSB( hWnd : HWND) : Bool
Proc UninitializeFlatSB( hWnd : HWND)
Func FlatSB_GetScrollProp( p1 : HWND; propIndex : Int; p3 : PInt) : Bool
Func FlatSB_SetScrollProp(p1: HWND; index: Int; newValue: Int; p4 : Bool):Bool
Func GET_APPCOMMAND_LPARAM( lParam : Int) : Word  //of JvWin32
Func GET_DEVICE_LPARAM( lParam : Int) : Word
Func GET_MOUSEORKEY_LPARAM( lParam : Int) : Int
Func GET_FLAGS_LPARAM( lParam : Int) : Word
Func GET_KEYSTATE_LPARAM( lParam : Int) : Word

//  ********************************************** 204 unit uPSI_ShellAPI;
Func DragQueryFile(Drop:HDROP;FileIndex:UINT; FileName: PChar; cb  UINT): UINT
Func DragQueryPoint( Drop : HDROP; var Point : TPoint): BOOL
Proc DragFinish( Drop : HDROP)
Proc DragAcceptFiles( Wnd : HWND; Accept : BOOL)
Func ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Int):HINST
Func FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
Func ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Int
Func DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
Func ExtractAssociatedIcon(hInst:HINST;lpIconPath:PChar;var lpiIcon:Word):HICON
Func ExtractIcon(hInst:HINST; lpszExeFileName:PChar; nIconIndex:UINT): HICON
Func SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
Func DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
```

```
6649: Func ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6650: Proc SHFreeNameMappings( hNameMappings : THandle)
6651:
6652: DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6653: DLLVER_PLATFORM_NT','LongWord( $00000002);
6654: DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6655: DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6656: DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6657: DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6658: Func MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6659: Func SimpleXMLEncode( const S :Str) :Str
6660: Proc SimpleXMLDecode( var S :Str; TrimBlanks :Bool)
6661: Func XMLEncode( const S :Str) :Str
6662: Func XMLDecode( const S :Str) :Str
6663: Func EntityEncode( const S :Str) :Str
6664: Func EntityDecode( const S :Str) :Str
6665:
6666: Proc RIRegister_CPort_Routines(S: TPSExec);
6667:  Proc EnumComPorts( Ports : TStrings)
6668:  Proc ListComPorts( Ports : TStrings)
6669:  Proc ComPorts( Ports : TStrings)  //Alias to Arduino
6670:  Func GetComPorts: TStringlist;
6671:  Func StrToBaudRate( Str :Str) : TBaudRate
6672:  Func StrToStopBits( Str :Str) : TStopBits
6673:  Func StrToDataBits( Str :Str) : TDataBits
6674:  Func StrToParity( Str :Str) : TParityBits
6675:  Func StrToFlowControl( Str :Str) : TFlowControl
6676:  Func BaudRateToStr( BaudRate : TBaudRate) :Str
6677:  Func StopBitsToStr( StopBits : TStopBits) :Str
6678:  Func DataBitsToStr( DataBits : TDataBits) :Str
6679:  Func ParityToStr( Parity : TParityBits) :Str
6680:  Func FlowControlToStr( FlowControl : TFlowControl) :Str
6681:  Func ComErrorsToStr( Errors : TComErrors) :Str
6682:  Func GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6683:  Func DispatchMessage( const lpMsg : TMsg) : Longint
6684:  Func TranslateMessage( const lpMsg : TMsg) : BOOL
6685:  Func SetMessageQueue( cMessagesMax : Int) : BOOL
6686:  Func PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6687:  Func GetMessagePos : DWORD
6688:  Func GetMessageTime : Longint
6689:  Func GetMessageExtraInfo : Longint
6690:  Func GetSpecialFolderPath( const FolderName :Str; CanCreate :Bool) :Str
6691:  Proc JAddToRecentDocs( const Filename :Str)
6692:  Proc ClearRecentDocs
6693:  Func ExtractIconFromFile( FileName :Str; Index : Int) : HICON
6694:  Func CreateShellLink( const AppName, Desc :Str; Dest :Str) :Str
6695:  Proc GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6696:  Proc SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6697:  Func RecycleFile( FileToRecycle :Str) :Bool
6698:  Func JCopyFile( FromFile, ToDir :Str) :Bool
6699:  Func ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6700:  Func ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6701:  Func QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel: DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6702:  Func QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel: DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6703:  Func QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel:DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD;
      var pcbBytesNeeded : DWORD) : BOOL
6704:  Func EnumServicesStatusExA(hSCManager:SC_HANDLE;InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6705:  Func EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6706:  Func EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState:DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6707:  Func ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6708:
6709: ******************************************** unit uPSI_JclPeImage;
6710:  Func IsValidPeFile( const FileName : TFileName) :Bool
6711: // Func PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders): Bool
6712:  Func PeCreateNameHintTable( const FileName : TFileName) :Bool
6713:  Func PeRebaseImage(const ImageName:TFileName; NewBase:DWORD; TimeStamp : DWORD; MaxNewSize:DWORD):
      TJclRebaseImageInfo
6714:  Func PeVerifyCheckSum( const FileName : TFileName) :Bool
6715:  Func PeClearCheckSum( const FileName : TFileName) :Bool
6716:  Func PeUpdateCheckSum( const FileName : TFileName) :Bool
6717:  Func PeDoesExportFunction(const FileName:TFileName;const FuncName:str;Options:TJclSmartCompOptions):Bool;
6718:  Func PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName:str; var
      ForwardedName:str;Options:TJclSmartCompOptions):Boolean
6719:  Func PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:str;Options:TJclSmartCompOptions):Bool
6720:  Func PeDoesImportFunction(const FileName:TFileName;const FunctionName:str;const LibraryName:str;Options:
      TJclSmartCompOptions):Bool
6721:  Func PeDoesImportLibrary(const FileName:TFileName;const LibraryName:str;Recursive:Boolean):Boolean);
6722:  Func PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :Bool;
      FullPathName :Bool) :Bool
6723:  Func PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const LibraryName:str;
      IncludeLibNames :Bool):Bool
```

```
6724:  Func PeExportedFunctions(const FileName:TFileName;const FunctionsList:TStrings):Boolean
6725:  Func PeExportedNames(const FileName:TFileName;const FunctionsList:TStrings):Bool
6726:  Func PeExportedVariables(const FileName:TFileName;const FunctionsList:TStrings):Boolean
6727:  Func PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
       NamesList:TStrings):Bool
6728:  Func PeBorFormNames(const FileName:TFileName;const NamesList:TStrings :Boolean
6729:  Func PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
       Descript:Bool):Bool;
6730:  Func PeFindMissingImports(const FileName:TFileName;MissingImportsList:TStrings):Bool;
6731:  Func PeFindMissingImports1(RequiredImportsList,MissingImportsList:TStrings):Bool;
6732:  Func PeCreateRequiredImportList(const FileName:TFileName;RequiredImportsList:TStrings): Bool;
6733:  //Func PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6734:  //Func PeMapImgLibraryName( const BaseAddress : Pointer) :Str
6735:  //Func PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6736:  //Func PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName :Str) :
       PImageSectionHeader
6737:  //Func PeMapImgExportedVariables(const Module:HMODULE;const VariablesList:TStrings):Bool
6738:  //Func PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6739:  Func PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:str): ___Pointer;
6740:   SIRegister_TJclPeSectionStream(CL);
6741:   SIRegister_TJclPeMapImgHookItem(CL);
6742:   SIRegister_TJclPeMapImgHooks(CL);
6743:  //Func PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var NtHeaders:TImageNtHeaders):Boolean
6744:  //Func PeDbgImgLibraryName(ProcessHandle:THandle;BaseAddress:Pointer;var Name:str):Bool
6745:   Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6746:   TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6747:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6748:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6749:   TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6750:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6751:  Func PeBorUnmangleName( const Name :Str; var Unmangled :Str; var Description : TJclBorUmDescription; var
       BasePos : Int) : TJclBorUmResult;
6752:  Func PeBorUnmangleName1(const Name:str;var Unmangled:str;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6753:  Func PeBorUnmangleName2( const Name :Str; var Unmangled :Str) : TJclBorUmResult;
6754:  Func PeBorUnmangleName3( const Name :Str) :Str;
6755:  Func PeIsNameMangled( const Name :Str) : TJclPeUmResult
6756:  Func PeUnmangleName( const Name :Str; var Unmangled :Str) : TJclPeUmResult
6757:
6758:
6759: //****************** SysTools uPSI_StSystem; ****************************************
6760:  Func StCopyFile( const SrcPath, DestPath : Ansistr) :Card
6761:  Func CreateTempFile( const aFolder : Ansistr; const aPrefix : Ansistr): Ansistr
6762:  Func DeleteVolumeLabel( Drive : Char) :Card
6763:  //Proc EnumerateDirectories(const StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6764:  //Proc EnumerateFiles(const StartDir:Ansistr;FL:TStrings;SubDirs:Bool IncludeItem:TIncludeItemFunc);
6765:  Func FileHandlesLeft( MaxHandles :Card) :Card
6766:  Func FileMatchesMask( const FileName, FileMask : Ansistr) :Bool
6767:  Func FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6768:  Func FindNthSlash( const Path : Ansistr; n : Int) : Int
6769:  Func FlushOsBuffers( Handle : Int) :Bool
6770:  Func GetCurrentUser : Ansistr
6771:  Func GetDiskClass( Drive : Char) : DiskClass
6772:  Func GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,SectorsPerCluster:Card):Bool;
6773:  Func GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6774:  Func GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var DiskSize:Comp):Bool;
6775:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6776:  Func getDiskSpace2(const path:Str; index: Int): int64;
6777:  Func GetFileCreateDate( const FileName : Ansistr) : TDateTime
6778:  Func StGetFileLastAccess( const FileName : Ansistr) : TDateTime
6779:  Func GetFileLastModify( const FileName : Ansistr) : TDateTime
6780:  Func GetHomeFolder( aForceSlash :Bool) : Ansistr
6781:  Func GetLongPath( const APath : Ansistr) : Ansistr
6782:  Func GetMachineName : Ansistr
6783:  Func GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) :Card
6784:  Func GetParentFolder( const APath : Ansistr; aForceSlash :Bool) : Ansistr
6785:  Func GetShortPath( const APath : Ansistr) : Ansistr
6786:  Func GetSystemFolder( aForceSlash :Bool) : Ansistr
6787:  Func GetTempFolder( aForceSlash :Bool) : Ansistr
6788:  Func StGetWindowsFolder( aForceSlash :Bool) : Ansistr
6789:  Func GetWorkingFolder( aForceSlash :Bool) : Ansistr
6790:  Func GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Int) : TStDateTimeRec
6791:  Func StIsDirectory( const DirName : Ansistr) :Bool
6792:  Func IsDirectoryEmpty( const S : Ansistr) : Int
6793:  Func IsDriveReady( Drive : Char) :Bool
6794:  Func IsFile( const FileName : Ansistr) :Bool
6795:  Func IsFileArchive( const S : Ansistr) : Int
6796:  Func IsFileHidden( const S : Ansistr) : Int
6797:  Func IsFileReadOnly( const S : Ansistr) : Int
6798:  Func IsFileSystem( const S : Ansistr) : Int
6799:  Func LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Int) : TStDateTimeRec
6800:  Func ReadVolumeLabel( var VolName : Ansistr; Drive : Char) :Card
6801:  Func SameFile( const FilePath1, FilePath2 : Ansistr; var ErrorCode : Int) :Bool
6802:  Func SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) :Card
6803:  Proc SplitPath( const APath : Ansistr; Parts : TStrings)
6804:  Func StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6805:  Func StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6806:  Func UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
```

```
6807:  Func ValidDrive( Drive : Char) :Bool
6808:  Func WriteVolumeLabel( const VolName : Ansistr; Drive : Char) :Card
6809:
6810: //*****************************unit uPSI_JclLANMan;*****************************************
6811: Func CreateAccount( const Server,Username,Fullname,Password,Description,Homedir,Script :str; const
      PasswordNeverExpires : Bool) : Bool
6812:  Func CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script :Str; const
      PasswordNeverExpires :Bool) :Bool
6813:  Func DeleteAccount( const Servername, Username :Str) :Bool
6814:  Func DeleteLocalAccount( Username :Str) :Bool
6815:  Func CreateLocalGroup( const Server, Groupname, Description :Str) :Bool
6816:  Func CreateGlobalGroup( const Server, Groupname, Description :Str) :Bool
6817:  Func DeleteLocalGroup( const Server, Groupname :Str) :Bool
6818:  Func GetLocalGroups( const Server :Str; const Groups : TStrings) :Bool
6819:  Func GetGlobalGroups( const Server :Str; const Groups : TStrings) :Bool
6820:  Func LocalGroupExists( const Group :Str) :Bool
6821:  Func GlobalGroupExists( const Server, Group :Str) :Bool
6822:  Func AddAccountToLocalGroup( const Accountname, Groupname :Str) :Bool
6823:  Func LookupGroupName( const Server :Str; const RID : TNetWellKnownRID) :Str
6824:  Proc ParseAccountName( const QualifiedName :Str; var Domain, UserName :Str)
6825:  Func IsLocalAccount( const AccountName :Str) :Bool
6826:  Func TimeStampInterval( StartStamp, EndStamp : TDateTime) : Int
6827:  Func GetRandomString( NumChar :Card) :Str
6828:
6829: //*****************************unit uPSI_cUtils;*******************************************
6830: TypeS(TUnitType', ( utSrc, utHead, utRes, utPrj, utOther )
6831: Func cIsWinNT :Bool
6832:  Proc cFilesFromWildcard(Directory,Mask:Str;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Boolean;
6833:  Func cExecuteFile( const FileName, Params, DefaultDir :Str; ShowCmd : Int) : THandle
6834:  Func cRunAndGetOutput(Cmd,WorkDir:str; ErrFunc:TErrFunc;LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc:TCheckAbortFunc;ShowReturnValue:Bool):str
6835:  Func cGetShortName( FileName :Str) :Str
6836:  Proc cShowError( Msg :Str)
6837:  Func cCommaStrToStr( s :Str; formatstr :Str) :Str
6838:  Func cIncludeQuoteIfSpaces( s :Str) :Str
6839:  Func cIncludeQuoteIfNeeded( s :Str) :Str
6840:  Proc cLoadFilefromResource( const FileName :Str; ms : TMemoryStream)
6841:  Func cValidateFile(const FileName:str;const WorkPath:str;const CheckDirs:bool):str;
6842:  Func cBuildFilter( var value :Str; const FLTStyle : TFILTERSET) :Bool;
6843:  Func cBuildFilter1( var value :Str; const _filters : array of string) :Bool;
6844:  Func cCodeInstoStr( s :Str) :Str
6845:  Func cStrtoCodeIns( s :Str) :Str
6846:  Proc cStrtoAttr( var Attr : TSynHighlighterAttributes; Value :Str)
6847:  Func cAttrtoStr( const Attr : TSynHighlighterAttributes) :Str
6848:  Proc cStrtoPoint( var pt : TPoint; value :Str)
6849:  Func cPointtoStr( const pt : TPoint) :Str
6850:  Func cListtoStr( const List : TStrings) :Str
6851:  Func ListtoStr( const List : TStrings) :Str
6852:  Proc StrtoList( s :Str; const List : TStrings; const delimiter : char)
6853:  Proc cStrtoList( s :Str; const List : TStrings; const delimiter : char)
6854:  Func cGetFileTyp( const FileName :Str) : TUnitType
6855:  Func cGetExTyp( const FileName :Str) : TExUnitType
6856:  Proc cSetPath( Add :Str; const UseOriginal :Bool)
6857:  Func cExpandFileto( const FileName :Str; const BasePath :Str) :Str
6858:  Func cFileSamePath( const FileName :Str; const TestPath :Str) :Bool
6859:  Proc cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6860:  Func cGetLastPos( const SubStr :Str; const S :Str) : Int
6861:  Func cGenMakePath( FileName :Str) :Str;
6862:  Func cGenMakePath2( FileName :Str) :Str
6863:  Func cGenMakePath1(FileName:Str; EscapeSpaces,EncloseInQuotes:Bool):str;
6864:  Func cGetRealPath( BrokenFileName :Str; Directory :Str) :Str
6865:  Func cCalcMod( Count : Int) : Int
6866:  Func cGetVersionString( FileName :Str) :Str
6867:  Func cCheckChangeDir( var Dir :Str) :Bool
6868:  Func cGetAssociatedProgram(const Extension:str;var Filename,Description:str):bool
6869:  Func cIsNumeric( s :Str) :Bool
6870:  Proc StrtoAttr( var Attr : TSynHighlighterAttributes; Value :Str)
6871:  Func AttrtoStr( const Attr : TSynHighlighterAttributes) :Str
6872:  Func GetFileTyp( const FileName :Str) : TUnitType
6873:  Func Atoi(const aStr:Str): Int
6874:  Func Itoa(const aint: Int):Str
6875:  Func Atof(const aStr:Str): double;
6876:  Func Atol(const aStr:Str): longint;
6877:
6878: Proc SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6879: begin
6880:   FindClass('TOBJECT'),'EHTTP
6881:   FindClass('TOBJECT'),'EHTTPParser
6882:   //AnsiCharSet', 'set of AnsiChar
6883:   AnsistrArray', 'array of Ansistr
6884:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6885:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6886:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6887:    +'TTPProtocolEnum; CustomProtocol : Ansistr; CustomMajVersion : Int; '
6888:    +'CustomMinVersion : Int; end
6889:   THTTPHeaderNameEnum','( hntCustom, hntHost, hntContentType, hnt'
6890:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6891:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6892:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
```

```
6893:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6894:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6895:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6896:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6897:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6898:    +'nection, hntOrigin, hntKeepAlive )
6899:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: Ansistr; end
6900:   THTTPCustomHeader', 'record FieldName : Ansistr; FieldValue :'
6901:    +' Ansistr; end
6902:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6903:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6904:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6905:   //PHTTPContentLength', '^THTTPContentLength // will not work
6906:   THTTPContentTypeMajor','( hctmCustom, hctmText, hctmImage )
6907:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6908:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6909:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6910:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6911:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6912:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6913:    +ajor:Ansistr;CustomMinor:Ansistr;Parameters:AnsistrArray;CustomStr:Ansistr;end
6914:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6915:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6916:    +' Int; Day : Int; Month : Int; Year : Int; Hour : Int; '
6917:    +'Min : Int; Sec : Int; TimeZoneGMT :Bool; CustomTimeZone : Ansi'
6918:    +'String; DateTime : TDateTime; Custom : Ansistr; end
6919:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6920:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum;Custom:Ansistr;end
6921:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6922:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;Custom:Ansistr; end
6923:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6924:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:Ansistr; end
6925:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6926:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6927:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6928:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6929:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6930:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6931:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6932:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6933:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6934:   THTTPContentEncoding','record Value:THTTPContentEncodingEnum;Custom:Ansistr;end;
6935:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6936:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6937:    +'FieldEnum; List : array of THTTPContentEncoding; end
6938:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6939:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6940:    +' Custom : Ansistr; Date : TDateTime; Seconds : Int64; end
6941:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6942:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6943:    +num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom: Ansistr; end
6944:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6945:   THTTPSetCookieCustomField', 'record Name : Ansistr; Value : Ansistr; end
6946:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6947:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6948:    +'omain : Ansistr; Path : Ansistr; Expires : THTTPDateField; MaxAge : '
6949:    +'Int64; HttpOnly :Bool; Secure :Bool; CustomFields : THTTPSetCookie'
6950:    +'CustomFieldArray; Custom : Ansistr; end
6951:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6952:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6953:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6954:   THTTPCookieFieldEntry', 'record Name : Ansistr; Value : Ansistr; end
6955:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6956:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6957:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6958:    +' : THTTPCookieFieldEntryArray; Custom : Ansistr; end
6959:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6960:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6961:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6962:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6963:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6964:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of Ansistr
6965:   THTTPFixedHeaders','array[0..42] of Ansistr
6966:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6967:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6968:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : Ansistr; end
6969:   THTTPRequestStartLine','record Method:THTTPMethod;URI:Ansistr;Version:THTTPVersion;end
6970:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6971:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6972:    +'kie:THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6973:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6974:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6975:    +' : THTTPRequestHeader; HeaderComplete :Bool; HasContent :Bool; end
6976:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6977:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6978:    +'Int; Msg : THTTPResponseStartLineMessage; CustomMsg : Ansistr; end
6979:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6980:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6981:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
```

```
6982:    +' THTTPDateField; Age : THTTPAgeField; end
6983:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6984:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6985:    +er : THTTPResponseHeader; HeaderComplete :Bool; HasContent :Bool; end
6986:  Func HTTPMessageHasContent( const H : THTTPCommonHeaders) :Bool
6987:  Proc InitHTTPRequest( var A : THTTPRequest)
6988:  Proc InitHTTPResponse( var A : THTTPResponse)
6989:  Proc ClearHTTPVersion( var A : THTTPVersion)
6990:  Proc ClearHTTPContentLength( var A : THTTPContentLength)
6991:  Proc ClearHTTPContentType( var A : THTTPContentType)
6992:  Proc ClearHTTPDateField( var A : THTTPDateField)
6993:  Proc ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6994:  Proc ClearHTTPConnectionField( var A : THTTPConnectionField)
6995:  Proc ClearHTTPAgeField( var A : THTTPAgeField)
6996:  Proc ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6997:  Proc ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6998:  Proc ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6999:  Proc ClearHTTPSetCookieField( var A : THTTPSetCookieField)
7000:  Proc ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
7001:  //Proc ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
7002:  Proc ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
7003:  Proc ClearHTTPCookieField( var A : THTTPCookieField)
7004:  Proc ClearHTTPMethod( var A : THTTPMethod)
7005:  Proc ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
7006:  Proc ClearHTTPRequestHeader( var A : THTTPRequestHeader)
7007:  Proc ClearHTTPRequest( var A : THTTPRequest)
7008:  Proc ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
7009:  Proc ClearHTTPResponseHeader( var A : THTTPResponseHeader)
7010:  Proc ClearHTTPResponse( var A : THTTPResponse)
7011:   THTTPStringOption', '( hsoNone )
7012:   THTTPStringOptions', 'set of THTTPStringOption
7013:   FindClass('TOBJECT'),'TAnsistrBuilder
7014:  Proc BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsistrBuilder; PTHTTPStringOptions;
7015:  Proc BuildStrHTTPContentLengthValue(const A:THTTPContentLength;B:TAnsistrBuilder;P:THTTPStringOptions)
7016:  Proc BuildStrHTTPContentLength(const A : THTTPContentLength; B:TAnsistrBuilder;PTHTTPStringOptions)
7017:  Proc BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsistrBuilder;const P:THTTPStringOptions)
7018:  Proc BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsistrBuilder; const P:THTTPStringOptions)
7019:  Proc BuildStrRFCDateTime(const DOW,Da,Mo,Ye,Ho,Mi,Se:Int;const TZ:Ansistr;const B:TAnsistrBuilder;const
       P:THTTPStringOptions)
7020:  Proc BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7021:  Proc BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsistrBuilder;const PTHTTPStringOptions);
7022:  Proc BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B : TAnsistrBuilder; const
       P : THTTPStringOptions)
7023:  Proc BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7024:  Proc BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsistrBuilder; const P
       : THTTPStringOptions)
7025:  Proc BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsistrBuilder; const P
       : THTTPStringOptions)
7026:  Proc BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7027:  Proc BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsistrBuilder;const P:THTTPStringOptions);
7028:  Proc BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7029:  Proc BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const B:TAnsistrBuilder;const
       P:THTTPStringOptions)
7030:  Proc BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsistrBuilder; const P
       : THTTPStringOptions)
7031:  Proc BuildStrHTTPCommonHeaders( const A: THTTPCommonHeaders; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7032:  Proc BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions)
7033:  Proc BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7034:  Proc BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7035:  Proc BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7036:  Proc BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const P:THTTPStringOptions);
7037:  Proc BuildStrHTTPMethod(const A:THTTPMethod;const B:TAnsistrBuilder;const P: THTTPStringOptions)
7038:  Proc BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7039:  Proc BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsistrBuilder;const
       P:THTTPStrOptions);
7040:  Proc BuildStrHTTPRequest(const A : THTTPRequest; const B: TAnsistrBuilder; const P : THTTPStringOptions)
7041:  Proc BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B : TAnsistrBuilder;
       const P : THTTPStringOptions)
7042:  Proc BuildStrHTTPResponseStartLine(const A:THTTPResponsStartLine;const B:TAnsiStrBldr;const P
       THTTPStrOptions);
7043:  Proc BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
7044:  Proc BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsistrBuilder; const P:THTTPStringOptions);
7045:  Func HTTPContentTypeValueToStr( const A : THTTPContentType) : Ansistr
7046:  Func HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : Ansistr
7047:  Func HTTPCookieFieldValueToStr( const A : THTTPCookieField) : Ansistr
7048:  Func HTTPMethodToStr( const A : THTTPMethod) : Ansistr
7049:  Func HTTPRequestToStr( const A : THTTPRequest) : Ansistr
7050:  Func HTTPResponseToStr( const A : THTTPResponse) : Ansistr
```

```
7051:  Proc PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const Domain:Ansistr;const
      Secure:Bool; THTTPParserHeaderParseFunc','Func (const HeaderName : THTT'
7052:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) :Bool
7053:   SIRegister_THTTPParser(CL);
7054:   FindClass('TOBJECT'),'THTTPContentDecoder
7055:   THTTPContentDecoderProc', 'Proc ( const Sender : THTTPContentDecoder)
7056:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
7057:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
7058:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
7059:   THTTPContentDecoderLogEvent,Procedure(const Sender:THTTPContentDecoder;const LogMsg:str)
7060:   SIRegister_THTTPContentDecoder(CL);
7061:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
7062:   FindClass('TOBJECT'),'THTTPContentReader
7063:   THTTPContentReaderProc','Proc ( const Sender : THTTPContentReader)
7064:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:str;const
      LogLevel:Int;
7065:   SIRegister_THTTPContentReader(CL);
7066:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
7067:   FindClass('TOBJECT'),'THTTPContentWriter
7068:   THTTPContentWriterLogEvent,Procedure(const Sender:THTTPContentWriter;const LogMsg:Ansistr);
7069:   SIRegister_THTTPContentWriter(CL);
7070:  Proc SelfTestcHTTPUtils
7071: end;
7072:
7073: (*----------------------------------------------------------------------------*)
7074: Proc SIRegister_cTLSUtils(CL: TPSPascalCompiler);
7075: begin
7076:  'TLSLibraryVersion','String '1.00
7077:  'TLSError_None','LongInt'( 0);
7078:  'TLSError_InvalidBuffer','LongInt'( 1);
7079:  'TLSError_InvalidParameter','LongInt'( 2);
7080:  'TLSError_InvalidCertificate','LongInt'( 3);
7081:  'TLSError_InvalidState','LongInt'( 4);
7082:  'TLSError_DecodeError','LongInt'( 5);
7083:  'TLSError_BadProtocol','LongInt'( 6);
7084:  Func TLSErrorMessage( const TLSError : Int) :Str
7085:   SIRegister_ETLSError(CL);
7086:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
7087:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
7088:  Proc InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
7089:  Proc InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
7090:  Proc InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
7091:  Proc InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
7092:  Func IsTLSProtocolVersion( const A, B : TTLSProtocolVersion):Bool
7093:  Func IsSSL2( const A : TTLSProtocolVersion) :Boolean
7094:  Func IsSSL3( const A : TTLSProtocolVersion) :Boolean
7095:  Func IsTLS10( const A : TTLSProtocolVersion):Boolean
7096:  Func IsTLS11( const A : TTLSProtocolVersion):Boolean
7097:  Func IsTLS12( const A : TTLSProtocolVersion):Boolean
7098:  Func IsTLS10OrLater( const A : TTLSProtocolVersion) :Bool
7099:  Func IsTLS11OrLater( const A : TTLSProtocolVersion) :Bool
7100:  Func IsTLS12OrLater( const A : TTLSProtocolVersion) :Bool
7101:  Func IsFutureTLSVersion( const A : TTLSProtocolVersion) :Bool
7102:  Func IsKnownTLSVersion( const A : TTLSProtocolVersion) :Bool
7103:  Func TLSProtocolVersionToStr( const A : TTLSProtocolVersion):Str
7104:  Func TLSProtocolVersionName( const A : TTLSProtocolVersion) :Str
7105:   PTLSRandom', '^TTLSRandom // will not work
7106:  Proc InitTLSRandom( var Random : TTLSRandom)
7107:  Func TLSRandomToStr( const Random : TTLSRandom) : Ansistr
7108:  'TLSSessionIDMaxLen','LongInt'( 32);
7109:  Proc InitTLSSessionID( var SessionID : TTLSSessionID; const A : Ansistr)
7110:  Func EncodeTLSSessionID(var Buff:str;const Size:Int;const SessionID:TTLSSessionID):Int;
7111:  Func DecodeTLSSessionID(const Buff:str;const Size:Int;var SessionID:TTLSSessionID):Int;
7112:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
7113:    +'; Signature : TTLSSignatureAlgorithm; end
7114:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
7115:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
7116:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
7117:    +DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
7118:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
7119:    +HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
7120:   TTLSMacAlgorithmInfo', 'record Name:Ansistr;DigestSize:Integer; Supported:Bool; end
7121:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
7122:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
7123:   TTLSPRFAlgorithm', '( tlspaSHA256 )
7124:  Func tlsP_MD5( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7125:  Func tlsP_SHA1( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7126:  Func tlsP_SHA256( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7127:  Func tlsP_SHA512( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7128:  Func tls10PRF( const Secret, ALabel, Seed : Ansistr; const Size : Int) : Ansistr
7129:  Func tls12PRF_SHA256(const Secret,ALabel,Seed:Ansistr; const Size : Int): Ansistr
7130:  Func tls12PRF_SHA512(const Secret,ALabel,Seed :Ansistr; const Size: Int): Ansistr
7131:  Func TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
7132:  Func tls10KeyBlock(const MasterSecret,ServerRandom,ClientRandom:Ansistr;const Size:Int):AnsiStr
7133:  Func tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: Ansistr;const Size:Int):Ansistr;
7134:  Func tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: Ansistr;const Size:Int):Ansistr;
7135:  Func TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion;const MasterSecret,ServerRandom,ClientRandom:
      Ansistr;const Size:Int):Ansistr
```

```
7136:  Func tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:Ansistr) :Ansistr;
7137:  Func tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiStr):AnsiStr;
7138:  Func tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiStr): AnsiStr;
7139:  Func TLSMasterSecret(const ProtocolVersion:TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
       ServerRandom:Ansistr):Ansistr
7140:    TTLSKeys', 'record KeyBlock : Ansistr; ClientMACKey : AnsiStr'
7141:     +'ing; ServerMACKey : Ansistr; ClientEncKey : Ansistr; ServerEncKey : '
7142:     +'Ansistr; ClientIV : Ansistr; ServerIV : Ansistr; end
7143:  Proc GenerateTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const MACKeyBits,CipherKeyBits,
       IVBits:Int;const MasterSecret,ServerRandom,ClientRandom:Ansistr;var TLSKeys: TTLSKeys)
7144:  Proc GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Bool;const
       ExpandedKeyBits:Int;const ServerRandom,ClientRandom:Ansistr;var TLSKeys:TTLSKeys)
7145:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
7146:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7147:  Proc SelfTestcTLSUtils
7148: end;
7149:
7150: (*----------------------------------------------------------------------------*)
7151: Proc SIRegister_Reversi(CL: TPSPascalCompiler);
7152: begin
7153:  sPosData','record
       corner:boolean;square2x2:boolean;edge:bool;stable:Int;internal:Int;disks:Int;mx:Int;my:Int; end
7154:  // pBoard', '^tBoard // will not work
7155:  Func rCalculateData( cc : byte; cx, cy : Int) : sPosData
7156:  Func rCheckMove( color : byte; cx, cy : Int) : Int
7157:  //Func rDoStep( data : pBoard) : word
7158:  Func winExecAndWait( const sAppPath :Str; wVisible : word) :Bool
7159: end;
7160:
7161: Proc SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7162: begin
7163: Func InEditMode( ADataset : TDataset) :Bool
7164: Func CheckDataSource( ADataSource : TDataSource) :Bool;
7165: Func CheckDataSource1(ADataSource:TDataSource;const AFieldName:str;var VField:TField):bool;
7166: Func GetFieldText( AField:TField) :Str
7167: end;
7168:
7169: Proc SIRegister_SortGrid(CL: TPSPascalCompiler);
7170: begin
7171:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7172:   TMyPrintRange', '( prAll, prSelected )
7173:   TSortStyle', '(ssAutomatic,ssNormal,ssNumeric,ssNumericExtended,ssDateTime,ssTime,ssCustom )
7174:   TSortDirection', '( sdAscending, sdDescending )
7175:   TSetChecked', 'Proc ( Sender : TObject; ACol, ARow : Int; State :Bool)
7176:   TGetComboBox', 'Proc ( Sender : TObject; ACol, ARow : integ'
7177:    +er; var Strs : TStringList; var Width, Height : Int; var Sorted :Bool)
7178:   TSetComboBox', 'Proc ( Sender : TObject; ACol, ARow : Int; Str :Str)
7179:   TSetEllipsis', 'Proc ( Sender : TObject; ACol, ARow : Int)
7180:   SIRegister_TSortOptions(CL);
7181:   SIRegister_TPrintOptions(CL);
7182:   TSortedListEntry', 'record Str :Str; RowNum : Int; SortOption : TSortOptions; end
7183:   SIRegister_TSortedList(CL);
7184:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7185:   TCellBevel', 'record Style: TCellBevelStyle;UpperLeftColor:TColor;LowerRightColor:TColor;end
7186:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7187:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7188:    +Horz : TAlignment;AlignmentVert:TVertAlignment;Bevel:TCellBevel;HideText:Boolean; end
7189:   SIRegister_TFontSetting(CL);
7190:   SIRegister_TFontList(CL);
7191:   AddTypeS(TFormatDrawCellEvent', 'Proc ( Sender : TObject; Col, Row :'
7192: +Int;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7193:   TSetFilterEvent', 'Proc ( ARows : TStrings; var Accept :Bool)
7194:   TSearchEvent', 'Proc ( ARows : TStrings; var Accept :Bool)
7195:   TUpdateGridEvent', 'Proc ( Sender : TObject; ARow : Int)
7196:   TSizeChangedEvent', 'Proc ( Sender : TObject; OldColCount, OldRowCount : Int)
7197:   TBeginSortEvent', 'Proc ( Sender : TObject; var Col : Int)
7198:   TEndSortEvent', 'Proc ( Sender : TObject; Col : Int)
7199:   TGetSortStyleEvent', 'Proc ( Sender : TObject; Col : integer; var SortStyle : TSortStyle)
7200:   TCellValidateEvent', 'Proc ( Sender : TObject; ACol, ARow :'
7201:    + Int; const OldValue :Str; var NewValue :Str; var Valid :Bool)
7202:   SIRegister_TSortGrid(CL);
7203:  Func ExtendedCompare( const Str1, Str2 :Str) : Int
7204:  Func NormalCompare( const Str1, Str2 :Str) : Int
7205:  Func DateTimeCompare( const Str1, Str2 :Str) : Int
7206:  Func NumericCompare( const Str1, Str2 :Str) : Int
7207:  Func TimeCompare( const Str1, Str2 :Str) : Int
7208:  //Func Compare( Item1, Item2 : Pointer) : Int
7209: end;
7210:
7211: ************************************* Proc Register_IB(CL: TPSPascalCompiler);
7212:  Proc IBAlloc( var P, OldSize, NewSize : Int)
7213:  Proc IBError( ErrMess : TIBClientError; const Args : array of const)
7214:  Proc IBDataBaseError
7215:  Func StatusVector : PISC_STATUS
7216:  Func StatusVectorArray : PStatusVector
7217:  Func CheckStatusVector( ErrorCodes : array of ISC_STATUS) :Bool
7218:  Func StatusVectorAsText :Str
7219:  Proc SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7220:  Func GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
```

```
7221:
7222: //***********************unit uPSI_BoldUtils;**********************************
7223:  Func CharCount( c : char; const s :Str) : Int
7224:  Func BoldNamesEqual( const name1, name2 :Str) :Bool
7225:  Proc BoldAppendToStrings(strings:TStrings; const aString:str;const ForceNewLine:Bool)
7226:  Func BoldSeparateStringList(strings:TStringList;const Separatr,PreString,PostString:Str)Str
7227:  Func BoldSeparatedAppend( const S1, S2 :Str; const Separator :Str) :Str
7228:  Func BoldTrim( const S :Str) :Str
7229:  Func BoldIsPrefix( const S, Prefix :Str) :Bool
7230:  Func BoldStrEqual( P1, P2 : PChar; Len : Int) :Bool
7231:  Func BoldStrAnsiEqual( P1, P2 : PChar; Len : Int) :Bool
7232:  Func BoldAnsiEqual( const S1, S2 :Str) :Bool
7233:  Func BoldStrStringEqual( const S1 :Str; P2 : PChar; Len : Int) :Bool
7234:  Func BoldCaseIndependentPos( const Substr, S :Str) : Int
7235:  //Proc EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7236:  Func CapitalisedToSpaced( Capitalised :Str) :Str
7237:  Func SpacedToCapitalised( Spaced :Str) :Str
7238:  Func BooleanToString( BoolValue :Bool) :Str
7239:  Func StringToBoolean( StrValue :Str) :Bool
7240:  Func GetUpperLimitForMultiplicity( const Multiplicity :Str) : Int
7241:  Func GetLowerLimitForMultiplicity( const Multiplicity :Str) : Int
7242:  Func StringListToVarArray( List : TStringList) : variant
7243:  Func IsLocalMachine( const Machinename : WideString) :Bool
7244:  Func GetComputerNameStr :Str
7245:  Func TimeStampComp( const Time1, Time2 : TTimeStamp) : Int
7246:  Func BoldStrToDateFmt(const S:str;const DateFormat:str;const DateSeparatorChar:char):TDateTime
7247:  Func BoldDateToStrFmt(const aDate:TDateTime;DateFormat:str;const DateSeparatorChar:char):Str;
7248:  Func BoldParseFormattedDateList(const value:str;const formats:TStrings;var Date:TDateTime):Boolean;
7249:  Func BoldParseFormattedDate(const value:str;const formats:array of string; var Date:TDateTime):Boolean;
7250:  Proc EnsureTrailing( var Str :Str; ch : char)
7251:  Func BoldDirectoryExists( const Name :Str) :Bool
7252:  Func BoldForceDirectories( Dir :Str) :Bool
7253:  Func BoldRootRegistryKey :Str
7254:  Func GetModuleFileNameAsString( IncludePath :Bool) :Str
7255:  Func BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Int
7256:  Func LogicalAnd( A, B : Int) :Bool
7257:  record TByHandleFileInformation dwFileAttributes: DWORD; '
7258:    +ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7259:    +: TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7260:    +eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow:DWORD; end
7261:  Func GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7262:  Func IsFirstInstance :Bool
7263:  Proc ActivateFirst( AString : PChar)
7264:  Proc ActivateFirstCommandLine
7265:  Func MakeAckPkt(const BlockNumber: Word):Str;
7266:  Proc SendError(UDPBase:TIdUDPBase;APeerIP:str;const APort:Int;const ErrNumb:Word;ErrStr:str;
7267:  Proc SendError(UDPClient:TIdUDPClient;const ErrNumber:Word; ErrorString:str);overload;
7268:  Proc SendError(UDPBase:TIdUDPBase;APeerIP:str;const APort:Int; E:Exception);overload;
7269:  Proc SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7270:  Func IdStrToWord(const Value:Str): Word;
7271:  Func IdWordToStr(const Value: Word): WordStr;
7272:  Func HasInstructionSet( const InstructionSet : TCPUInstructionSet) :Bool
7273:  Func CPUFeatures : TCPUFeatures
7274:
7275: Proc SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7276: begin
7277:   AddTypeS('TXRTLBitIndex', 'Int
7278:  Func XRTLSwapBits( Data :Card; Bit1Index, Bit2Index : TXRTLBitIndex) :Card
7279:  Func XRTLBitTest( Data :Card; BitIndex : TXRTLBitIndex) :Bool
7280:  Func XRTLBitSet( Data :Card; BitIndex : TXRTLBitIndex) :Card
7281:  Func XRTLBitReset( Data :Card; BitIndex : TXRTLBitIndex) :Card
7282:  Func XRTLBitComplement( Data :Card; BitIndex : TXRTLBitIndex) :Card
7283:  Func XRTLSwapHiLo16( X : Word) : Word
7284:  Func XRTLSwapHiLo32( X :Card) :Card
7285:  Func XRTLSwapHiLo64( X : Int64) : Int64
7286:  Func XRTLROL32( A, S :Card) :Card
7287:  Func XRTLROR32( A, S :Card) :Card
7288:  Func XRTLROL16( A : Word; S :Card) : Word
7289:  Func XRTLROR16( A : Word; S :Card) : Word
7290:  Func XRTLROL8( A : Byte; S :Card) : Byte
7291:  Func XRTLROR8( A : Byte; S :Card) : Byte
7292:  //Proc XRTLXorBlock( I1, I2, O1 : PByteArray; Len : Int)
7293:  //Proc XRTLIncBlock( P : PByteArray; Len : Int)
7294:  Proc XRTLUMul64( const A, B : Int; var MulL, MulH : Int)
7295:  Func XRTLPopulation( A :Card):Card
7296: end;
7297: Func XRTLURLDecode( const ASrc : WideString) : WideString
7298: Func XRTLURLEncode( const ASrc : WideString) : WideString
7299: Func XRTLURINormalize( const AURI : WideString) : WideString
7300: Proc XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword:WideString)
7301: Func XRTLExtractLongPathName(APath:Str):Str;
7302:
7303: Proc SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7304: begin
7305:   Int8', 'ShortInt
7306:   AddTypeS('Int16', 'SmallInt
7307:   Int32', 'LongInt
7308:   UInt8', 'Byte
```

```
7309:   UInt16', 'Word
7310:   UInt32', 'LongWord
7311:   UInt64', 'Int64
7312:   Word8', 'UInt8
7313:   Word16', 'UInt16
7314:   Word32', 'UInt32
7315:   Word64', 'UInt64
7316:   LargeInt', 'Int64
7317:   NativeInt', 'Int
7318:   AddTypeS('NativeUInt', 'Cardinal
7319:   Const('BitsPerByte','LongInt'( 8);
7320:   Const('BitsPerWord','LongInt'( 16);
7321:   Const('BitsPerLongWord','LongInt'( 32);
7322:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7323:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7324:  Func MinI( const A,B Int) : Int
7325:  Func MaxI( const A,B:Int) : Int
7326:  Func MinC( const A,B:Card) :Card
7327:  Func MaxC( const A,B:Card) :Card
7328:  Func SumClipI( const A, I : Int) : Int
7329:  Func SumClipC( const A :Card; const I : Int) :Card
7330:  Func InByteRange( const A : Int64) :Bool
7331:  Func InWordRange( const A : Int64) :Bool
7332:  Func InLongWordRange( const A : Int64): Bool
7333:  Func InShortIntRange( const A : Int64): Bool
7334:  Func InSmallIntRange( const A : Int64 : Bool
7335:  Func InLongIntRange( const A : Int64) : Bool
7336:   AddTypeS('Bool8', 'ByteBool
7337:   AddTypeS('Bool16', 'WordBool
7338:   AddTypeS('Bool32', 'LongBool
7339:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7340:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7341:  Func ReverseCompareResult( const C : TCompareResult) : TCompareResult
7342:  Const('MinSingle','Single').setExtended( 1.5E-45);
7343:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7344:  Const('MinDouble','Double').setExtended( 5.0E-324);
7345:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7346:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7347:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7348:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7349:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7350:  Func MinF( const A, B : Float) : Float
7351:  Func MaxF( const A, B : Float) : Float
7352:  Func ClipF( const Value : Float; const Low, High : Float) : Float
7353:  Func InSingleRange( const A : Float) :Bool
7354:  Func InDoubleRange( const A : Float) :Bool
7355:  Func InCurrencyRange( const A : Float) :Bool;
7356:  Func InCurrencyRange1( const A : Int64) :Bool;
7357:  Func FloatExponentBase2( const A : Extended; var Exponent : Int) :Bool
7358:  Func FloatExponentBase10( const A : Extended; var Exponent : Int):Bool
7359:  Func FloatIsInfinity( const A : Extended) :Bool
7360:  Func FloatIsNaN( const A : Extended) :Bool
7361:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7362:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7363:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7364:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7365:  Func FloatZero( const A : Float; const CompareDelta : Float) :Bool
7366:  Func FloatOne( const A : Float; const CompareDelta : Float) :Bool
7367:  Func FloatsEqual( const A, B : Float; const CompareDelta : Float) :Bool
7368:  Func FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7369:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7370:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7371:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7372:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7373:  Func ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) :Bool
7374:  Func ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7375:  Func cClearBit( const Value, BitIndex : LongWord) : LongWord
7376:  Func cSetBit( const Value, BitIndex : LongWord) : LongWord
7377:  Func cIsBitSet( const Value, BitIndex : LongWord) :Bool
7378:  Func cToggleBit( const Value, BitIndex : LongWord) : LongWord
7379:  Func cIsHighBitSet( const Value : LongWord) :Bool
7380:  Func SetBitScanForward( const Value : LongWord) : Int;
7381:  Func SetBitScanForward1( const Value, BitIndex : LongWord) : Int;
7382:  Func SetBitScanReverse( const Value : LongWord) : Int;
7383:  Func SetBitScanReverse1( const Value, BitIndex : LongWord) : Int;
7384:  Func ClearBitScanForward( const Value : LongWord) : Int;
7385:  Func ClearBitScanForward1( const Value, BitIndex : LongWord) : Int;
7386:  Func ClearBitScanReverse( const Value : LongWord) : Int;
7387:  Func ClearBitScanReverse1( const Value, BitIndex : LongWord) : Int;
7388:  Func cReverseBits( const Value : LongWord) : LongWord;
7389:  Func cReverseBits1( const Value : LongWord; const BitCount : Int) : LongWord;
7390:  Func cSwapEndian( const Value : LongWord) : LongWord
7391:  Func cTwosComplement( const Value : LongWord) : LongWord
7392:  Func RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7393:  Func RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7394:  Func RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7395:  Func RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7396:  Func cBitCount( const Value : LongWord) : LongWord
7397:  Func cIsPowerOfTwo( const Value : LongWord) :Bool
```

```
7398:  Func LowBitMask( const HighBitIndex : LongWord) : LongWord
7399:  Func HighBitMask( const LowBitIndex : LongWord) : LongWord
7400:  Func RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7401:  Func SetBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex: LongWord): LongWord
7402:  Func ClearBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord): LongWord
7403:  Func ToggleBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):LongWord
7404:  Func IsBitRangeSet(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):Bool
7405:  Func IsBitRangeClear(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):Boolean
7406: //  AddTypeS('CharSet', 'set of AnsiChar
7407:   AddTypeS('CharSet', 'set of Char       //!!!
7408:   AddTypeS('AnsiCharSet', 'TCharSet
7409:   AddTypeS('ByteSet', 'set of Byte
7410:   AddTypeS('AnsiChar', 'Char
7411:    // Func AsCharSet( const C : array of AnsiChar) : CharSet
7412:  Func AsByteSet( const C : array of Byte) : ByteSet
7413:  Proc ComplementChar( var C : CharSet; const Ch : Char)
7414:  Proc ClearCharSet( var C : CharSet)
7415:  Proc FillCharSet( var C : CharSet)
7416:  Proc FillCharSearchRec;  // with 0
7417:  Proc ComplementCharSet( var C : CharSet)
7418:  Proc AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7419:  Proc Union( var DestSet : CharSet; const SourceSet : CharSet)
7420:  Proc Difference( var DestSet : CharSet; const SourceSet : CharSet)
7421:  Proc Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7422:  Proc XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7423:  Func IsSubSet( const A, B : CharSet) :Bool
7424:  //Func IsEqual( const A, B : CharSet) :Bool
7425:  Func IsEqualCharset( const A, B : CharSet) :Bool
7426:  Func IsEqual( const D1, D2 : TDateTime) :Bool;
7427:  Func IsEmpty( const C : CharSet) :Bool
7428:  Func IsEmptyCharset( const C : CharSet) :Bool
7429:  Func IsComplete( const C : CharSet) :Bool
7430:  Func cCharCount( const C : CharSet) : Int
7431:  Proc ConvertCaseInsensitive( var C : CharSet)
7432:  Func CaseInsensitiveCharSet( const C : CharSet) : CharSet
7433:  Func IntRangeLength( const Low, High : Int) : Int64
7434:  Func IntRangeAdjacent( const Low1, High1, Low2, High2 : Int) :Bool
7435:  Func IntRangeOverlap( const Low1, High1, Low2, High2 : Int) :Bool
7436:  Func IntRangeHasElement( const Low, High, Element : Int) :Bool
7437:  Func IntRangeIncludeElement( var Low, High : Int; const Element : Int) :Bool
7438:  Func IntRangeIncludeElementRange(var Low,High:Int;const LowElement,HighElement:Int):Boolean
7439:  Func CardRangeLength( const Low, High :Card) : Int64
7440:  Func CardRangeAdjacent( const Low1, High1, Low2, High2 :Card) :Bool
7441:  Func CardRangeOverlap( const Low1, High1, Low2, High2 :Card) :Bool
7442:  Func CardRangeHasElement( const Low, High, Element :Card) :Bool
7443:  Func CardRangeIncludeElement( var Low, High :Card; const Element :Card) :Bool
7444:  Func CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElemnt:Card):Bool;
7445:   AddTypeS('UnicodeChar', 'WideChar
7446:  Func Compare( const I1, I2 :Bool) : TCompareResult;
7447:  Func Compare1( const I1, I2 : Int) : TCompareResult;
7448:  Func Compare2( const I1, I2 : Int64) : TCompareResult;
7449:  Func Compare3( const I1, I2 : Extended) :TCompareResult;
7450:  Func CompareA( const I1, I2 : Ansistr): TCompareResult
7451:  Func CompareW( const I1, I2 : WideString): TCompareResult
7452:  Func cSgn( const A : LongInt) : Int;
7453:  Func cSgn1( const A : Int64) : Int;
7454:  Func cSgn2( const A : Extended) : Int;
7455:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7456:  Func AnsiCharToInt( const A : AnsiChar) : Int
7457:  Func WideCharToInt( const A : WideChar) : Int
7458:  Func CharToInt( const A : Char) : Int
7459:  Func IntToAnsiChar( const A : Int) : AnsiChar
7460:  Func IntToWideChar( const A : Int) : WideChar
7461:  Func IntToChar( const A : Int) : Char
7462:  Func IsHexAnsiChar( const Ch : AnsiChar) :Bool
7463:  Func IsHexWideChar( const Ch : WideChar) :Bool
7464:  Func IsHexChar( const Ch : Char) :Bool
7465:  Func HexAnsiCharToInt( const A : AnsiChar) : Int
7466:  Func HexWideCharToInt( const A : WideChar) : Int
7467:  Func HexCharToInt( const A : Char) : Int
7468:  Func IntToUpperHexAnsiChar( const A : Int) : AnsiChar
7469:  Func IntToUpperHexWideChar( const A : Int) : WideChar
7470:  Func IntToUpperHexChar( const A : Int) : Char
7471:  Func IntToLowerHexAnsiChar( const A : Int) : AnsiChar
7472:  Func IntToLowerHexWideChar( const A : Int) : WideChar
7473:  Func IntToLowerHexChar( const A : Int) : Char
7474:  Func IntToStringA( const A : Int64) : Ansistr
7475:  Func IntToStringW( const A : Int64) : WideString
7476:  Func IntToString( const A : Int64) :Str
7477:  Func UIntToStringA( const A : NativeUInt) : Ansistr
7478:  Func UIntToStringW( const A : NativeUInt) : WideString
7479:  Func UIntToString( const A : NativeUInt) :Str
7480:  Func LongWordToStrA( const A : LongWord; const Digits : Int) : Ansistr
7481:  Func LongWordToStrW( const A : LongWord; const Digits : Int) : WideString
7482:  Func LongWordToStrU( const A : LongWord; const Digits : Int) : UnicodeString
7483:  Func LongWordToStr( const A : LongWord; const Digits : Int) :Str
7484:  Func LongWordToHexA(const A:LongWord;const Digits:Int;const UpperCase:Bool):Ansistr;
7485:  Func LongWordToHexW(const A:LongWord;const Digits:Int;const UpperCase:Bool):WideString;
7486:  Func LongWordToHex( const A : LongWord; const Digits : Int;const UpperCase:Boolean):str
```

```
7487:  Func LongWordToOctA( const A : LongWord; const Digits : Int) : Ansistr
7488:  Func LongWordToOctW( const A : LongWord; const Digits : Int) : WideString
7489:  Func LongWordToOct( const A : LongWord; const Digits : Int) :Str
7490:  Func LongWordToBinA( const A : LongWord; const Digits : Int) : Ansistr
7491:  Func LongWordToBinW( const A : LongWord; const Digits : Int) : WideString
7492:  Func LongWordToBin( const A : LongWord; const Digits : Int) :Str
7493:  Func TryStringToInt64A( const S : Ansistr; out A : Int64) :Bool
7494:  Func TryStringToInt64W( const S : WideString; out A : Int64) :Bool
7495:  Func TryStringToInt64( const S :Str; out A : Int64) :Bool
7496:  Func StringToInt64DefA( const S : Ansistr; const Default : Int64): Int64
7497:  Func StringToInt64DefW( const S : WideString; const Default : Int64): Int64
7498:  Func StringToInt64Def( const S :Str; const Default : Int64) : Int64
7499:  Func StringToInt64A( const S : Ansistr) : Int64
7500:  Func StringToInt64W( const S : WideString) : Int64
7501:  Func StringToInt64( const S :Str) : Int64
7502:  Func TryStringToIntA( const S : Ansistr; out A : Int) :Bool
7503:  Func TryStringToIntW( const S : WideString; out A : Int) :Bool
7504:  Func TryStringToInt( const S :Str; out A : Int) :Bool
7505:  Func StringToIntDefA( const S : Ansistr; const Default : Int) : Int
7506:  Func StringToIntDefW( const S : WideString; const Default : Int) : Int
7507:  Func StringToIntDef( const S :Str; const Default : Int) : Int
7508:  Func StringToIntA( const S : Ansistr) : Int
7509:  Func StringToIntW( const S : WideString) : Int
7510:  Func StringToInt( const S :Str) : Int
7511:  Func TryStringToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7512:  Func TryStringToLongWordW( const S : WideString; out A : LongWord) :Bool
7513:  Func TryStringToLongWord( const S :Str; out A : LongWord) :Bool
7514:  Func StringToLongWordA( const S : Ansistr) : LongWord
7515:  Func StringToLongWordW( const S : WideString) : LongWord
7516:  Func StringToLongWord( const S :Str) : LongWord
7517:  Func HexToUIntA( const S : Ansistr) : NativeUInt
7518:  Func HexToUIntW( const S : WideString) : NativeUInt
7519:  Func HexToUInt( const S :Str) : NativeUInt
7520:  Func TryHexToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7521:  Func TryHexToLongWordW( const S : WideString; out A : LongWord) :Bool
7522:  Func TryHexToLongWord( const S :Str; out A : LongWord) :Bool
7523:  Func HexToLongWordA( const S : Ansistr) : LongWord
7524:  Func HexToLongWordW( const S : WideString) : LongWord
7525:  Func HexToLongWord( const S :Str) : LongWord
7526:  Func TryOctToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7527:  Func TryOctToLongWordW( const S : WideString; out A : LongWord) :Bool
7528:  Func TryOctToLongWord( const S :Str; out A : LongWord) :Bool
7529:  Func OctToLongWordA( const S : Ansistr) : LongWord
7530:  Func OctToLongWordW( const S : WideString) : LongWord
7531:  Func OctToLongWord( const S :Str) : LongWord
7532:  Func TryBinToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7533:  Func TryBinToLongWordW( const S : WideString; out A : LongWord) :Bool
7534:  Func TryBinToLongWord( const S :Str; out A : LongWord) :Bool
7535:  Func BinToLongWordA( const S : Ansistr) : LongWord
7536:  Func BinToLongWordW( const S : WideString) : LongWord
7537:  Func BinToLongWord( const S :Str) : LongWord
7538:  Func FloatToStringA( const A : Extended) : Ansistr
7539:  Func FloatToStringW( const A : Extended) : WideString
7540:  Func FloatToString( const A : Extended) :Str
7541:  Func FloatToStringA( const A : Extended) :Str
7542:  Func TryStringToFloatA( const A : Ansistr; out B : Extended) :Bool
7543:  Func TryStringToFloatW( const A : WideString; out B : Extended) :Bool
7544:  Func TryStringToFloat( const A :Str; out B : Extended) :Bool
7545:  Func StringToFloatA( const A : Ansistr) : Extended
7546:  Func StringToFloatW( const A : WideString) : Extended
7547:  Func StringToFloat( const A :Str) : Extended
7548:  Func StringToFloatDefA( const A : Ansistr; const Default : Extended) : Extended
7549:  Func StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7550:  Func StringToFloatDef( const A :Str; const Default : Extended) : Extended
7551:  Func EncodeBase64(const S,Alphabet:Ansistr;const Pad:Boolean;const PadMultiple:Int;const
     PadChar:AnsiChar):Ansistr
7552:  Func DecodeBase64( const S, Alphabet:Ansistr;const PadSet:CharSet):Ansistr
7553:  unit uPSI_cFundamentUtils;
7554:  Const('b64_MIMEBase64',Str).String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7555:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7556:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7557:  Const('CCHARSET','Stringb64_XXEncode);
7558:  Const('CHEXSET','String'0123456789ABCDEF
7559:  Const('HEXDIGITS','String'0123456789ABCDEF
7560:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7561:  Const('DIGISET','String'0123456789
7562:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7563:  Const('DIGISET2','TCharset').SetSet('0123456789'
7564:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7565:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF;
7566:  Const('NUMBERSET','TCharset').SetSet('0123456789;
7567:  Const('NUMBERS','String'0123456789;
7568:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ;
7569:  Const('NUMBLETTS','String').SetString('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ;
7570:  Const('NUMBLETTSET','TCharset').SetSet('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ;
7571:  Func CharSetToStr( const C : CharSet) : Ansistr
7572:  Func StrToCharSet( const S : Ansistr) : CharSet
7573:  Func MIMEBase64Decode( const S : Ansistr) : Ansistr
7574:  Func MIMEBase64Encode( const S : Ansistr) : Ansistr
```

```
7575:  Func UUDecode( const S: Ansistr) : Ansistr
7576:  Func XXDecode( const S  Ansistr) : Ansistr
7577:  Func BytesToHex( const P : array of Byte; const UpperCase :Bool) : Ansistr
7578:  Func InterfaceToStrA( const I : IInterface) : Ansistr
7579:  Func InterfaceToStrW( const I : IInterface) : WideString
7580:  Func InterfaceToStr( const I : IInterface) :Str
7581:  Func ObjectClassName( const O : TObject) :Str
7582:  Func ClassClassName( const C : TClass) :Str
7583:  Func ObjectToStr( const O : TObject) :Str
7584:  Func ObjectToString( const O : TObject) :Str
7585:  Func CharSetToStr( const C : CharSet) : Ansistr
7586:  Func StrToCharSet( const S : Ansistr) : CharSet
7587:  Func HashStrA(const S : Ansistr; const Index : Int; const Count: Int;const AsciiCaseSensitive :Bool;
      const Slots : LongWord) : LongWord
7588:  Func HashStrW(const S:WideString;const Index:Int;const Count:Int;const AsciiCaseSensitive:Bool;const
      Slots:LongWord):LongWord
7589:  Func HashStr(const S:str;const Index:Int;const Count:Int;const AsciiCaseSensitive:Bool;const Slots:
      LongWord):LongWord
7590:  Func HashInt( const I : Int; const Slots : LongWord) : LongWord
7591:  Func HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7592:  Const('Bytes1KB','LongInt'( 1024);
7593:   SIRegister_IInterface(CL);
7594:  Proc SelfTestCFundamentUtils
7595:
7596: Func CreateSchedule : IJclSchedule
7597: Func NullStamp : TTimeStamp
7598: Func CompareTimeStamps(const Stamp1, Stamp2 : TTimeStamp) : Int64
7599: Func EqualTimeStamps(const Stamp1, Stamp2 : TTimeStamp) :Bool
7600: Func IsNullTimeStamp(const Stamp : TTimeStamp) :Bool
7601:
7602: Proc SIRegister_uwinplot(CL: TPSPascalCompiler);
7603: begin
7604:  AddTypeS(TFunc', 'function(X : Float) : Float;
7605: Func InitGraphics( Width, Height : Int) :Bool
7606: Proc SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Int; GraphBorder :Bool)
7607: Proc SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7608: Proc SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7609: Proc GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7610: Proc GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7611: Proc SetGraphTitle( Title :Str)
7612: Proc SetOxTitle( Title :Str)
7613: Proc SetOyTitle( Title :Str)
7614: Func GetGraphTitle :Str
7615: Func GetOxTitle :Str
7616: Func GetOyTitle :Str
7617: Proc PlotOxAxis( Canvas : TCanvas)
7618: Proc PlotOyAxis( Canvas : TCanvas)
7619: Proc PlotGrid( Canvas : TCanvas; Grid : TGrid)
7620: Proc WriteGraphTitle( Canvas : TCanvas)
7621: Func SetMaxCurv( NCurv : Byte) :Bool
7622: Proc SetPointParam( CurvIndex, Symbol, Size : Int; Color : TColor)
7623: Proc SetLineParam(CurvIndex: Int; Style : TPenStyle; Width : Int; Color : TColor)
7624: Proc SetCurvLegend( CurvIndex : Int; Legend :Str)
7625: Proc SetCurvStep( CurvIndex, Step : Int)
7626: Func GetMaxCurv : Byte
7627: Proc GetPointParam( CurvIndex : Int; var Symbol, Size : Int; var Color : TColor)
7628: Proc GetLineParam(CurvIndex:Int;var Style:TPenStyle;var Width:Int;var Color:TColor);
7629: Func GetCurvLegend( CurvIndex : Int) :Str
7630: Func GetCurvStep( CurvIndex : Int) : Int
7631: Proc PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Int)
7632: Proc PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Int)
7633: Proc PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Int)
7634: Proc PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Int)
7635: Proc WriteLegend( Canvas : TCanvas; NCurv : Int; ShowPoints, ShowLines :Bool)
7636: Proc ConRec( Canvas : TCanvas; Nx, Ny, Nc : Int; X, Y, Z : TVector; F : TMatrix)
7637: Func Xpixel(X:Float) : Int
7638: Func Ypixel(Y:Float) : Int
7639: Func Xuser( X:Int) : Float
7640: Func Yuser( Y:Int) : Float
7641: end;
7642:
7643: Proc FFT( NumSamples : Int; InArray, OutArray : TCompVector)
7644: Proc IFFT( NumSamples : Int; InArray, OutArray : TCompVector)
7645: Proc FFT_Int( NumSamples : Int; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7646: Proc FFT_Int_Cleanup
7647: Proc CalcFrequency(NumSamples,FrequencyIndex: Int;InArray: TCompVector;var FT : Complex)
7648: //unit uPSI_JclStreams;
7649: Func StreamSeek(Stream: TStream; const Offset: Int64; const Origin : TSeekOrigin) : Int64
7650: Func StreamCopy( Source : TStream; Dest : TStream; BufferSize : Int) : Int64
7651: Func CompareStreams( A, B : TStream; BufferSize : Int) :Bool
7652: Func JCompareFiles( const FileA, FileB : TFileName; BufferSize : Int) :Bool
7653:
7654: Proc SIRegister_FmxUtils(CL: TPSPascalCompiler);
7655: begin
7656:  FindClass('TOBJECT'),'EInvalidDest
7657:  FindClass('TOBJECT'),'EFCantMove
7658:  Proc fmxCopyFile( const FileName, DestName :Str)
7659:  Proc fmxMoveFile( const FileName, DestName :Str)
7660:  Func fmxGetFileSize( const FileName :Str) : LongInt
```

```
7661:  Func fmxFileDateTime( const FileName :Str) : TDateTime
7662:  Func fmxHasAttr( const FileName :Str; Attr : Word) :Bool
7663:  Func fmxExecuteFile( const FileName, Params, DefaultDir:Str; ShowCmd : Int):THandle;
7664: end;
7665:
7666: Proc SIRegister_FindFileIter(CL: TPSPascalCompiler);
7667: begin
7668:  SIRegister_IFindFileIterator(CL);
7669:  Func CreateFindFile(const Path:str; IncludeAttr:Int;out iffi:IFindFileIterator):Bool;
7670: end;
7671:
7672: Proc SIRegister_PCharUtils(CL: TPSPascalCompiler);
7673: begin
7674:  Func SkipWhite( cp : PChar) : PChar
7675:  Func ReadStringDoubleQuotedMaybe( cp : PChar; var AStr :Str) : PChar
7676:  Func ReadStringSingleQuotedMaybe( cp : PChar; var AStr :Str) : PChar
7677:  Func ReadIdent( cp : PChar; var ident :Str) : PChar
7678: end;
7679:
7680: Proc SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7681: begin
7682:   SIRegister_TStringHashMapTraits(CL);
7683:  Func CaseSensitiveTraits : TStringHashMapTraits
7684:  Func CaseInsensitiveTraits : TStringHashMapTraits
7685:   THashNode','record Str:str;Ptr:Pointer; Left:PHashNode;Right:PHashNode;end
7686:   //PHashArray', '^THashArray // will not work
7687:   SIRegister_TStringHashMap(CL);
7688:  THashValue', 'Cardinal
7689: Func StrHash( const s :Str) : THashValue
7690: Func TextHash( const s :Str) : THashValue
7691: Func DataHash( var AValue, ASize :Card) : THashValue
7692: Func Iterate_FreeObjects(AUserData:Pointer;const AStr:str; var AData:Pointer):Boolean
7693: Func Iterate_Dispose(AUserData: Pointer;const AStr:str; var AData: Pointer):Bool
7694: Func Iterate_FreeMem(AUserData: Pointer;const AStr:str; var AData  Pointer):Bool
7695:  SIRegister_TCaseSensitiveTraits(CL);
7696:  SIRegister_TCaseInsensitiveTraits(CL);
7697: //*************************************************************unit uPSI_umath;
7698:  Func uExpo( X : Float) : Float
7699:  Func uExp2( X : Float) : Float
7700:  Func uExp10( X : Float) : Float
7701:  Func uLog( X : Float) : Float
7702:  Func uLog2( X : Float) : Float
7703:  Func uLog10( X : Float) : Float
7704:  Func uLogA( X, A : Float) : Float
7705:  Func uIntPower( X : Float; N : Int): Float
7706:  Func uPower( X, Y : Float) : Float
7707:  Func SgnGamma( X : Float) : Int
7708:  Func Stirling( X : Float) : Float
7709:  Func StirLog( X : Float) : Float
7710:  Func Gamma( X : Float) : Float
7711:  Func LnGamma( X : Float) : Float
7712:  Func DiGamma( X : Float) : Float
7713:  Func TriGamma( X : Float) : Float
7714:  Func IGamma( X : Float) : Float
7715:  Func JGamma( X : Float) : Float
7716:  Func InvGamma( X : Float) : Float
7717:  Func Erf( X : Float) : Float
7718:  Func Erfc( X : Float) : Float
7719: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float;
7720: { Correlation coefficient between samples X and Y }
7721: Func DBeta(A, B, X : Float) : Float;
7722: { Density of Beta distribution with parameters A and B }
7723: Func LambertW( X : Float; UBranch, Offset :Bool) : Float
7724: Func Beta(X, Y : Float) : Float
7725: Func Binomial( N, K : Int) : Float
7726: Func PBinom( N : Int; P : Float; K : Int) : Float
7727: Proc Cholesky( A, L : TMatrix; Lb, Ub : Int)
7728: Proc LU_Decomp( A : TMatrix; Lb, Ub : Int)
7729: Proc LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Int; X : TVector)
7730: Func DNorm( X : Float) : Float
7731: Func DGamma(A, B, X : Float) : Float;
7732: { Density of Gamma distribution with parameters A and B }
7733: Func DKhi2(Nu : Int; X : Float) : Float;
7734: { Density of Khi-2 distribution with Nu d.o.f. }
7735: Func DStudent(Nu : Int; X : Float) : Float;
7736: { Density of Student distribution with Nu d.o.f. }
7737: Func DSnedecor(Nu1, Nu2 : Int; X : Float) : Float;
7738: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7739: Func IBeta(A, B, X : Float) : Float;
7740: { Incomplete Beta function}
7741: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float;
7742:
7743: Proc SIRegister_unlfit(CL: TPSPascalCompiler);
7744: begin Proc SetOptAlgo( Algo : TOptAlgo)
7745: Proc SetOptAlgo(Algo : TOptAlgo);
7746: { ----------------------------------------------------------------
7747:   Sets the optimization algorithm according to Algo, which must be
7748:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7749:  Func GetOptAlgo : TOptAlgo
```

```
7750:  Proc SetMaxParam( N : Byte)
7751:  Func GetMaxParam : Byte
7752:  Proc SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7753:  Proc GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7754:  Func NullParam( B : TVector; Lb, Ub : Int) :Bool
7755:  Proc NLFit(RegFunc:TRegFunc;DerivProc:TDerivProc;X,Y:TVector;Lb Ub:Int;MaxIter:Int;Tol:
      Float;B:TVector;FirstPar,LastPar:Int;V:TMatrix)
7756:  Proc WNLFit(RegFunc:TRegFunc;DerivProc:TDerivProc;X,Y,S:TVector;Lb,Ub:Int;MaxIter:Int;Tol
      :Float;B:TVector;FirstPar,LastPar:Int;V:TMatrix)
7757:  Proc SetMCFile( FileName :Str)
7758:  Proc SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Int;B:TVector;FirstPar,LastPar:Int;V:TMatrix;
7759:  Proc WSimFit(RegFunc:TRegFunc;X,Y,S:TVector;Lb,Ub:Int;B:TVector;FirstPar,LastPar:Int;V:TMatrix);
7760: end;
7761:
7762: (*----------------------------------------------------------------------------*)
7763: Proc SIRegister_usimplex(CL: TPSPascalCompiler);
7764: begin
7765:  Proc SaveSimplex( FileName :Str)
7766:  Proc Simplex(Func:TFuncNVar;X:TVector;Lb,Ub:Int; MaxIter:Int;Tol:Float;var F_min:Float);
7767: end;
7768: (*----------------------------------------------------------------------------*)
7769: Proc SIRegister_uregtest(CL: TPSPascalCompiler);
7770: begin
7771:  Proc RegTest(Y,Ycalc:TVector;LbY,UbY:Int;V:TMatrix;LbV,UbV:Int;var Test:TRegTest)
7772:  Proc WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Int;V:TMatrix;LbV,UbV:Int;var Test:TRegTest);
7773: end;
7774:
7775: Proc SIRegister_ustrings(CL: TPSPascalCompiler);
7776: begin
7777:  Func LTrim( S:Str) :Str
7778:  Func RTrim( S:Str) :Str
7779:  Func uTrim( S:Str) :Str
7780:  Func StrChar(N : Byte; C : Char) :Str
7781:  Func RFill( S:Str; L : Byte) :Str
7782:  Func LFill( S:Str; L : Byte) :Str
7783:  Func CFill( S:Str; L : Byte) :Str
7784:  Func Replace(S :Str; C1, C2 : Char) :Str
7785:  Func Extract(S :Str; var Index : Byte; Delim : Char) :Str
7786:  Proc Parse( S :Str; Delim : Char; Field : TStrVector; var N : Byte)
7787:  Proc SetFormat( NumLength, MaxDec : Int; FloatPoint, NSZero :Bool)
7788:  Func FloatStr( X : Float) :Str
7789:  Func IntStr( N : LongInt) :Str
7790:  Func uCompStr( Z : Complex) :Str
7791: end;
7792:
7793: Proc SIRegister_uhyper(CL: TPSPascalCompiler);
7794: begin
7795:  Func uSinh( X : Float) : Float
7796:  Func uCosh( X : Float) : Float
7797:  Func uTanh( X : Float) : Float
7798:  Func uArcSinh( X : Float) : Float
7799:  Func uArcCosh( X : Float) : Float
7800:  Func ArcTanh( X : Float) : Float
7801:  Proc SinhCosh( X : Float; var SinhX, CoshX : Float)
7802: end;
7803:
7804: Proc SIRegister_urandom(CL: TPSPascalCompiler);
7805: begin
7806: type RNG_Type =
7807:   (RNG_MWC,       { Multiply-With-Carry }
7808:    RNG_MT,        { Mersenne Twister }
7809:    RNG_UVAG);     { Universal Virtual Array Generator }
7810:  Proc SetRNG( RNG : RNG_Type)
7811:  Proc InitGen( Seed : RNG_IntType)
7812:  Proc SRand( Seed : RNG_IntType)
7813:  Func IRanGen : RNG_IntType
7814:  Func IRanGen31 : RNG_IntType
7815:  Func RanGen1 : Float
7816:  Func RanGen2 : Float
7817:  Func RanGen3 : Float
7818:  Func RanGen53 : Float
7819: end;
7820:
7821: //  Optimization by Simulated Annealing
7822: Proc SIRegister_usimann(CL: TPSPascalCompiler);
7823: begin
7824:  Proc InitSAParams( NT, NS, NCycles : Int; RT : Float)
7825:  Proc SA_CreateLogFile( FileName :Str)
7826:  Proc SimAnn(Func: TFuncNVar; X,Xmin,Xmax: TVector; Lb, Ub : Int; var F_min : Float);
7827: end;
7828:
7829: Proc SIRegister_uranuvag(CL: TPSPascalCompiler);
7830: begin
7831:  Proc InitUVAGbyString( KeyPhrase :Str)
7832:  Proc InitUVAG( Seed : RNG_IntType)
7833:  Func IRanUVAG : RNG_IntType
7834: end;
7835:
7836: Proc SIRegister_ugenalg(CL: TPSPascalCompiler);
```

```
7837: begin
7838:  Proc InitGAParams( NP, NG : Int; SR, MR, HR : Float)
7839:  Proc GA_CreateLogFile( LogFileName :Str)
7840:  Proc GenAlg(Func: TFuncNVar; X,Xmin,Xmax : TVector; Lb, Ub : Int; var F_min : Float);
7841: end;
7842:
7843:  TVector', 'array of Float
7844: Proc SIRegister_uqsort(CL: TPSPascalCompiler);
7845: begin
7846:  Proc QSort( X : TVector; Lb, Ub : Int)
7847:  Proc DQSort( X : TVector; Lb, Ub : Int)
7848: end;
7849:
7850: Proc SIRegister_uinterv(CL: TPSPascalCompiler);
7851: begin
7852:  Proc Interval( X1, X2 : Float; MinDiv, MaxDiv : Int; var Min, Max, Step : Float)
7853:  Proc AutoScale(X: TVector; Lb, Ub : Int; Scale : TScale; var XMin, XMax, XStep:Float)
7854: end;
7855:
7856: Proc SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7857: begin
7858:   FT_Result', 'Int
7859:   //TDWordptr', '^DWord // will not work
7860:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7861:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7862:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7863:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7864:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7865:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7866:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7867:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7868:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7869:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7870:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7871:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7872:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7873:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7874:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7875:    yte; end
7876: end;
7877:
7878: //************************************** PaintFX****************************
7879: Proc SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7880: begin
7881: //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7882: with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7883:   Proc Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Int)
7884:   Proc Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Int)
7885:   Proc Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7886:   Proc Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount: Single)
7887:   Proc ExtractColor( const Dst : TBitmap; AColor : TColor)
7888:   Proc ExcludeColor( const Dst : TBitmap; AColor : TColor)
7889:   Proc Turn( Src, Dst : TBitmap)
7890:   Proc TurnRight( Src, Dst : TBitmap)
7891:   Proc HeightMap( const Dst : TBitmap; Amount : Int)
7892:   Proc TexturizeTile( const Dst : TBitmap; Amount : Int)
7893:   Proc TexturizeOverlap( const Dst : TBitmap; Amount : Int)
7894:   Proc RippleRandom( const Dst : TBitmap; Amount : Int)
7895:   Proc RippleTooth( const Dst : TBitmap; Amount : Int)
7896:   Proc RippleTriangle( const Dst : TBitmap; Amount : Int)
7897:   Proc Triangles( const Dst : TBitmap; Amount : Int)
7898:   Proc DrawMandelJulia(const Dst:TBitmap;x0,y0,x1,y1:Single;Niter:Int;Mandel:Bool)
7899:   Proc FilterXBlue( const Dst : TBitmap; Min, Max : Int)
7900:   Proc FilterXGreen( const Dst : TBitmap; Min, Max : Int)
7901:   Proc FilterXRed( const Dst : TBitmap; Min, Max : Int)
7902:   Proc FilterBlue( const Dst : TBitmap; Min, Max : Int)
7903:   Proc FilterGreen( const Dst : TBitmap; Min, Max : Int)
7904:   Proc FilterRed( const Dst : TBitmap; Min, Max : Int)
7905:   Proc Emboss( var Bmp : TBitmap)
7906:   Proc Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7907:   Proc Shake( Src, Dst : TBitmap; Factor : Single)
7908:   Proc ShakeDown( Src, Dst : TBitmap; Factor : Single)
7909:   Proc KeepBlue( const Dst : TBitmap; Factor : Single)
7910:   Proc KeepGreen( const Dst : TBitmap; Factor : Single)
7911:   Proc KeepRed( const Dst : TBitmap; Factor : Single)
7912:   Proc Mandelbrot( const Dst : TBitmap; Factor : Int)
7913:   Proc MaskMandelbrot( const Dst : TBitmap; Factor : Int)
7914:   Proc FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7915:   Proc QuartoOpaque( Src, Dst : TBitmap)
7916:   Proc SemiOpaque( Src, Dst : TBitmap)
7917:   Proc ShadowDownLeft( const Dst : TBitmap)
7918:   Proc ShadowDownRight( const Dst : TBitmap)
7919:   Proc ShadowUpLeft( const Dst : TBitmap)
7920:   Proc ShadowUpRight( const Dst : TBitmap)
7921:   Proc Darkness( const Dst : TBitmap; Amount : Int)
7922:   Proc Trace( const Dst : TBitmap; Intensity : Int)
7923:   Proc FlipRight( const Dst : TBitmap)
7924:   Proc FlipDown( const Dst : TBitmap)
7925:   Proc SpotLight( const Dst : TBitmap; Amount : Int; Spot : TRect)
```

```
7926:   Proc SplitLight( const Dst : TBitmap; Amount : Int)
7927:   Proc MakeSeamlessClip( var Dst : TBitmap; Seam : Int)
7928:   Proc Wave( const Dst : TBitmap; Amount, Inference, Style : Int)
7929:   Proc Mosaic( const Bm : TBitmap; Size : Int)
7930:   Proc SmoothRotate( var Src, Dst : TBitmap; CX, CY : Int; Angle : Single)
7931:   Proc SmoothResize( var Src, Dst : TBitmap)
7932:   Proc Twist( var Bmp, Dst : TBitmap; Amount : Int)
7933:   Proc SplitBlur( const Dst : TBitmap; Amount : Int)
7934:   Proc GaussianBlur( const Dst : TBitmap; Amount : Int)
7935:   Proc Smooth( const Dst : TBitmap; Weight : Int)
7936:   Proc GrayScale( const Dst : TBitmap)
7937:   Proc AddColorNoise( const Dst : TBitmap; Amount : Int)
7938:   Proc AddMonoNoise( const Dst : TBitmap; Amount : Int)
7939:   Proc Contrast( const Dst : TBitmap; Amount : Int)
7940:   Proc Lightness( const Dst : TBitmap; Amount : Int)
7941:   Proc Saturation( const Dst : TBitmap; Amount : Int)
7942:   Proc Spray( const Dst : TBitmap; Amount : Int)
7943:   Proc AntiAlias( const Dst : TBitmap)
7944:   Proc AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Int)
7945:   Proc SmoothPoint( const Dst : TBitmap; XK, YK : Int)
7946:   Proc FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7947:   Proc Marble(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7948:   Proc Marble2(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7949:   Proc Marble3(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7950:   Proc Marble4(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7951:   Proc Marble5(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7952:   Proc Marble6(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7953:   Proc Marble7(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7954:   Proc Marble8(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7955:   Proc SqueezeHor( Src, Dst : TBitmap; Amount : Int; Style : TLightBrush)
7956:   Proc SplitRound( Src, Dst : TBitmap; Amount : Int; Style : TLightBrush)
7957:   Proc Tile( Src, Dst : TBitmap; Amount : Int)
7958:   Proc Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7959:   Proc Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Int)
7960:   Proc Invert( Src : TBitmap)
7961:   Proc MirrorRight( Src : TBitmap)
7962:   Proc MirrorDown( Src : TBitmap)
7963: end;
7964: end;
7965:
7966: (*----------------------------------------------------------------------------*)
7967: Proc SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7968: begin
7969:   AddTypeS('TLightBrush','( lbBrightness, lbContrast, lbSaturation, lbFishe'
7970:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7971:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7972:   SIRegister_TJvPaintFX(CL);
7973:  Func SplineFilter( Value : Single) : Single
7974:  Func BellFilter( Value : Single) : Single
7975:  Func TriangleFilter( Value : Single) : Single
7976:  Func BoxFilter( Value : Single) : Single
7977:  Func HermiteFilter( Value : Single) : Single
7978:  Func Lanczos3Filter( Value : Single) : Single
7979:  Func MitchellFilter( Value : Single) : Single
7980: end;
7981:
7982: (*----------------------------------------------------------------------------*)
7983: Proc SIRegister_Chart(CL: TPSPascalCompiler);
7984: begin
7985:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7986:  TeeMsg_DefaultSeriesName','String 'Series
7987:  TeeMsg_DefaultToolName','String 'ChartTool
7988:  ChartComponentPalette','String 'TeeChart
7989:  TeeMaxLegendColumns',LongInt'( 2);
7990:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7991:  TeeTitleFootDistance,LongInt( 5);
7992:   SIRegister_TCustomChartWall(CL);
7993:   SIRegister_TChartWall(CL);
7994:   SIRegister_TChartLegendGradient(CL);
7995:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7996:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7997:   FindClass('TOBJECT'),'LegendException
7998:   TOnGetLegendText', 'Proc ( Sender : TCustomAxisPanel; Legen'
7999:    +'dStyle : TLegendStyle; Index : Int; var LegendText :Str)
8000:   FindClass('TOBJECT'),'TCustomChartLegend
8001:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
8002:   TLegendSymbolPosition', '( spLeft, spRight )
8003:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Int;R:TRect);
8004:   TSymbolCalcHeight', 'Func  : Int
8005:   SIRegister_TLegendSymbol(CL);
8006:   SIRegister_TTeeCustomShapePosition(CL);
8007:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
8008:   SIRegister_TLegendTitle(CL);
8009:   SIRegister_TLegendItem(CL);
8010:   SIRegister_TLegendItems(CL);
8011:   TLegendCalcSize', 'Proc ( Sender : TCustomChartLegend; var ASize : Int)
8012:   FindClass('TOBJECT'),'TCustomChart
8013:   SIRegister_TCustomChartLegend(CL);
8014:   SIRegister_TChartLegend(CL);
```

```
8015:   SIRegister_TChartTitle(CL);
8016:   SIRegister_TChartFootTitle(CL);
8017:   TChartClick', 'Procedure(ender:TCustomChart;Button:TMouseButton;Shift:TShiftState;X,Y:Int)
8018:   TChartClickAxis', 'Proc ( Sender : TCustomChart; Axis : TCh'
8019:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Int)
8020:   TChartClickSeries', 'Proc ( Sender : TCustomChart; Series :'
8021:    +TChartSeries; ValueIndex : Int; Button: TMouseButton; Shift:TShiftState;X,Y:Int)
8022:   TChartClickTitle', 'Proc ( Sender : TCustomChart; ATitle : '
8023:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Int)
8024:   TOnGetLegendPos', 'Proc (Sender: TCustomChart; Index: Int; var X,Y,XColor:Int)
8025:   TOnGetLegendRect', 'Proc ( Sender : TCustomChart; var Rect : TRect)
8026:   TAxisSavedScales', 'record Auto:Boolean;AutoMin:Bool;AutoMax:Bool;Min:Double;Max:Double;end
8027:   TAllAxisSavedScales', 'array of TAxisSavedScales
8028:   SIRegister_TChartBackWall(CL);
8029:   SIRegister_TChartRightWall(CL);
8030:   SIRegister_TChartBottomWall(CL);
8031:   SIRegister_TChartLeftWall(CL);
8032:   SIRegister_TChartWalls(CL);
8033:   TChartAllowScrollEvent,Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Bool);
8034:   SIRegister_TCustomChart(CL);
8035:   SIRegister_TChart(CL);
8036:   SIRegister_TTeeSeriesTypes(CL);
8037:   SIRegister_TTeeToolTypes(CL);
8038:   SIRegister_TTeeDragObject(CL);
8039:   SIRegister_TColorPalettes(CL);
8040:  Proc RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Int;
8041:  Proc RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
8042:  Proc RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Int;
8043:  Proc RegisterTeeBasicFunction(AFunctionClass: TTeeFunctionClass; ADescription : PString)
8044:  Proc RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription,AGalleryPage:PString;ANumGallerySeries:Int;ASubIndex : Int)
8045:  Proc UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
8046:  Proc UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
8047:  Proc AssignSeries( var OldSeries, NewSeries : TChartSeries)
8048:  Func CreateNewTeeFunction(ASeries:TChartSeries;AClass:TTeeFunctionClass): TTeeFunction
8049:  Func CreateNewSeries( AOwner: TComponent; AChar :TCustomAxisPanel;AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
8050:  Func CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
8051:  Func CloneChartSeries1(ASeries: TChartSeries; AChart: TCustomAxisPanel): TChartSeries;
8052:  Func CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
8053:  Func CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
8054:  Func ChangeSeriesType(var ASeries:TChartSeries;NewType:TChartSeriesClass) : TChartSeries
8055:  Proc ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
8056:  Func GetNewSeriesName( AOwner : TComponent) : TComponentName
8057:  Proc RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
8058:  Proc UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
8059:  Func GetGallerySeriesName( ASeries : TChartSeries) :Str
8060:  Proc PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const R:TRect;ReferenceChart:TCustomChart);
8061:   SIRegister_TChartTheme(CL); //TChartThemeClass', 'class of TChartTheme
8062:   //TCanvasClass', 'class of TCanvas3D
8063:  Func SeriesNameOrIndex( ASeries : TCustomChartSeries) :Str
8064:  Func SeriesTitleOrName( ASeries : TCustomChartSeries) :Str
8065:  Proc FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles :Bool)
8066:  Proc ShowMessageUser( const S :Str)
8067:  Func HasNoMandatoryValues( ASeries : TChartSeries) :Bool
8068:  Func HasLabels( ASeries : TChartSeries):Bool
8069:  Func HasColors( ASeries : TChartSeries):Bool
8070:  Func SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
8071:  Proc TeeDrawBitmapEditor(Canvas: TCanvas;Element: TCustomChartElement;Left,Top:Int)
8072: end;
8073:
8074: Proc SIRegister_TeeProcs(CL: TPSPascalCompiler);
8075: begin
8076:  //'TeeFormBorderStyle',' bsNone);
8077:   SIRegister_TMetafile(CL);
8078:  'TeeDefVerticalMargin','LongInt'( 4);
8079:  'TeeDefHorizMargin','LongInt'( 3);
8080:  'crTeeHand','LongInt'( TCursor ( 2020 ));
8081:  'TeeMsg_TeeHand','String 'crTeeHand
8082:  'TeeNormalPrintDetail','LongInt'( 0);
8083:  'TeeHighPrintDetail','LongInt'( - 100);
8084:  'TeeDefault_PrintMargin','LongInt'( 15);
8085:  'MaxDefaultColors','LongInt'( 19);
8086:  'TeeTabDelimiter',Char #9);
8087:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
8088:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
8089:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
8090:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
8091:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
8092:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
8093:   SIRegister_TCustomPanelNoCaption(CL);
8094:   FindClass('TOBJECT'),'TCustomTeePanel
8095:   SIRegister_TZoomPanning(CL);
8096:   SIRegister_TTeeEvent(CL);
8097:   //SIRegister_TTeeEventListeners(CL);
8098:   TTeeMouseEventKind', '( meDown, meUp, meMove )
8099:   SIRegister_TTeeMouseEvent(CL);
```

```
8100:   SIRegister_TCustomTeePanel(CL);
8101:   //TChartGradient', 'TTeeGradient
8102:   //TChartGradientClass', 'class of TChartGradient
8103:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
8104:   SIRegister_TTeeZoomPen(CL);
8105:   SIRegister_TTeeZoomBrush(CL);
8106:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
8107:   SIRegister_TTeeZoom(CL);
8108:   FindClass('TOBJECT'),'TCustomTeePanelExtended
8109:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
8110:   SIRegister_TBackImage(CL);
8111:   SIRegister_TCustomTeePanelExtended(CL);
8112:   //TChartBrushClass', 'class of TChartBrush
8113:   SIRegister_TTeeCustomShapeBrushPen(CL);
8114:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
8115:   TTextFormat', '( ttfNormal, ttfHtml )
8116:   SIRegister_TTeeCustomShape(CL);
8117:   SIRegister_TTeeShape(CL);
8118:   SIRegister_TTeeExportData(CL);
8119:  Func TeeStr( const Num : Int) :Str
8120:  Func DateTimeDefaultFormat( const AStep : Double) :Str
8121:  Func TEEDaysInMonth( Year, Month : Word) : Word
8122:  Func FindDateTimeStep( const StepValue : Double) : TDateTimeStep
8123:  Func NextDateTimeStep( const AStep : Double) : Double
8124:  Func PointInLine( const P : TPoint; const px, py, qx, qy : Int):Bool;
8125:  Func PointInLine1( const P, FromPoint, ToPoint : TPoint) :Bool;
8126:  Func PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Int):Boo;
8127:  Func PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Int):Bool;
8128:  Func PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Int):Bool;
8129:  Func PointInPolygon( const P : TPoint; const Poly : array of TPoint) :Bool
8130:  Func PointInTriangle( const P, P0, P1, P2 : TPoint) :Bool;
8131:  Func PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Int) :Bool;
8132:  Func PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Int) :Bool
8133:  Func PointInEllipse( const P : TPoint; const Rect : TRect) :Bool;
8134:  Func PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Int) :Bool;
8135:  Func DelphiToLocalFormat( const Format :Str) :Str
8136:  Func LocalToDelphiFormat( const Format :Str) :Str
8137:  Proc TEEEnableControls(Enable:Bool; const ControlArray : array of TControl)
8138:  Func TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8139:  Proc TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const AnIncrement:Double;
     tmpWhichDateTime:TDateTimeStep)
8140:   TTeeSortCompare', 'Func ( a, b : Int) : Int
8141:   TTeeSortSwap', 'Proc ( a, b : Int)
8142:  Proc TeeSort(StartIndex,EndIndex:Int;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8143:  Func TeeGetUniqueName( AOwner : TComponent; const AStartName :Str) :Str
8144:  Func TeeExtractField( St :Str; Index : Int) :Str;
8145:  Func TeeExtractField1( St :Str; Index : Int; const Separator :Str) :Str;
8146:  Func TeeNumFields( St :Str) : Int;
8147:  Func TeeNumFields1( const St, Separator :Str) : Int;
8148:  Proc TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8149:  Proc TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 :Str)
8150:  // TColorArray', 'array of TColor
8151:  Func GetDefaultColor( const Index : Int) : TColor
8152:  Proc SetDefaultColorPalette;
8153:  Proc SetDefaultColorPalette1( const Palette : array of TColor);
8154:  'TeeCheckBoxSize','LongInt'( 11);
8155:  Proc TeeDrawCheckBox(x,y:Int;Canvas:TCanvas;Checked:Bool;ABackColor:TColor;CheckBox:Bool);
8156:  Func TEEStrToFloatDef( const S :Str; const Default : Extended) : Extended
8157:  Func TryStrToFloat( const S :Str; var Value : Double) :Bool
8158:  Func CrossingLines(const X1,Y1,X2,Y2,X3,Y3,X4,Y4: Double; out x, y : Double) :Bool
8159:  Proc TeeTranslateControl( AControl : TControl);
8160:  Proc TeeTranslateControl1(AControl: TControl;const ExcludeChilds: array of TControl);
8161:  Func ReplaceChar(const AString:str;const Search: Char; const Replace : Char) :Str
8162:  //Proc RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8163:  Func TeeAntiAlias( Panel : TCustomTeePanel; ChartRect :Bool) : TBitmap
8164:  //Proc DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Int;Round:Int);
8165:  //Func ScreenRatio( ACanvas : TCanvas3D) : Double
8166:  Func TeeReadBoolOption( const AKey :Str; DefaultValue :Bool) :Bool
8167:  Proc TeeSaveBoolOption( const AKey :Str; Value :Bool)
8168:  Func TeeReadIntOption( const AKey :Str; DefaultValue : Int) : Int
8169:  Proc TeeSaveIntOption( const AKey :Str; Value : Int)
8170:  Func TeeReadStringOption( const AKey, DefaultValue :Str) :Str
8171:  Proc TeeSaveStringOption( const AKey, Value :Str)
8172:  Func TeeDefaultXMLEncoding :Str
8173:  Proc ConvertTextToXML(Stream: TStream; XMLHeader :Bool)
8174:   TeeWindowHandle', 'Int
8175:  Proc TeeGotoURL( Handle : TeeWindowHandle; const URL :Str)
8176:  Proc HtmlTextOut( ACanvas : TCanvas; x, y : Int; Text :Str)
8177:  Func HtmlTextExtent( ACanvas : TCanvas; const Text :Str) : TSize
8178: end;
8179:
8180: fix and add teEngine:  4.7.6.20
8181: TChartAxisTitle=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8182: TChartSeriesList=class {$IFDEF CLR}sealed{$ENDIF} (TCustomSeriesList)
8183: TSeriesGroup=class {$IFDEF CLR}sealed{$ENDIF} (TCollectionItem)
8184:   Destructor Destroy; override;
8185: TSeriesGroups=class {$IFDEF CLR}sealed{$ENDIF} (TOwnedCollection)
8186:  TChartCustomAxes=class {$IFDEF CLR}sealed{$ENDIF} (TOwnedCollection)
8187:  TChartChangePage=class {$IFDEF CLR}sealed{$ENDIF} (TTeeEvent);
```

```
8188:  TSeriesMarksPositions=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8189:  TMarksItem=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8190:    property Bevel;
8191:     property BevelWidth;
8192:     property Color default ChartMarkColor;
8193:     property Font;
8194:     property Gradient;
8195:     property Picture;
8196:     property Shadow;
8197:     property ShapeStyle;
8198:     property Text:TStrings read GetText write SetText;
8199:     property Transparency;
8200:     property Transparent;
8201:   TMarksItems=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8202:   TSeriesMarksGradient=class {$IFDEF CLR}sealed{$ENDIF} (TChartGradient)
8203:   property Direction default gdRightLeft;
8204:     property EndColor default clWhite;
8205:     property StartColor default clSilver;
8206:     TSeriesMarksSymbol=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8207:       TChartValueLists=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8208:    Procedure Clear; override;
8209:  TEEProcs
8210:  TTeeMouseEvent=class {$IFDEF CLR}sealed{$ENDIF} (TTeeEvent)
8211:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TTeeMouseEvent') do begin
8212:  CNN CreateAscentImages   - unit neuraldatasetsv;
8213:
8214:
8215: using mXBDEUtils
8216: ******************************************************************************
8217:  Proc SetAlias( aAlias, aDirectory :str)
8218:  Proc CheckRegistryEntry(Reg:TRegistry;const Path,Value:Str;const Default,Desired:Variant;Size:Byte);
8219:  Func GetFileVersionNumber( const FileName :Str) : TVersionNo
8220:  Proc SetBDE( aPath, aNode, aValue :Str)
8221:  Func RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Int): Int; stdcall;
8222:  Func GetSystemDirectory( lpBuffer :Str; uSize : UINT) : UINT
8223:  Func GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8224:  Func GetTempPath( nBufferLength : DWORD; lpBuffer :Str) : DWORD
8225:  Func GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer :Str) : DWORD
8226:  Func GetTempFileName(lpPathName,lpPrefixString:str;uUnique:UINT;lpTempFileName:str):UINT;
8227:
8228: Proc SIRegister_cDateTime(CL: TPSPascalCompiler);
8229: begin
8230: AddClassN(FindClass('TOBJECT'),'EDateTime
8231: Func DatePart( const D : TDateTime) : Int
8232: Func TimePart( const D : TDateTime) : Double
8233: Func Century( const D : TDateTime) : Word
8234: Func Year( const D : TDateTime) : Word
8235: Func Month( const D : TDateTime) : Word
8236: Func Day( const D : TDateTime) : Word
8237: Func Hour( const D : TDateTime) : Word
8238: Func Minute( const D : TDateTime) : Word
8239: Func Second( const D : TDateTime) : Word
8240: Func Millisecond( const D : TDateTime) : Word
8241: ('OneDay','Extended').setExtended( 1.0);
8242: ('OneHour','Extended').SetExtended( OneDay / 24);
8243: ('OneMinute','Extended').SetExtended( OneHour / 60);
8244: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8245: ('OneMillisecond','Extended').SetExtended( OneSecond/ 1000);
8246: ('OneWeek','Extended').SetExtended( OneDay * 7);
8247: ('HoursPerDay','Extended').SetExtended( 24);
8248: ('MinutesPerHour','Extended').SetExtended( 60);
8249: ('SecondsPerMinute','Extended').SetExtended( 60);
8250: Proc SetYear( var D : TDateTime; const Year : Word)
8251: Proc SetMonth( var D : TDateTime; const Month : Word)
8252: Proc SetDay( var D : TDateTime; const Day : Word)
8253: Proc SetHour( var D : TDateTime; const Hour : Word)
8254: Proc SetMinute( var D : TDateTime; const Minute : Word)
8255: Proc SetSecond( var D : TDateTime; const Second : Word)
8256: Proc SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8257: Func IsEqual( const D1, D2 : TDateTime) :Bool;
8258: Func IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8259: Func IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8260: Func IsAM( const D : TDateTime) :Bool
8261: Func IsPM( const D : TDateTime) :Bool
8262: Func IsMidnight( const D : TDateTime) :Bool
8263: Func IsNoon( const D : TDateTime) :Bool
8264: Func IsSunday( const D : TDateTime) :Bool
8265: Func IsMonday( const D : TDateTime) :Bool
8266: Func IsTuesday( const D : TDateTime) :Bool
8267: Func IsWedneday( const D : TDateTime) :Bool
8268: Func IsThursday( const D : TDateTime) :Bool
8269: Func IsFriday( const D : TDateTime) :Bool
8270: Func IsSaturday( const D : TDateTime) :Bool
8271: Func IsWeekend( const D : TDateTime) :Bool
8272: Func Noon( const D : TDateTime) : TDateTime
8273: Func Midnight( const D : TDateTime) : TDateTime
8274: Func FirstDayOfMonth( const D : TDateTime) : TDateTime
8275: Func LastDayOfMonth( const D : TDateTime) : TDateTime
8276: Func NextWorkday( const D : TDateTime) : TDateTime
```

```
8277: Func PreviousWorkday( const D : TDateTime) : TDateTime
8278: Func FirstDayOfYear( const D : TDateTime) : TDateTime
8279: Func LastDayOfYear( const D : TDateTime) : TDateTime
8280: Func EasterSunday( const Year : Word) : TDateTime
8281: Func GoodFriday( const Year : Word) : TDateTime
8282: Func AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8283: Func AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8284: Func AddMinutes( const D : TDateTime; const N : Int) : TDateTime
8285: Func AddHours( const D : TDateTime; const N : Int) : TDateTime
8286: Func AddDays( const D : TDateTime; const N : Int) : TDateTime
8287: Func AddWeeks( const D : TDateTime; const N : Int) : TDateTime
8288: Func AddMonths( const D : TDateTime; const N : Int) : TDateTime
8289: Func AddYears( const D : TDateTime; const N : Int) : TDateTime
8290: Func DayOfYear( const Ye, Mo, Da : Word) : Int
8291: Func DayOfYear( const D : TDateTime) : Int
8292: Func DaysInMonth( const Ye, Mo : Word) : Int
8293: Func DaysInMonth( const D : TDateTime) : Int
8294: Func DaysInYear( const Ye : Word) : Int
8295: Func DaysInYearDate( const D : TDateTime) : Int
8296: Func WeekNumber( const D : TDateTime) : Int
8297: Func ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8298: Proc ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8299: Func DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8300: Func DiffSeconds( const D1, D2 : TDateTime) : Int
8301: Func DiffMinutes( const D1, D2 : TDateTime) : Int
8302: Func DiffHours( const D1, D2 : TDateTime) : Int
8303: Func DiffDays( const D1, D2 : TDateTime) : Int
8304: Func DiffWeeks( const D1, D2 : TDateTime) : Int
8305: Func DiffMonths( const D1, D2 : TDateTime) : Int
8306: Func DiffYears( const D1, D2 : TDateTime) : Int
8307: Func GMTBias : Int
8308: Func GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8309: Func LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8310: Func NowAsGMTTime : TDateTime
8311: Func DateTimeToISO8601String( const D : TDateTime) : Ansistr
8312: Func ISO8601StringToTime( const D : Ansistr) : TDateTime
8313: Func ISO8601StringAsDateTime( const D : Ansistr) : TDateTime
8314: Func DateTimeToANSI( const D : TDateTime) : Int
8315: Func ANSIToDateTime( const Julian : Int) : TDateTime
8316: Func DateTimeToISOInt( const D : TDateTime) : Int
8317: Func DateTimeToISOString( const D : TDateTime) : Ansistr
8318: Func ISOIntToDateTime( const ISOInt : Int) : TDateTime
8319: Func TwoDigitRadix2000YearToYear( const Y : Int) : Int
8320: Func DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):Ansistr
8321: Func UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8322: Func DateTimeToUnixTime( const D : TDateTime) : LongWord
8323: Func EnglishShortDayOfWeekStrA( const DayOfWeek : Int) : Ansistr
8324: Func EnglishShortDayOfWeekStrU( const DayOfWeek : Int) : UnicodeString
8325: Func EnglishLongDayOfWeekStrA( const DayOfWeek : Int) : Ansistr
8326: Func EnglishLongDayOfWeekStrU( const DayOfWeek : Int) : UnicodeString
8327: Func EnglishShortMonthStrA( const Month : Int) : Ansistr
8328: Func EnglishShortMonthStrU( const Month : Int) : UnicodeString
8329: Func EnglishLongMonthStrA( const Month : Int) : Ansistr
8330: Func EnglishLongMonthStrU( const Month : Int) : UnicodeString
8331: Func EnglishShortDayOfWeekA( const S : Ansistr) : Int
8332: Func EnglishShortDayOfWeekU( const S : UnicodeString) : Int
8333: Func EnglishLongDayOfWeekA( const S : Ansistr) : Int
8334: Func EnglishLongDayOfWeekU( const S : UnicodeString) : Int
8335: Func EnglishShortMonthA( const S : Ansistr) : Int
8336: Func EnglishShortMonthU( const S : UnicodeString) : Int
8337: Func EnglishLongMonthA( const S : Ansistr) : Int
8338: Func EnglishLongMonthU( const S : UnicodeString) : Int
8339: Func RFC850DayOfWeekA( const S : Ansistr) : Int
8340: Func RFC850DayOfWeekU( const S : UnicodeString) : Int
8341: Func RFC1123DayOfWeekA( const S : Ansistr) : Int
8342: Func RFC1123DayOfWeekU( const S : UnicodeString) : Int
8343: Func RFCMonthA( const S : Ansistr) : Word
8344: Func RFCMonthU( const S : UnicodeString) : Word
8345: Func GMTTimeToRFC1123TimeA( const D : TDateTime;const IncludeSeconds:Boolean) : Ansistr
8346: Func GMTTimeToRFC1123TimeU( const D : TDateTime;const IncludeSeconds:Boolean):UnicodeString
8347: Func GMTDateTimeToRFC1123DateTimeA(const D:TDateTime;const IncludeDayOfWeek:Bool):AnsiStr;
8348: Func GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodStr;
8349: Func DateTimeToRFCDateTimeA( const D : TDateTime) : Ansistr
8350: Func DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8351: Func NowAsRFCDateTimeA : Ansistr
8352: Func NowAsRFCDateTimeU : UnicodeString
8353: Func RFCDateTimeToGMTDateTime( const S : Ansistr) : TDateTime
8354: Func RFCDateTimeToDateTime( const S : Ansistr) : TDateTime
8355: Func RFCTimeZoneToGMTBias( const Zone : Ansistr) : Int
8356: Func TimePeriodStr( const D : TDateTime) : Ansistr
8357: Proc SelfTestCDate
8358: end;
8359: //*****************************************CFileUtils
8360: Func PathHasDriveLetterA( const Path : Ansistr) :Bool
8361: Func PathHasDriveLetter( const Path :Str) :Bool
8362: Func PathIsDriveLetterA( const Path : Ansistr) :Bool
8363: Func PathIsDriveLetter( const Path :Str) :Bool
8364: Func PathIsDriveRootA( const Path : Ansistr) :Bool
8365: Func PathIsDriveRoot( const Path :Str) :Bool
```

```
8366: Func PathIsRootA( const Path : Ansistr) :Bool
8367: Func PathIsRoot( const Path :Str) :Bool
8368: Func PathIsUNCPathA( const Path : Ansistr) :Bool
8369: Func PathIsUNCPath( const Path :Str) :Bool
8370: Func PathIsAbsoluteA( const Path : Ansistr) :Bool
8371: Func PathIsAbsolute( const Path :Str) :Bool
8372: Func PathIsDirectoryA( const Path : Ansistr) :Bool
8373: Func PathIsDirectory( const Path :Str) :Bool
8374: Func PathInclSuffixA( const Path : Ansistr; const PathSep : Char) : Ansistr
8375: Func PathInclSuffix( const Path :Str; const PathSep : Char) :Str
8376: Func PathExclSuffixA( const Path : Ansistr; const PathSep : Char) : Ansistr
8377: Func PathExclSuffix( const Path :Str; const PathSep : Char) :Str
8378: Proc PathEnsureSuffixA( var Path : Ansistr; const PathSep : Char)
8379: Proc PathEnsureSuffix( var Path :Str; const PathSep : Char)
8380: Proc PathEnsureNoSuffixA( var Path : Ansistr; const PathSep : Char)
8381: Proc PathEnsureNoSuffix( var Path :Str; const PathSep : Char)
8382: //Func PathCanonicalA( const Path : Ansistr; const PathSep : Char) : Ansistr
8383: Func PathCanonical( const Path :Str; const PathSep : Char) :Str
8384: Func PathExpandA(const Path:Ansistr;const BasePath:AnsiStr;const PathSep:Char):AnsiStr
8385: Func PathExpand(const Path:str; const BasePath :Str; const PathSep : Char) :Str
8386: Func PathLeftElementA( const Path : Ansistr; const PathSep : Char) : Ansistr
8387: Func PathLeftElement( const Path :Str; const PathSep : Char) :Str
8388: Proc PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8389: Proc PathSplitLeftElement(const Path:str; var LeftElement,RightPath:Str;const PathSep:Char);
8390: Proc DecodeFilePathA(const FilePath:AnsiStr;var Path,FileName:AnsiStr;const PathSep:Char;
8391: Proc DecodeFilePath(const FilePath:str; var Path,FileName:str;const PathSep: Char)
8392: Func FileNameValidA( const FileName : Ansistr) : Ansistr
8393: Func FileNameValid( const FileName :Str) :Str
8394: Func FilePathA(const FileName,Path:Ansistr;const BasePath:AnsiStr;const PathSep:Char):Ansistr;
8395: Func FilePath(const FileName,Path:str;const BasePath:Str;const PathSep:Char):Str
8396: Func DirectoryExpandA(const Path:Ansistr;const BasePath:Ansistr;const PathSep:Char):Ansistr
8397: Func DirectoryExpand(const Path:str;const BasePath:str;const PathSep:Char):str
8398: Func UnixPathToWinPath( const Path : Ansistr) : Ansistr
8399: Func WinPathToUnixPath( const Path : Ansistr) : Ansistr
8400: Proc CCopyFile( const FileName, DestName :Str)
8401: Proc CMoveFile( const FileName, DestName :Str)
8402: Func CDeleteFiles( const FileMask :Str) :Bool
8403: Func FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64;const FilePos:TFileSeekPos):Int64;
8404: Proc FileCloseEx( const FileHandle : TFileHandle)
8405: Func FileExistsA( const FileName : Ansistr) :Bool
8406:  Func CFileExists( const FileName :Str) :Bool
8407:  Func CFileGetSize( const FileName :Str) : Int64
8408: Func FileGetDateTime( const FileName :Str) : TDateTime
8409: Func FileGetDateTime2( const FileName :Str) : TDateTime
8410: Func FileIsReadOnly( const FileName :Str) :Bool
8411: Proc FileDeleteEx( const FileName :Str)
8412: Proc FileRenameEx( const OldFileName, NewFileName :Str)
8413: Func ReadFileStrA( const FileName:Ansistr; const FileSharing : TFileSharing; const FileCreationMode :
      TFileCreationMode; const FileOpenWait : PFileOpenWait) : Ansistr
8414: Func DirectoryEntryExists( const Name :Str) :Bool
8415: Func DirectoryEntrySize( const Name :Str) : Int64
8416:  Func CDirectoryExists( const DirectoryName :Str) :Bool
8417: Func DirectoryGetDateTime( const DirectoryName :Str) : TDateTime
8418: Proc CDirectoryCreate( const DirectoryName :Str)
8419: Func GetFirstFileNameMatching( const FileMask :Str) :Str
8420: Func DirEntryGetAttr( const FileName : Ansistr) : Int
8421: Func DirEntryIsDirectory( const FileName : Ansistr) :Bool
8422: Func FileHasAttr( const FileName :Str; const Attr : Word) :Bool
8423:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8424:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8425: Func DriveIsValid( const Drive : Char) :Bool
8426: Func DriveGetType( const Path : Ansistr) : TLogicalDriveType
8427: Func DriveFreeSpace( const Path : Ansistr) : Int64
8428:
8429: Proc SIRegister_cTimers(CL: TPSPascalCompiler);
8430: begin
8431:  AddClassN(FindClass('TOBJECT'),'ETimers
8432:  Const('TickFrequency','LongInt'( 1000);Func GetTick : LongWord
8433: Func TickDelta( const D1, D2 : LongWord) : Int
8434: Func TickDeltaW( const D1, D2 : LongWord) : LongWord
8435:   AddTypeS('THPTimer', 'Int64
8436: Proc StartTimer( var Timer : THPTimer)
8437: Proc StopTimer( var Timer : THPTimer)
8438: Proc ResumeTimer( var StoppedTimer : THPTimer)
8439: Proc InitStoppedTimer( var Timer : THPTimer)
8440: Proc InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Int)
8441: Func MillisecondsElapsed( const Timer: THPTimer; const TimerRunning :Bool) : Int
8442: Func MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning :Bool) : Int64
8443: Proc WaitMicroseconds( const MicroSeconds : Int)
8444: Func GetHighPrecisionFrequency : Int64
8445: Func GetHighPrecisionTimerOverhead : Int64
8446: Proc AdjustTimerForOverhead(var StoppedTimer:THPTimer; const Overhead:Int64)
8447: Proc SelfTestCTimer
8448: end;
8449:
8450: Proc SIRegister_cRandom(CL: TPSPascalCompiler);
8451: begin
8452:  Func RandomSeed : LongWord
8453:  Proc AddEntropy( const Value : LongWord)
```

```
8454:  Func RandomUniform : LongWord;
8455:  Func RandomUniform1( const N : Int) : Int;
8456:  Func RandomBoolean :Bool
8457:  Func RandomByte : Byte
8458:  Func RandomByteNonZero : Byte
8459:  Func RandomWord : Word
8460:  Func RandomInt64 : Int64;
8461:  Func RandomInt641( const N : Int64) : Int64;
8462:  Func RandomHex( const Digits : Int) :Str
8463:  Func RandomFloat : Extended
8464:  Func RandomAlphaStr( const Length : Int) : Ansistr
8465:  Func RandomPseudoWord( const Length : Int) : Ansistr
8466:  Func RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiStr;
8467:  Func mwcRandomLongWord : LongWord
8468:  Func urnRandomLongWord : LongWord
8469:  Func moaRandomFloat : Extended
8470:  Func mwcRandomFloat : Extended
8471:  Func RandomNormalF : Extended
8472:  Proc SelfTestCRandom
8473: end;
8474:
8475: Proc SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8476: begin
8477:  // PIntArray', '^TIntArray // will not work
8478:  Addtypes('TConvertTabsProc','function(const Line:Ansistr; TabWidth: Int):Ansistr
8479:  TConvertTabsProcEx,function(const Line:Ansistr;TabWidth:Int;var HasTabs:boolean):Ansistr
8480:  Func synMax( x, y : Int) : Int
8481:  Func synMin( x, y : Int) : Int
8482:  Func synMinMax( x, mi, ma : Int) : Int
8483:  Proc synSwapInt( var l, r : Int)
8484:  Func synMaxPoint( const P1, P2 : TPoint) : TPoint
8485:  Func synMinPoint( const P1, P2 : TPoint) : TPoint
8486:  //Func synGetIntArray( Count :Card; InitialValue : Int) : PIntArray
8487:  Proc synInternalFillRect( dc : HDC; const rcPaint : TRect)
8488:  Func synGetBestConvertTabsProc( TabWidth : Int) : TConvertTabsProc
8489:  Func synConvertTabs( const Line : Ansistr; TabWidth : Int) : Ansistr
8490:  Func synGetBestConvertTabsProcEx( TabWidth : Int) : TConvertTabsProcEx
8491:  Func synConvertTabsEx(const Line:Ansistr;TabWidth:Int;var HasTabs:bool):Ansistr;
8492:  Func synGetExpandedLength( const aStr :Str; aTabWidth : Int) : Int
8493:  Func synCharIndex2CaretPos( Index, TabWidth : Int; const Line :Str) : Int
8494:  Func synCaretPos2CharIndex(Positin,TabWidth:int;const Line:str;var InsideTabChar:bool):int;
8495:  Func synStrScanForCharInSet(const Line:str;Start:Int;AChars:TSynIdentChars):int;
8496:  Func synStrRScanForCharInSet(const Line:str;Start:Int;AChars:TSynIdentChars):Int;
8497:   TStringType', '(stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8498:  +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida)
8499:  ('C3_NONSPACING','LongInt'( 1);
8500:  'C3_DIACRITIC','LongInt'( 2);
8501:  'C3_VOWELMARK','LongInt'( 4);
8502:  ('C3_SYMBOL','LongInt'( 8);
8503:  ('C3_KATAKANA',LongWord( $0010);
8504:  ('C3_HIRAGANA',LongWord( $0020);
8505:  ('C3_HALFWIDTH',LongWord( $0040);
8506:  ('C3_FULLWIDTH',LongWord( $0080);
8507:  ('C3_IDEOGRAPH',LongWord( $0100);
8508:  ('C3_KASHIDA',LongWord( $0200);
8509:  ('C3_LEXICAL',LongWord( $0400);
8510:  ('C3_ALPHA',LongWord( $8000);
8511:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8512:  Func synStrScanForMultiByteChar( const Line :Str; Start : Int) : Int
8513:  Func synStrRScanForMultiByteChar( const Line :Str; Start : Int) : Int
8514:  Func synIsStringType( Value : Word) : TStringType
8515:  Func synGetEOL( Line : PChar) : PChar
8516:  Func synEncodeString( s :Str) :Str
8517:  Func synDecodeString( s :Str) :Str
8518:  Proc synFreeAndNil( var Obj: TObject)
8519:  Proc synAssert( Expr :Bool)
8520:  Func synLastDelimiter( const Delimiters, S :Str) : Int
8521:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8522:   TReplaceFlags', 'set of TReplaceFlag )
8523:  Func synStringReplace(const S, OldPattern,NewPattern:str;Flags:TReplaceFlags):str
8524:  Func synGetRValue( RGBValue : TColor) : byte
8525:  Func synGetGValue( RGBValue : TColor) : byte
8526:  Func synGetBValue( RGBValue : TColor) : byte
8527:  Func synRGB( r, g, b : Byte) :Card
8528: //THighlighterAttriProc', 'Func ( Highlighter : TSynCustomHigh'
8529: //+'lighter;Attri:TSynHighlighterAttributes;UniqueAttriName:str;Params array of Pointr):Bool;
8530:  //Func synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates :Bool;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) :Bool
8531:  Func synCalcFCS( const ABuf, ABufSize :Card) : Word
8532:  Proc synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,AEndColor:TColor;ASteps:Int;const ARect :
      TRect; const AHorizontal:Bool) end;
8533: Func GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8534: Func GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8535: Func GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8536:
8537: Proc SIRegister_synautil(CL: TPSPascalCompiler);
8538: begin
8539:  Func STimeZoneBias : Int
8540:  Func TimeZone :Str
```

```
8541:  Func Rfc822DateTime( t : TDateTime) :Str
8542:  Func CDateTime( t : TDateTime) :Str
8543:  Func SimpleDateTime( t : TDateTime) :Str
8544:  Func AnsiCDateTime( t : TDateTime) :Str
8545:  Func GetMonthNumber( Value :Str) : Int
8546:  Func GetTimeFromStr( Value :Str) : TDateTime
8547:  Func GetDateMDYFromStr( Value :Str) : TDateTime
8548:  Func DecodeRfcDateTime( Value :Str) : TDateTime
8549:  Func GetUTTime : TDateTime
8550:  Func SetUTTime( Newdt : TDateTime) :Bool
8551:  Func SGetTick : LongWord
8552:  Func STickDelta( TickOld, TickNew : LongWord) : LongWord
8553:  Func CodeInt( Value : Word) : Ansistr
8554:  Func DecodeInt( const Value : Ansistr; Index : Int) : Word
8555:  Func CodeLongInt( Value : LongInt) : Ansistr
8556:  Func DecodeLongInt( const Value : Ansistr; Index : Int) : LongInt
8557:  Func DumpStr( const Buffer : Ansistr) :Str
8558:  Func DumpExStr( const Buffer : Ansistr) :Str
8559:  Proc Dump( const Buffer : Ansistr; DumpFile :Str)
8560:  Proc DumpEx( const Buffer : Ansistr; DumpFile :Str)
8561:  Func TrimSPLeft( const S :Str) :Str
8562:  Func TrimSPRight( const S :Str) :Str
8563:  Func TrimSP( const S :Str) :Str
8564:  Func SeparateLeft( const Value, Delimiter :Str) :Str
8565:  Func SeparateRight( const Value, Delimiter :Str) :Str
8566:  Func SGetParameter( const Value, Parameter :Str) :Str
8567:  Proc ParseParametersEx(Value,Delimiter:str;const Parameters : TStrings)
8568:  Proc ParseParameters( Value :Str; const Parameters : TStrings)
8569:  Func IndexByBegin( Value :Str; const List : TStrings) : Int
8570:  Func GetEmailAddr( const Value :Str) :Str
8571:  Func GetEmailDesc( Value :Str) :Str
8572:  Func CStrToHex( const Value : Ansistr) :Str
8573:  Func CIntToBin( Value : Int; Digits : Byte) :Str
8574:  Func CBinToInt( const Value :Str) : Int
8575:  Func ParseURL(URL:str;var Prot,User,Pass,Host,Port,Path,Para:str):str
8576:  Func CReplaceString( Value, Search, Replace : Ansistr) : Ansistr
8577:  Func CRPosEx( const Sub, Value :Str; From : Int) : Int
8578:  Func CRPos( const Sub, Value :Str) : Int
8579:  Func FetchBin( var Value :Str; const Delimiter :Str) :Str
8580:  Func CFetch( var Value :Str; const Delimiter :Str) :Str
8581:  Func FetchEx(var Value:Str;const Delimiter,Quotation:str):Str
8582:  Func IsBinaryString( const Value : Ansistr) :Bool
8583:  Func PosCRLF( const Value : Ansistr; var Terminator : Ansistr) : Int
8584:  Proc StringsTrim( const value : TStrings)
8585:  Func PosFrom( const SubStr, Value :Str; From : Int) : Int
8586:  Func IncPoint( const p : ___pointer; Value : Int) : ___pointer
8587:  Func GetBetween( const PairBegin, PairEnd, Value :Str) :Str
8588:  Func CCountOfChar( const Value :Str; aChr : char) : Int
8589:  Func UnquoteStr( const Value :Str; Quote : Char) :Str
8590:  Func QuoteStr( const Value :Str; Quote : Char) :Str
8591:  Proc HeadersToList( const Value : TStrings)
8592:  Proc ListToHeaders( const Value : TStrings)
8593:  Func SwapBytes( Value : Int) : Int
8594:  Func ReadStrFromStream( const Stream : TStream; len : Int) : Ansistr
8595:  Proc WriteStrToStream( const Stream : TStream; Value : Ansistr)
8596:  Func GetTempFile( const Dir, prefix : Ansistr) : Ansistr
8597:  Func CPadString( const Value:Ansistr;len:Int;Pad:AnsiChar): Ansistr
8598:  Func CXorString( Indata1, Indata2 : Ansistr) : Ansistr
8599:  Func NormalizeHeader( Value : TStrings; var Index : Int) :Str
8600: end;
8601:
8602: Proc SIRegister_StCRC(CL: TPSPascalCompiler);
8603: begin
8604: ('CrcBufSize','LongInt'( 2048);
8605: Func Adler32Prim( var Data, DataSize :Card; CurCrc : LongInt): LongInt
8606: Func Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8607: Func Adler32OfFile( FileName : Ansistr) : LongInt
8608: Func Crc16Prim( var Data, DataSize, CurCrc :Card) :Card
8609: Func Crc16OfStream( Stream : TStream; CurCrc :Card) :Card
8610: Func Crc16OfFile( FileName : Ansistr) :Card
8611: Func Crc32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
8612: Func Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8613: Func Crc32OfFile( FileName : Ansistr) : LongInt
8614: Func InternetSumPrim( var Data, DataSize, CurCrc :Card) :Card
8615: Func InternetSumOfStream( Stream : TStream; CurCrc :Card) :Card
8616: Func InternetSumOfFile( FileName : Ansistr) :Card
8617: Func Kermit16Prim( var Data, DataSize, CurCrc :Card) :Card
8618: Func Kermit16OfStream( Stream : TStream; CurCrc :Card) :Card
8619: Func Kermit16OfFile( FileName : Ansistr) :Card
8620: end;
8621:
8622: Proc SIRegister_ComObj(cl: TPSPascalCompiler);
8623: begin
8624:  Func CreateOleObject(const ClassName:Str): IDispatch;
8625:  Func GetActiveOleObject(const ClassName:Str): IDispatch;
8626:  Func ProgIDToClassID(const ProgID:Str): TGUID;
8627:  Func ClassIDToProgID(const ClassID: TGUID):Str;
8628:  Func CreateClassID:Str;
8629:  Func CreateGUIDString:Str;
```

```
8630:  Func CreateGUIDID:Str;
8631:  Proc OleError(ErrorCode: longint)
8632:  Proc OleCheck(Result: HResult);
8633: end;
8634:
8635:  Func xCreateOleObject( const ClassName :Str) : Variant //or IDispatch
8636:  Func xGetActiveOleObject( const ClassName :Str) : Variant
8637:  //Func DllGetClassObject(const CLSID:TCLSID;const IID TIID;var Obj):HResult
8638:  Func DllCanUnloadNow : HResult
8639:  Func DllRegisterServer : HResult
8640:  Func DllUnregisterServer : HResult
8641:  Func VarFromInterface( Unknown : IUnknown) : Variant
8642:  Func VarToInterface( const V : Variant) : IDispatch
8643:  Func VarToAutoObject( const V : Variant) : TAutoObject
8644:  //Proc DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8645:  //Proc DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8646:  Proc OleError( ErrorCode : HResult)
8647:  Proc OleCheck( Result : HResult)
8648:  Func StringToClassID( const S :Str) : TCLSID
8649:  Func ClassIDToString( const ClassID : TCLSID) :Str
8650:  Func xProgIDToClassID( const ProgID :Str) : TCLSID
8651:  Func xClassIDToProgID( const ClassID : TCLSID) :Str
8652:  Func xWideCompareStr( const S1, S2 : WideString) : Int
8653:  Func xWideSameStr( const S1, S2 : WideString) :Bool
8654:  Func xGUIDToString( const ClassID : TGUID) :Str
8655:  Func xStringToGUID( const S :Str) : TGUID
8656:  Func xGetModuleName( Module : HMODULE) :Str
8657:  Func xAcquireExceptionObject : TObject
8658:  Func xIfThen(AValue :Bool; const ATrue: Int; const AFalse: Int): Int
8659:  Func xUtf8Encode( const WS : WideString) : UTF8String
8660:  Func xUtf8Decode( const S : UTF8String) : WideString
8661:  Func xExcludeTrailingPathDelimiter( const S :Str) :Str
8662:  Func xIncludeTrailingPathDelimiter( const S :Str) :Str
8663:  Func XRTLHandleCOMException : HResult
8664:  Proc XRTLCheckArgument( Flag :Bool)
8665:  //Proc XRTLCheckOutArgument( out Arg)
8666:  Proc XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8667:  Proc XRTLInterfaceDisconnect(const Source:IUnknown;const IID:TIID;var Connection:Longint)
8668:  Func XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8669:  Func XRTLUnRegisterActiveObject( var RegisterCookie : Int) : HResult
8670:  //Func XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8671:  Proc XRTLEnumActiveObjects( Strings : TStrings)
8672: Func XRTLDefaultCategoryManager: IUnknown;
8673: Func XRTLIsCategoryEmpty(CatID:TGUID;const CategoryManager:IUnknown=nil):Bool;
8674: // ICatRegister helper functions
8675: Func  XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8676:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8677:                                const CategoryManager: IUnknown = nil): HResult;
8678: Func  XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8679:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8680:                                const CategoryManager: IUnknown = nil): HResult;
8681: Func  XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8682:                                const CategoryManager: IUnknown = nil): HResult;
8683: Func  XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8684:                                const CategoryManager: IUnknown = nil): HResult;
8685: // ICatInformation helper functions
8686: Func  XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8687:                                    LocaleID: TLCID = LOCALE_USER_DEFAULT;
8688:                                const CategoryManager: IUnknown = nil): HResult;
8689: Func  XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8690:                                const CategoryManager: IUnknown = nil): HResult;
8691: Func  XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8692:                                const CategoryManager: IUnknown = nil): HResult;
8693: Func  XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8694:                                const CategoryManager: IUnknown = nil): HResult;
8695: Func XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8696:                    const ADelete:Bool = True): WideString;
8697: Func XRTLRPos(const ASub, AIn: WideString; AStart: Int = -1): Int;
8698: Func XRTLGetVariantAsString( const Value : Variant) :Str
8699: Func XRTLDateTimeToTimeZoneTime(DT:TDateTime;TimeZone:TXRTLTimeZone):TDateTime
8700: Func XRTLGetTimeZones : TXRTLTimeZones
8701: Func XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8702: Func DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8703: Func GMTNow : TDateTime
8704: Func GMTToLocalTime( GMT : TDateTime) : TDateTime
8705: Func LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8706: Proc XRTLNotImplemented
8707: Proc XRTLRaiseError( E : Exception)
8708: Proc XRTLRaise( E : Exception);
8709: Proc XRaise( E : Exception);
8710: Proc XRTLInvalidOperation(ClassName:str;OperationName:str;Description:Str)
8711:
8712: Proc SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8713: begin
8714:   SIRegister_IXRTLValue(CL);
8715:   SIRegister_TXRTLValue(CL);
8716:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
```

```
8717:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8718: Func XRTLValue( const AValue :Card) : IXRTLValue;
8719: Func XRTLSetValue(const IValue:IXRTLValue; const AValue:Card) :Card;
8720: Func XRTLGetAsCardinal( const IValue : IXRTLValue) :Card
8721: Func XRTLGetAsCardinalDef(const IValue:IXRTLValue;const DefValue:Card): Card
8722: Func XRTLValue1( const AValue : Int) : IXRTLValue;
8723: Func XRTLSetValue1( const IValue : IXRTLValue; const AValue : Int) : Int;
8724: Func XRTLGetAsInt( const IValue : IXRTLValue) : Int
8725: Func XRTLGetAsIntDef( const IValue : IXRTLValue; const DefValue : Int) : Int
8726: Func XRTLValue2( const AValue : Int64) : IXRTLValue;
8727: Func XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8728: Func XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8729: Func XRTLGetAsInt64Def(const IValue:IXRTLValue;const DefValue:Int64) : Int64
8730: Func XRTLValue3( const AValue : Single) : IXRTLValue;
8731: Func XRTLSetValue3( const IValue : IXRTLValue; const AValue:Single) : Single;
8732: Func XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8733: Func XRTLGetAsSingleDef(const IValue: RTLValue;const DefValue: Single):Single
8734: Func XRTLValue4( const AValue : Double) : IXRTLValue;
8735: Func XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8736: Func XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8737: Func XRTLGetAsDoubleDef(const IValue:IXRTLValue;const DefValue:Double):Double
8738: Func XRTLValue5( const AValue : Extended) : IXRTLValue;
8739: Func XRTLSetValue5(const IValue:IXRTLValue;const AValue:Extended): Extended;
8740: Func XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8741: Func XRTLGetAsExtendedDef(const IValue:IXRTLValue;const DefValue:Extended): Extended
8742: Func XRTLValue6( const AValue : IInterface) : IXRTLValue;
8743: Func XRTLSetValue6(const IValue:IXRTLValue;const AValue:IInterface):IInterface;
8744: Func XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8745:  //Func XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8746: Func XRTLGetAsInterfaceDef(const IValue:IXRTLValue;const DefValue:IInterface):IInterface;
8747: Func XRTLValue7( const AValue : WideString) : IXRTLValue;
8748: Func XRTLSetValue7(const IValue:IXRTLValue;const AValue:WideString):WideString;
8749: Func XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8750: Func XRTLGetAsWideStringDef(const IValue:IXRTLValue;const DefValue:WideString): WideString
8751: Func XRTLValue8(const AValue:TObject; const AOwnValue:Boolean): IXRTLValue;
8752: Func XRTLSetValue8(const IValue:IXRTLValue;const AValue:TObject) : TObject;
8753: Func XRTLGetAsObject(const IValue:IXRTLValue;const ADetachOwnership:Bool): TObject;
8754: Func XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8755: //Func XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8756: //Func XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8757: //Func XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8758: //Func XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8759: Func XRTLValueV( const AValue : Variant) : IXRTLValue;
8760: Func XRTLSetValueV(const IValue:IXRTLValue; const AValue : Variant): Variant;
8761: Func XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8762: Func XRTLGetAsVariantDef(const IValue:IXRTLValue;const DefValue:Variant): Variant
8763: Func XRTLValue10( const AValue : Currency) : IXRTLValue;
8764: Func XRTLSetValue10(const IValue:IXRTLValue; const AValue:Currency):Currency;
8765: Func XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8766: Func XRTLGetAsCurrencyDef(const IValue:IXRTLValue;const DefValue:Currency): Currency
8767: Func XRTLValue11( const AValue : Comp) : IXRTLValue;
8768: Func XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8769: Func XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8770: Func XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8771: Func XRTLValue12( const AValue : TClass) : IXRTLValue;
8772: Func XRTLSetValue12( const IValue:IXRTLValue; const AValue : TClass) : TClass;
8773: Func XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8774: Func XRTLGetAsClassDef(const IValue: IXRTLValue;const DefValue:TClass):TClass
8775: Func XRTLValue13( const AValue : TGUID) : IXRTLValue;
8776: Func XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8777: Func XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8778: Func XRTLGetAsGUIDDef(const IValue:IXRTLValue; const DefValue : TGUID) : TGUID
8779: Func XRTLValue14( const AValue :Bool) : IXRTLValue;
8780: Func XRTLSetValue14( const IValue:IXRTLValue;const AValue:Boolean):Boolean;
8781: Func XRTLGetAsBoolean( const IValue : IXRTLValue) :Bool
8782: Func XRTLGetAsBooleanDef(const IValue:IXRTLValue;const DefValue:Bool):Bool
8783: end;
8784:
8785: //*****************************unit uPSI_GR32;*******************************
8786:
8787:  Func Color32( WinColor : TColor) : TColor32;
8788:  Func Color321( R, G, B : Byte; A : Byte) : TColor32;
8789:  Func Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8790:  Func Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8791:  Func WinColor( Color32 : TColor32) : TColor
8792:  Func ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8793:  Proc Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8794:  Proc Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8795:  Func Color32Components( R, G, B, A :Bool) : TColor32Components
8796:  Func RedComponent( Color32 : TColor32) : Int
8797:  Func GreenComponent( Color32 : TColor32) : Int
8798:  Func BlueComponent( Color32 : TColor32) : Int
8799:  Func AlphaComponent( Color32 : TColor32) : Int
8800:  Func Intensity( Color32 : TColor32) : Int
8801:  Func SetAlpha( Color32 : TColor32; NewAlpha : Int) : TColor32
8802:  Func HSLtoRGB( H, S, L : Single) : TColor32;
8803:  Proc RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8804:  Func HSLtoRGB1( H, S, L : Int) : TColor32;
```

```
8805:  Proc RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8806:  Func WinPalette( const P : TPalette32) : HPALETTE
8807:  Func FloatPoint( X, Y : Single) : TFloatPoint;
8808:  Func FloatPoint1( const P : TPoint) : TFloatPoint;
8809:  Func FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8810:  Func FixedPoint( X, Y : Int) : TFixedPoint;
8811:  Func FixedPoint1( X, Y : Single) : TFixedPoint;
8812:  Func FixedPoint2( const P : TPoint) : TFixedPoint;
8813:  Func FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8814:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8815:  Func MakeRect( const L, T, R, B : Int) : TRect;
8816:  Func MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8817:  Func MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8818:  Func GFixedRect( const L, T, R, B : TFixed) : TRect;
8819:  Func FixedRect1( const ARect : TRect) : TRect;
8820:  Func FixedRect2( const FR : TFloatRect) : TRect;
8821:  Func GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8822:  Func FloatRect1( const ARect : TRect) : TFloatRect;
8823:  Func FloatRect2( const FXR : TRect) : TFloatRect;
8824:  Func GIntersectRect( out Dst : TRect; const R1, R2 : TRect) :Bool;
8825:  Func IntersectRect1(out Dst:TFloatRect;const FR1,FR2:TFloatRect):Boolean;
8826:  Func GUnionRect( out Rect : TRect; const R1, R2 : TRect) :Bool;
8827:  Func UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect):Boolean;
8828:  Func GEqualRect( const R1, R2 : TRect) :Bool;
8829:  Func EqualRect1( const R1, R2 : TFloatRect) :Bool;
8830:  Proc GInflateRect( var R : TRect; Dx, Dy : Int);
8831:  Proc InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8832:  Proc GOffsetRect( var R : TRect; Dx, Dy : Int);
8833:  Proc OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8834:  Func IsRectEmpty( const R : TRect) :Bool;
8835:  Func IsRectEmpty1( const FR : TFloatRect) :Bool;
8836:  Func GPtInRect( const R : TRect; const P : TPoint) :Bool;
8837:  Func PtInRect1( const R : TFloatRect; const P : TPoint) :Bool;
8838:  Func PtInRect2( const R : TRect; const P : TFloatPoint) :Bool;
8839:  Func PtInRect3( const R : TFloatRect; const P : TFloatPoint) :Bool;
8840:  Func EqualRectSize( const R1, R2 : TRect) :Bool;
8841:  Func EqualRectSize1( const R1, R2 : TFloatRect) :Bool;
8842:  Func MessageBeep( uType : UINT) : BOOL
8843:  Func ShowCursor( bShow : BOOL) : Int
8844:  Func SetCursorPos( X, Y : Int) : BOOL
8845:  Func SetCursor( hCursor : HICON) : HCURSOR
8846:  Func GetCursorPos( var lpPoint : TPoint) : BOOL
8847:  //Func ClipCursor( lpRect : PRect) : BOOL
8848:  Func GetClipCursor( var lpRect : TRect) : BOOL
8849:  Func GetCursor : HCURSOR
8850:  Func CreateCaret(hWnd: HWND;hBitmap:HBITMAP;nWidth, nHeight:Int):BOOL
8851:  Func GetCaretBlinkTime : UINT
8852:  Func SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8853:  Func DestroyCaret : BOOL
8854:  Func HideCaret( hWnd : HWND) : BOOL
8855:  Func ShowCaret( hWnd : HWND) : BOOL
8856:  Func SetCaretPos( X, Y : Int) : BOOL
8857:  Func GetCaretPos( var lpPoint : TPoint) : BOOL
8858:  Func ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8859:  Func ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8860:  Func MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints: UINT):Int
8861:  Func WindowFromPoint( Point : TPoint) : HWND
8862:  Func ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8863:
8864: Proc SIRegister_GR32_Math(CL: TPSPascalCompiler);
8865: begin
8866:  Func FixedFloor( A : TFixed) : Int
8867:  Func FixedCeil( A : TFixed) : Int
8868:  Func FixedMul( A, B : TFixed) : TFixed
8869:  Func FixedDiv( A, B : TFixed) : TFixed
8870:  Func OneOver( Value : TFixed) : TFixed
8871:  Func FixedRound( A : TFixed) : Int
8872:  Func FixedSqr( Value : TFixed) : TFixed
8873:  Func FixedSqrtLP( Value : TFixed) : TFixed
8874:  Func FixedSqrtHP( Value : TFixed) : TFixed
8875:  Func FixedCombine( W, X, Y : TFixed) : TFixed
8876:  Proc GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8877:  Proc GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8878:  Func GRHypot( const X, Y : TFloat) : TFloat;
8879:  Func Hypot1( const X, Y : Int) : Int;
8880:  Func FastSqrt( const Value : TFloat) : TFloat
8881:  Func FastSqrtBab1( const Value : TFloat) : TFloat
8882:  Func FastSqrtBab2( const Value : TFloat) : TFloat
8883:  Func FastInvSqrt( const Value : Single) : Single;
8884:  Func MulDiv( Multiplicand, Multiplier, Divisor : Int) : Int
8885:  Func GRIsPowerOf2( Value : Int) :Bool
8886:  Func PrevPowerOf2( Value : Int) : Int
8887:  Func NextPowerOf2( Value : Int) : Int
8888:  Func Average( A, B : Int) : Int
8889:  Func GRSign( Value : Int) : Int
8890:  Func FloatMod( x, y : Double) : Double
8891: end;
8892:
8893: Proc SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
```

```
8894: begin
8895:  Func Clamp( const Value : Int) : Int;
8896:  Proc GRFillWord( var X, Count :Card; Value : Longword)
8897:  Func StackAlloc( Size : Int) : Pointer
8898:  Proc StackFree( P : Pointer)
8899:  Proc Swap( var A, B : Pointer);
8900:  Proc Swap1( var A, B : Int);
8901:  Proc Swap2( var A, B : TFixed);
8902:  Proc Swap3( var A, B : TColor32);
8903:  Proc TestSwap( var A, B : Int);
8904:  Proc TestSwap1( var A, B : TFixed);
8905:  Func TestClip( var A, B : Int; const Size : Int) :Bool;
8906:  Func TestClip1( var A, B : Int; const Start, Stop : Int) :Bool;
8907:  Func GRConstrain( const Value, Lo, Hi : Int) : Int;
8908:  Func Constrain1( const Value, Lo, Hi : Single) : Single;
8909:  Func SwapConstrain( const Value:Int; Constrain1,Constrain2:Int) : Int
8910:  Func GRMin( const A, B, C : Int) : Int;
8911:  Func GRMax( const A, B, C : Int) : Int;
8912:  Func Clamp( Value, Max : Int) : Int;
8913:  Func Clamp1( Value, Min, Max : Int) : Int;
8914:  Func Wrap( Value, Max : Int) : Int;
8915:  Func Wrap1( Value, Min, Max : Int) : Int;
8916:  Func Wrap3( Value, Max : Single) : Single;;
8917:  Func WrapPow2( Value, Max : Int) : Int;
8918:  Func WrapPow21( Value, Min, Max : Int) : Int;
8919:  Func Mirror( Value, Max : Int) : Int;
8920:  Func Mirror1( Value, Min, Max : Int) : Int;
8921:  Func MirrorPow2( Value, Max : Int) : Int;
8922:  Func MirrorPow21( Value, Min, Max : Int) : Int;
8923:  Func GetOptimalWrap( Max : Int) : TWrapProc;
8924:  Func GetOptimalWrap1( Min, Max : Int) : TWrapProcEx;
8925:  Func GetOptimalMirror( Max : Int) : TWrapProc;
8926:  Func GetOptimalMirror1( Min, Max : Int) : TWrapProcEx;
8927:  Func GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8928:  Func GetWrapProc1( WrapMode : TWrapMode; Max : Int) : TWrapProc;
8929:  Func GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8930:  Func GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Int):TWrapProcEx;
8931:  Func Div255( Value :Card) :Card
8932:  Func SAR_4( Value : Int) : Int
8933:  Func SAR_8( Value : Int) : Int
8934:  Func SAR_9( Value : Int) : Int
8935:  Func SAR_11( Value : Int) : Int
8936:  Func SAR_12( Value : Int) : Int
8937:  Func SAR_13( Value : Int) : Int
8938:  Func SAR_14( Value : Int) : Int
8939:  Func SAR_15( Value : Int) : Int
8940:  Func SAR_16( Value : Int) : Int
8941:  Func ColorSwap( WinColor : TColor) : TColor32
8942: end;
8943:
8944: Proc SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8945: begin
8946:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8947:  Proc CopyComponents(Dst,Src:TCustomBitmap32;Components:TColor32Components);
8948:  Proc CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8949:  Proc AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8950:  Proc ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha :Bool)
8951:  Proc IntensityToAlpha( Dst, Src : TCustomBitmap32)
8952:  Proc Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8953:  Proc InvertRGB( Dst, Src : TCustomBitmap32)
8954:  Proc ApplyLUT(Dst,Src:TCustomBitmap32;const LUT:TLUT8;PreserveAlpha:Boolean)
8955:  Proc ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8956:  Func CreateBitmask( Components : TColor32Components) : TColor32
8957:  Proc ApplyBitmask(Dst:TCustomBitmap32;DstX,DstY:Int;Src:TCustomBitmap32;SrcRect:TRect;
      Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8958:  Proc ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8959:  Proc CheckParams( Dst, Src : TCustomBitmap32; ResizeDst :Bool)
8960: end;
8961:
8962: Proc SIRegister_JclNTFS(CL: TPSPascalCompiler);
8963: begin
8964:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8965:   AddTypeS('TFileCompressionState','(fcNoCompression,fcDefaultCompression,fcLZNT1Compression )
8966:  Func NtfsGetCompression const FileName:str; var State : Short):Bool;
8967:  Func NtfsGetCompression1( const FileName :Str) : TFileCompressionState;
8968:  Func NtfsSetCompression const FileName:str; const State:Short) :Bool
8969:  Proc NtfsSetFileCompression(const FileName:Str; const State: TFileCompressionState)
8970:  Proc NtfsSetDirectoryTreeCompression(const Directory:Str; const State : TFileCompressionState)
8971:  Proc NtfsSetDefaultFileCompression(const Directory:Str; const State:TFileCompressionState)
8972:  Proc NtfsSetPathCompression(const Path:str;const State:TFileCompressionState;Recursive:Bool;
8973:  //AddTypeS('TNtfsAllocRanges', 'record Entries : Int; Data : PFileAlloca'
8974:  //+'tedRangeBuffer; MoreData :Bool; end
8975:  Func NtfsSetSparse( const FileName :Str) :Bool
8976:  Func NtfsZeroDataByHandle(const Handle:THandle;const First,Last:Int64): Bool
8977:  Func NtfsZeroDataByName(const FileName:str;const First,Last:Int64): Bool
8978:  //Func NtfsQueryAllocRanges(const FileName:str;Offset,Count:Int64;var Ranges:TNtfsAllocRanges):Boolean;
8979:  //Func NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges; Index:Int):TFileAllocatedRangeBuffer
8980:  Func NtfsSparseStreamsSupported( const Volume :Str) :Bool
```

```
8981:  Func NtfsGetSparse( const FileName :Str) :Bool
8982:  Func NtfsDeleteReparsePoint(const FileName:str; ReparseTag:DWORD):Boolean
8983:  Func NtfsSetReparsePoint(const FileName:str;var ReparseData,Size:Longword): Bool
8984: //Func NtfsGetReparsePoint(const FileName:str;var ReparseData:TReparseGuidDataBuffer):Bool
8985:  Func NtfsGetReparseTag( const Path :Str; var Tag : DWORD) :Bool
8986:  Func NtfsReparsePointsSupported( const Volume :Str) :Bool
8987:  Func NtfsFileHasReparsePoint( const Path :Str) :Bool
8988:  Func NtfsIsFolderMountPoint( const Path :Str) :Bool
8989:  Func NtfsMountDeviceAsDrive( const Device :Str; Drive : Char):Boolean
8990:  Func NtfsMountVolume(const Volume:Char; const MountPoint:str):Bool
8991:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8992:  Func NtfsOpLockAckClosePending(Handle:THandle;Overlapped:TOverlapped):boolean
8993:  Func NtfsOpLockBreakAckNo2(Handle:THandle;Overlapped:TOverlapped):Bool
8994:  Func NtfsOpLockBreakAcknowledge(Handl:THandle;Overlapped:TOverlapped):Boolean
8995:  Func NtfsOpLockBreakNotify(Handle:THandle;Overlapped:TOverlapped) :Bool
8996:  Func NtfsRequestOpLock(Handle:THandle;Kind:TOpLock;Overlapped:TOverlapped): Bool
8997:  Func NtfsCreateJunctionPoint( const Source, Destination :Str) :Bool
8998:  Func NtfsDeleteJunctionPoint( const Source :Str) :Bool
8999:  Func NtfsGetJunctionPointDestination(const Source:Str;var Destination:str):Bool
9000:   AddTypeS('TStreamId','( siInvalid, siStandard, siExtendedAttribute, siSec'
9001:    +'urity, siAlternate,siHardLink,siProperty,siObjectIdentifier,siReparsePoints,siSparseFile )
9002:   AddTypeS('TStreamIds', 'set of TStreamId
9003:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
9004:    +': ___Pointer; StreamIds : TStreamIds; end
9005:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
9006:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
9007:  Func NtfsFindFirstStream(const FileName:str;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
9008:  Func NtfsFindNextStream( var Data : TFindStreamData) :Bool
9009:  Func NtfsFindStreamClose( var Data : TFindStreamData) :Bool
9010:  Func NtfsCreateHardLink(const LinkFileName,ExistingFileName:str):Boolean
9011:   AddTypeS('TNtfsHardLinkInfo','record LinkCount:Card;FileIndex:Int64; end
9012:  Func NtfsGetHardLinkInfo( const FileName :Str; var Info : TNtfsHardLinkInfo) :Bool
9013:  Func NtfsFindHardLinks(const Path:str;const FileIndexHigh,FileIndexLow:Card;const List:TStrings):Bool;
9014:  Func NtfsDeleteHardLinks( const FileName :Str) :Bool
9015:  Func JclAppInstances : TJclAppInstances;
9016:  Func JclAppInstances1( const UniqueAppIdGuidStr :Str) : TJclAppInstances;
9017:  Func ReadMessageCheck( var Message:TMessage;const IgnoredOriginatorWnd:HWND): TJclAppInstDataKind
9018:  Proc ReadMessageData(const Message:TMessage;var Data:___Pointer;var Size:Int)
9019:  Proc ReadMessageString( const Message : TMessage; var S :Str)
9020:  Proc ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
9021:
9022: (*----------------------------------------------------------------------------*)
9023: Proc SIRegister_JclGraphics(CL: TPSPascalCompiler);
9024: begin
9025:   FindClass('TOBJECT'),'EJclGraphicsError
9026:   TDynDynIntArrayArray', 'array of TDynIntArray
9027:   TDynPointArray', 'array of TPoint
9028:   TDynDynPointArrayArray', 'array of TDynPointArray
9029:   TPointF', 'record X : Single; Y : Single; end
9030:   TDynPointArrayF', 'array of TPointF
9031:   TDrawMode2', '( dmOpaque, dmBlend )
9032:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
9033:   TConversionKind', '(ckRed,ckGreen,ckBlue,ckAlpha ckUniformRGB, ckWeightedRGB )
9034:   TResamplingFilter','(rfBox,rfTriangle,rfHermite,rfBell,rfSpline,rfLanczos3,rfMitchell )
9035:   TMatrix3d', 'record array[0..2,0..2] of extended end
9036:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
9037:   TScanLine', 'array of Int
9038:   TScanLines', 'array of TScanLine
9039:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
9040:   TGradientDirection', '( gdVertical, gdHorizontal )
9041:   TPolyFillMode', '( fmAlternate, fmWinding )
9042:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
9043:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
9044:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
9045:   SIRegister_TJclDesktopCanvas(CL);
9046:   FindClass('TOBJECT'),'TJclRegion
9047:   SIRegister_TJclRegionInfo(CL);
9048:   SIRegister_TJclRegion(CL);
9049:   SIRegister_TJclThreadPersistent(CL);
9050:   SIRegister_TJclCustomMap(CL);
9051:   SIRegister_TJclBitmap32(CL);
9052:   SIRegister_TJclByteMap(CL);
9053:   SIRegister_TJclTransformation(CL);
9054:   SIRegister_TJclLinearTransformation(CL);
9055:  Proc Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
9056:  Proc Stretch1(NewWidth,NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
9057:  Proc DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Int)
9058:  Func GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
9059:  Proc BitmapToJPeg( const FileName :Str)
9060:  Proc JPegToBitmap( const FileName :Str)
9061:  Func ExtractIconCount( const FileName :Str) : Int
9062:  Func BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Int) : HICON
9063:  Func IconToBitmapJ( Icon : HICON) : HBITMAP
9064:  Proc BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
9065:  Proc
      StretchTransfer(Dst:TJclBitmap32;DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter;
      CombineOp:TDrawMode)
9066:  Proc Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
```

```
9067:  Proc SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
9068:  Func FillGradient(DC:HDC;ARect:TRect;ColorCount:Int;StartColor,
      EndColor:TColor;ADirection:TGradientDirection):Bool;
9069:  Func CreateRegionFromBitmap(Bitmap:TBitmap;RegionColor:TColor;RegionBitmapMode:TJclRegionBitmapMode): HRGN
9070:  Proc ScreenShot(bm:TBitmap; Left, Top, Width, Height : Int; Window : HWND);
9071:  Proc ScreenShot1( bm : TBitmap);
9072:  Proc PolyLineTS(Bitmap TJclBitmap32;const Points:TDynPointArray;Color: TColor32)
9073:  Proc PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
9074:  Proc PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
9075:  Proc PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
9076:  Proc PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
9077:  Proc PolygonFS(Bitmap:TJclBitmap32;const Points:TDynPointArrayF;Color: TColor32)
9078:  Proc PolyPolygonTS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
9079:  Proc PolyPolygonAS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
9080:  Proc PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Colr:TColr32);
9081:  Proc AlphaToGrayscale( Dst, Src : TJclBitmap32)
9082:  Proc IntensityToAlpha( Dst, Src : TJclBitmap32)
9083:  Proc Invert( Dst, Src : TJclBitmap32)
9084:  Proc InvertRGB( Dst, Src : TJclBitmap32)
9085:  Proc ColorToGrayscale( Dst, Src : TJclBitmap32)
9086:  Proc ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
9087:  Proc SetGamma( Gamma : Single)
9088: end;
9089:
9090: (*----------------------------------------------------------------------------*)
9091: Proc SIRegister_JclSynch(CL: TPSPascalCompiler);
9092: begin
9093:  Func LockedAdd( var Target : Int; Value : Int) : Int
9094:  Func LockedCompareExchange( var Target : Int; Exch, Comp : Int) : Int;
9095:  Func LockedCompareExchange1(var Target: ___Pointer;Exch,Comp:___Pointer): Pointer;
9096:  Func LockedDec( var Target : Int) : Int
9097:  Func LockedExchange( var Target : Int; Value : Int) : Int
9098:  Func LockedExchangeAdd( var Target : Int; Value : Int) : Int
9099:  Func LockedExchangeDec( var Target : Int) : Int
9100:  Func LockedExchangeInc( var Target : Int) : Int
9101:  Func LockedExchangeSub( var Target : Int; Value : Int) : Int
9102:  Func LockedInc( var Target : Int) : Int
9103:  Func LockedSub( var Target : Int; Value : Int) : Int
9104:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
9105:   SIRegister_TJclDispatcherObject(CL);
9106:  Func WaitForMultipleObjects(const Objects:array of TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardin):Card;
9107:  Func WaitAlertableForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Card):Card
9108:   SIRegister_TJclCriticalSection(CL);
9109:   SIRegister_TJclCriticalSectionEx(CL);
9110:   SIRegister_TJclEvent(CL);
9111:   SIRegister_TJclWaitableTimer(CL);
9112:   SIRegister_TJclSemaphore(CL);
9113:   SIRegister_TJclMutex(CL);
9114:   POptexSharedInfo', '^TOptexSharedInfo // will not work
9115:   TOptexSharedInfo,record SpinCount:Int;LockCount:Int;ThreadId:Longword;RecursionCount:Int;end
9116:   SIRegister_TJclOptex(CL);
9117:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
9118:   TMrewThreadInfo','record ThreadId:Longword;RecursionCount:Int;Reader:Bool; end
9119:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
9120:   SIRegister_TJclMultiReadExclusiveWrite(CL);
9121:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
9122:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
9123:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
9124:   PMeteredSection', '^TMeteredSection // will not work
9125:   TMeteredSection', 'record Event:THandle; FileMap:THandle; SharedInfo:PMetSectSharedInfo; end
9126:   SIRegister_TJclMeteredSection(CL);
9127:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
9128:   TMutexInfo', 'record SignalState:Longint; Owned:Boolean; Abandoned:Bool; end
9129:   TSemaphoreCounts', 'record CurrentCount: Longint; MaximumCount: Longint; end
9130:   TTimerInfo', 'record Remaining : TLargeInt; Signaled : LongBool; end
9131:  Func QueryCriticalSection(CS TJclCriticalSection;var Info:TRTLCriticalSection):Boolean
9132:  Func QueryEvent( Handle : THandle; var Info : TEventInfo) :Bool
9133:  Func QueryMutex( Handle : THandle; var Info : TMutexInfo) :Bool
9134:  Func QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) :Bool
9135:  Func QueryTimer( Handle : THandle; var Info : TTimerInfo) :Bool
9136:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
9137:   FindClass('TOBJECT'),'EJclDispatcherObjectError
9138:   FindClass('TOBJECT'),'EJclCriticalSectionError
9139:   FindClass('TOBJECT'),'EJclEventError
9140:   FindClass('TOBJECT'),'EJclWaitableTimerError
9141:   FindClass('TOBJECT'),'EJclSemaphoreError
9142:   FindClass('TOBJECT'),'EJclMutexError
9143:   FindClass('TOBJECT'),'EJclMeteredSectionError
9144: end;
9145:
9146: //***************************unit uPSI_mORMotReport;
9147: Proc SetCurrentPrinterAsDefault
9148: Func CurrentPrinterName :Str
9149: Func mCurrentPrinterPaperSize :Str
9150: Proc UseDefaultPrinter
9151:
9152: Proc SIRegisterTSTREAM(Cl: TPSPascalCompiler);
9153: begin
```

```
9154: with FindClass('TOBJECT'), 'TStream') do begin
9155:   IsAbstract := True;
9156:  //RegisterMethod('Func Read( var Buffer, Count : Longint) : Longint
9157:  // RegisterMethod('Func Write( const Buffer, Count : Longint) : Longint
9158:   Func Read(Buffer:str;Count:LongInt):LongInt
9159:   Func Write(Buffer:str;Count:LongInt):LongInt
9160:   Func ReadString(Buffer:str;Count:LongInt):LongInt     //FileStream
9161:   Func WriteString(Buffer:str;Count:LongInt):LongInt
9162:   Func ReadInt(Buffer:Int;Count:LongInt):LongInt
9163:   Func WriteInt(Buffer:Int;Count:LongInt):LongInt
9164:   Func ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt;
9165:   Func WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt;
9166:
9167:   Proc ReadAB(Buffer: TByteArray;Count:LongInt)
9168:   Proc WriteAB(Buffer: TByteArray;Count:LongInt)
9169:   Proc ReadABD(Buffer: TByteDynArray;Count:LongInt)
9170:   Proc WriteABD(Buffer: TByteDynArray;Count:LongInt)
9171:   Proc ReadAC(Buffer: TCharArray;Count:LongInt)
9172:   Proc WriteAC(Buffer: TCharArray;Count:LongInt)
9173:   Proc ReadACD(Buffer: TCharDynArray;Count:LongInt)
9174:   Proc WriteACD(Buffer: TCharDynArray;Count:LongInt)
9175:
9176:   Func Seek(Offset:LongInt;Origin:Word):LongInt
9177:   Proc ReadBuffer(Buffer:str;Count:LongInt)
9178:   Proc WriteBuffer(Buffer:str;Count:LongInt)
9179:   Proc ReadBufferInt(Buffer:Int;Count:LongInt);
9180:   Proc WriteBufferInt(Buffer:Int;Count:LongInt);
9181:   Proc ReadBufferFloat(Buffer:Double;Count:LongInt);
9182:   Proc WriteBufferFloat(Buffer:Double;Count:LongInt);
9183:
9184:   Proc ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9185:   Proc WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9186:   Proc ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9187:   Proc WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9188:   Proc ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9189:   Proc WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9190:   Proc ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9191:   Proc WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9192:   Proc ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9193:   Proc WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9194:   Proc ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9195:   Proc WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9196:   Proc ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9197:   Proc WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9198:
9199:   Proc ReadBufferP(Buffer: PChar;Count:LongInt)
9200:   Proc WriteBufferP(Buffer: PChar;Count:LongInt)
9201:   Proc ReadBufferO(Buffer: TObject;Count:LongInt);
9202:   Proc WriteBufferO(Buffer: TObject;Count:LongInt);
9203:   //READBUFFERAC
9204:   Func InstanceSize: Longint
9205:   Proc FixupResourceHeader( FixupInfo : Int)
9206:   Proc ReadResHeader
9207:   {$IFDEF DELPHI4UP}
9208:   Func CopyFrom(Source:TStream;Count:Int64):LongInt
9209:   {$ELSE}
9210:   Func CopyFrom(Source:TStream;Count:Int):LongInt
9211:   {$ENDIF}
9212:   RegisterProperty('Position', 'LongInt', iptrw);
9213:   RegisterProperty('Size', 'LongInt', iptrw);
9214: end;
9215: end;
9216:
9217: { ****************************************************************
9218:   Unit DMATH - Interface for DMATH.DLL
9219:   **************************************************************** }
9220: // see more docs/dmath_manual.pdf
9221:
9222: Func InitEval : Int
9223: Proc SetVariable( VarName : Char; Value : Float)
9224: Proc SetFunction( FuncName :Str; Wrapper : TWrapper)
9225: Func Eval( ExpressionString :Str) : Float
9226:
9227: unit dmath; //types are in built, others are external in DLL
9228: interface
9229: {$IFDEF DELPHI}
9230: uses
9231:   StdCtrls, Graphics;
9232: {$ENDIF}
9233: { ----------------------------------------------------------------
9234:   Types and constants
9235:   ---------------------------------------------------------------- }
9236: {$i types.inc}
9237: { ----------------------------------------------------------------
9238:   Error handling
9239:   ---------------------------------------------------------------- }
9240: Proc SetErrCode(ErrCode : Int); external 'dmath';
9241: { Sets the error code }
9242: Func DefaultVal(ErrCode : Int; DefVal : Float) : Float; external 'dmath';
```

```
9243: { Sets error code and default Func value }
9244: Func MathErr : Int; external 'dmath';
9245: { Returns the error code }
9246: { ----------------------------------------------------------------
9247:   Dynamic arrays
9248:   ---------------------------------------------------------------- }
9249: Proc SetAutoInit(AutoInit :Bool); external 'dmath';
9250: { Sets the auto-initialization of arrays }
9251: Proc DimVector(var V : TVector; Ub : Int); external 'dmath';
9252: { Creates floating point vector V[0..Ub] }
9253: Proc DimIntVector(var V : TIntVector; Ub : Int); external 'dmath';
9254: { Creates Int vector V[0..Ub] }
9255: Proc DimCompVector(var V : TCompVector; Ub : Int);external 'dmath';
9256: { Creates complex vector V[0..Ub] }
9257: Proc DimBoolVector(var V : TBoolVector; Ub : Int);external 'dmath';
9258: { Creates boolean vector V[0..Ub] }
9259: Proc DimStrVector(var V : TStrVector; Ub : Int); external 'dmath';
9260: { Creates string vector V[0..Ub] }
9261: Proc DimMatrix(var A : TMatrix; Ub1, Ub2 : Int); external 'dmath';
9262: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9263: Proc DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Int); external 'dmath';
9264: { Creates Int matrix A[0..Ub1, 0..Ub2] }
9265: Proc DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Int);external 'dmath';
9266: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9267: Proc DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Int);external 'dmath';
9268: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9269: Proc DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Int); external 'dmath';
9270: { Creates string matrix A[0..Ub1, 0..Ub2] }
9271: { ----------------------------------------------------------------
9272:   Minimum, maximum, sign and exchange
9273:   ---------------------------------------------------------------- }
9274: Func FMin(X, Y : Float) : Float; external 'dmath';
9275: { Minimum of 2 reals }
9276: Func FMax(X, Y : Float) : Float; external 'dmath';
9277: { Maximum of 2 reals }
9278: Func IMin(X, Y : Int) : Int; external 'dmath';
9279: { Minimum of 2 Ints }
9280: Func IMax(X, Y : Int) : Int; external 'dmath';
9281: { Maximum of 2 Ints }
9282: Func Sgn(X : Float) : Int; external 'dmath';
9283: { Sign (returns 1 if X = 0) }
9284: Func Sgn0(X : Float) : Int; external 'dmath';
9285: { Sign (returns 0 if X = 0) }
9286: Func DSgn(A, B : Float) : Float; external 'dmath';
9287: { Sgn(B) * |A| }
9288: Proc FSwap(var X, Y : Float); external 'dmath';
9289: { Exchange 2 reals }
9290: Proc ISwap(var X, Y : Int); external 'dmath';
9291: { Exchange 2 Ints }
9292: { ----------------------------------------------------------------
9293:   Rounding functions
9294:   ---------------------------------------------------------------- }
9295: Func RoundN(X : Float; N : Int) : Float; external 'dmath';
9296: { Rounds X to N decimal places }
9297: Func Ceil(X : Float) : Int; external 'dmath';
9298: { Ceiling Func }
9299: Func Floor(X : Float) : Int; external 'dmath';
9300: { Floor Func }
9301: { ----------------------------------------------------------------
9302:   Logarithms, exponentials and power
9303:   ---------------------------------------------------------------- }
9304: Func Expo(X : Float) : Float; external 'dmath';
9305: { Exponential }
9306: Func Exp2(X : Float) : Float; external 'dmath';
9307: { 2^X }
9308: Func Exp10(X : Float) : Float; external 'dmath';
9309: { 10^X }
9310: Func Log(X : Float) : Float; external 'dmath';
9311: { Natural log }
9312: Func Log2(X : Float) : Float; external 'dmath';
9313: { Log, base 2 }
9314: Func Log10(X : Float) : Float; external 'dmath';
9315: { Decimal log }
9316: Func LogA(X, A : Float) : Float; external 'dmath';
9317: { Log, base A }
9318: Func IntPower(X : Float; N : Int) : Float; external 'dmath';
9319: { X^N }
9320: Func Power(X, Y : Float) : Float; external 'dmath';
9321: { X^Y, X >= 0 }
9322: { ----------------------------------------------------------------
9323:   Trigonometric functions
9324:   ---------------------------------------------------------------- }
9325: Func Pythag(X, Y : Float) : Float; external 'dmath';
9326: { Sqrt(X^2 + Y^2) }
9327: Func FixAngle(Theta : Float) : Float; external 'dmath';
9328: { Set Theta in -Pi..Pi }
9329: Func Tan(X : Float) : Float; external 'dmath';
9330: { Tangent }
9331: Func ArcSin(X : Float) : Float; external 'dmath';
```

```
9332: { Arc sinus }
9333: Func ArcCos(X : Float) : Float; external 'dmath';
9334: { Arc cosinus }
9335: Func ArcTan2(Y, X : Float) : Float; external 'dmath';
9336: { Angle (Ox, OM) with M(X,Y) }
9337: { ----------------------------------------------------------------
9338:   Hyperbolic functions
9339:   ---------------------------------------------------------------- }
9340: Func Sinh(X : Float) : Float; external 'dmath';
9341: { Hyperbolic sine }
9342: Func Cosh(X : Float) : Float; external 'dmath';
9343: { Hyperbolic cosine }
9344: Func Tanh(X : Float) : Float; external 'dmath';
9345: { Hyperbolic tangent }
9346: Func ArcSinh(X : Float) : Float; external 'dmath';
9347: { Inverse hyperbolic sine }
9348: Func ArcCosh(X : Float) : Float; external 'dmath';
9349: { Inverse hyperbolic cosine }
9350: Func ArcTanh(X : Float) : Float; external 'dmath';
9351: { Inverse hyperbolic tangent }
9352: Proc SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9353: { Sinh & Cosh }
9354: { ----------------------------------------------------------------
9355:   Gamma Func and related functions
9356:   ---------------------------------------------------------------- }
9357: Func Fact(N : Int) : Float; external 'dmath';
9358: { Factorial }
9359: Func SgnGamma(X : Float) : Int; external 'dmath';
9360: { Sign of Gamma Func }
9361: Func Gamma(X : Float) : Float; external 'dmath';
9362: { Gamma Func }
9363: Func LnGamma(X : Float) : Float; external 'dmath';
9364: { Logarithm of Gamma Func }
9365: Func Stirling(X : Float) : Float; external 'dmath';
9366: { Stirling's formula for the Gamma Func }
9367: Func StirLog(X : Float) : Float; external 'dmath';
9368: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9369: Func DiGamma(X : Float ) : Float; external 'dmath';
9370: { Digamma Func }
9371: Func TriGamma(X : Float ) : Float; external 'dmath';
9372: { Trigamma Func }
9373: Func IGamma(A, X : Float) : Float; external 'dmath';
9374: { Incomplete Gamma function}
9375: Func JGamma(A, X : Float) : Float; external 'dmath';
9376: { Complement of incomplete Gamma Func }
9377: Func InvGamma(A, P : Float) : Float; external 'dmath';
9378: { Inverse of incomplete Gamma Func }
9379: Func Erf(X : Float) : Float; external 'dmath';
9380: { Error Func }
9381: Func Erfc(X : Float) : Float; external 'dmath';
9382: { Complement of error Func }
9383: { ----------------------------------------------------------------
9384:   Beta Func and related functions
9385:   ---------------------------------------------------------------- }
9386: Func Beta(X, Y : Float) : Float; external 'dmath';
9387: { Beta Func }
9388: Func IBeta(A, B, X : Float) : Float; external 'dmath';
9389: { Incomplete Beta Func }
9390: Func InvBeta(A, B, Y : Float) : Float; external 'dmath';
9391: { Inverse of incomplete Beta Func }
9392: { ----------------------------------------------------------------
9393:   Lambert's function
9394:   ---------------------------------------------------------------- }
9395: Func LambertW(X : Float; UBranch, Offset :Bool) : Float; external 'dmath';
9396:  ----------------------------------------------------------------
9397:   Binomial distribution
9398:   ---------------------------------------------------------------- }
9399: Func Binomial(N, K : Int) : Float; external 'dmath';
9400: { Binomial coefficient C(N,K) }
9401: Func PBinom(N : Int; P : Float; K : Int) : Float; external 'dmath';
9402: { Probability of binomial distribution }
9403: Func FBinom(N : Int; P : Float; K : Int) : Float; external 'dmath';
9404: { Cumulative probability for binomial distrib. }
9405: { ----------------------------------------------------------------
9406:   Poisson distribution
9407:   ---------------------------------------------------------------- }
9408: Func PPoisson(Mu : Float; K : Int) : Float; external 'dmath';
9409: { Probability of Poisson distribution }
9410: Func FPoisson(Mu : Float; K : Int) : Float; external 'dmath';
9411: { Cumulative probability for Poisson distrib. }
9412: { ----------------------------------------------------------------
9413:   Exponential distribution
9414:   ---------------------------------------------------------------- }
9415: Func DExpo(A, X : Float) : Float; external 'dmath';
9416: { Density of exponential distribution with parameter A }
9417: Func FExpo(A, X : Float) : Float; external 'dmath';
9418: { Cumulative probability Func for exponential dist. with parameter A }
9419: { ----------------------------------------------------------------
9420:   Standard normal distribution
```

```
9421:   ---------------------------------------------------------------- }
9422: Func DNorm(X : Float) : Float; external 'dmath';
9423: { Density of standard normal distribution }
9424: Func FNorm(X : Float) : Float; external 'dmath';
9425: { Cumulative probability for standard normal distrib. }
9426: Func PNorm(X : Float) : Float; external 'dmath';
9427: { Prob(|U| > X) for standard normal distrib. }
9428: Func InvNorm(P : Float) : Float; external 'dmath';
9429: { Inverse of standard normal distribution }
9430: { ----------------------------------------------------------------
9431:   Student's distribution
9432:   ---------------------------------------------------------------- }
9433: Func DStudent(Nu : Int; X : Float) : Float; external 'dmath';
9434: { Density of Student distribution with Nu d.o.f. }
9435: Func FStudent(Nu : Int; X : Float) : Float; external 'dmath';
9436: { Cumulative probability for Student distrib. with Nu d.o.f. }
9437: Func PStudent(Nu : Int; X : Float) : Float; external 'dmath';
9438: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9439: Func InvStudent(Nu : Int; P : Float) : Float; external 'dmath';
9440: { Inverse of Student's t-distribution Func }
9441: { ----------------------------------------------------------------
9442:   Khi-2 distribution
9443:   ---------------------------------------------------------------- }
9444: Func DKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9445: { Density of Khi-2 distribution with Nu d.o.f. }
9446: Func FKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9447: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9448: Func PKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9449: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9450: Func InvKhi2(Nu : Int; P : Float) : Float; external 'dmath';
9451: { Inverse of Khi-2 distribution Func }
9452: { ----------------------------------------------------------------
9453:   Fisher-Snedecor distribution
9454:   ---------------------------------------------------------------- }
9455: Func DSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9456: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9457: Func FSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9458: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9459: Func PSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9460: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9461: Func InvSnedecor(Nu1, Nu2 : Int; P : Float) : Float; external 'dmath';
9462: { Inverse of Snedecor's F-distribution Func }
9463: { ----------------------------------------------------------------
9464:   Beta distribution
9465:   ---------------------------------------------------------------- }
9466: Func DBeta(A, B, X : Float) : Float; external 'dmath';
9467: { Density of Beta distribution with parameters A and B }
9468: Func FBeta(A, B, X : Float) : Float; external 'dmath';
9469: { Cumulative probability for Beta distrib. with param. A and B }
9470: { ----------------------------------------------------------------
9471:   Gamma distribution
9472:   ---------------------------------------------------------------- }
9473: Func DGamma(A, B, X : Float) : Float; external 'dmath';
9474: { Density of Gamma distribution with parameters A and B }
9475: Func FGamma(A, B, X : Float) : Float; external 'dmath';
9476: { Cumulative probability for Gamma distrib. with param. A and B }
9477: { ----------------------------------------------------------------
9478:   Expression evaluation
9479:   ---------------------------------------------------------------- }
9480: Func InitEval : Int; external 'dmath';
9481: { Initializes built-in functions and returns their number }
9482: Func Eval(ExpressionString :Str) : Float; external 'dmath';
9483: { Evaluates an expression at run-time }
9484: Proc SetVariable(VarName : Char; Value : Float); external 'dmath';
9485: { Assigns a value to a variable }
9486: Proc SetFunction(FuncName :Str; Wrapper : TWrapper); external 'dmath';
9487: { Adds a Func to the parser }
9488: { ----------------------------------------------------------------
9489:   Matrices and linear equations
9490:   ---------------------------------------------------------------- }
9491: Proc GaussJordan(A            : TMatrix;
9492:                       Lb, Ub1, Ub2 : Int;
9493:                       var Det      : Float); external 'dmath';
9494: { Transforms a matrix according to the Gauss-Jordan method }
9495: Proc LinEq(A       : TMatrix;
9496:                 B       : TVector;
9497:                 Lb, Ub  : Int;
9498:                 var Det : Float); external 'dmath';
9499: { Solves a linear system according to the Gauss-Jordan method }
9500: Proc Cholesky(A, L : TMatrix; Lb, Ub : Int); external 'dmath';
9501: { Cholesky factorization of a positive definite symmetric matrix }
9502: Proc LU_Decomp(A : TMatrix; Lb, Ub : Int); external 'dmath';
9503: { LU decomposition }
9504: Proc LU_Solve(A      : TMatrix;
9505:                   B      : TVector;
9506:                   Lb, Ub : Int;
9507:                   X      : TVector); external 'dmath';
9508: { Solution of linear system from LU decomposition }
9509: Proc QR_Decomp(A            : TMatrix;
```

```
                    Lb, Ub1, Ub2 : Int;
                    R            : TMatrix); external 'dmath';
{ QR decomposition }
Proc QR_Solve(Q, R         : TMatrix;
                   B            : TVector;
                   Lb, Ub1, Ub2 : Int;
                   X            : TVector); external 'dmath';
{ Solution of linear system from QR decomposition }
Proc SV_Decomp(A            : TMatrix;
                    Lb, Ub1, Ub2 : Int;
                    S            : TVector;
                    V            : TMatrix); external 'dmath';
{ Singular value decomposition }
Proc SV_SetZero(S      : TVector;
                     Lb, Ub : Int;
                     Tol    : Float); external 'dmath';
{ Set lowest singular values to zero }
Proc SV_Solve(U            : TMatrix;
                   S            : TVector;
                   V            : TMatrix;
                   B            : TVector;
                   Lb, Ub1, Ub2 : Int;
                   X            : TVector); external 'dmath';
{ Solution of linear system from SVD }
Proc SV_Approx(U            : TMatrix;
                    S            : TVector;
                    V            : TMatrix;
                    Lb, Ub1, Ub2 : Int;
                    A            : TMatrix); external 'dmath';
{ Matrix approximation from SVD }
Proc EigenVals(A      : TMatrix;
                    Lb, Ub : Int;
                    Lambda : TCompVector); external 'dmath';
{ Eigenvalues of a general square matrix }
Proc EigenVect(A      : TMatrix;
                    Lb, Ub : Int;
                    Lambda : TCompVector;
                    V      : TMatrix); external 'dmath';
{ Eigenvalues and eigenvectors of a general square matrix }
Proc EigenSym(A      : TMatrix;
                   Lb, Ub : Int;
                   Lambda : TVector;
                   V      : TMatrix); external 'dmath';
{ Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
Proc Jacobi(A               : TMatrix;
                 Lb, Ub, MaxIter : Int;
                 Tol             : Float;
                 Lambda          : TVector;
                 V               : TMatrix); external 'dmath';
{ Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
{ ----------------------------------------------------------------
  Optimization
  ---------------------------------------------------------------- }
Proc MinBrack(Func                    : TFunc;
                   var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
{ Brackets a minimum of a Func }
Proc GoldSearch(Func           : TFunc;
                     A, B           : Float;
                     MaxIter        : Int;
                     Tol            : Float;
                     var Xmin, Ymin : Float); external 'dmath';
{ Minimization of a Func of one variable (golden search) }
Proc LinMin(Func      : TFuncNVar;
                 X, DeltaX : TVector;
                 Lb, Ub    : Int;
                 var R     : Float;
                 MaxIter   : Int;
                 Tol       : Float;
                 var F_min : Float); external 'dmath';
{ Minimization of a Func of several variables along a line }
Proc Newton(Func      : TFuncNVar;
                 HessGrad  : THessGrad;
                 X         : TVector;
                 Lb, Ub    : Int;
                 MaxIter   : Int;
                 Tol       : Float;
                 var F_min : Float;
                 G         : TVector;
                 H_inv     : TMatrix;
                 var Det   : Float); external 'dmath';
{ Minimization of a Func of several variables (Newton's method) }
Proc SaveNewton(FileName :Str); external 'dmath';
{ Save Newton iterations in a file }
Proc Marquardt(Func      : TFuncNVar;
                    HessGrad  : THessGrad;
                    X         : TVector;
                    Lb, Ub    : Int;
                    MaxIter   : Int;
                    Tol       : Float;
```

```
                    var F_min : Float;
                    G         : TVector;
                    H_inv     : TMatrix;
                    var Det   : Float); external 'dmath';
{ Minimization of a Func of several variables (Marquardt's method) }
Proc SaveMarquardt(FileName :Str); external 'dmath';
{ Save Marquardt iterations in a file }
Proc BFGS(Func      : TFuncNVar;
               Gradient  : TGradient;
               X         : TVector;
               Lb, Ub    : Int;
               MaxIter   : Int;
               Tol       : Float;
               var F_min : Float;
               G         : TVector;
               H_inv     : TMatrix); external 'dmath';
{ Minimization of a Func of several variables (BFGS method) }
Proc SaveBFGS(FileName :Str); external 'dmath';
{ Save BFGS iterations in a file }
Proc Simplex(Func      : TFuncNVar;
                  X         : TVector;
                  Lb, Ub    : Int;
                  MaxIter   : Int;
                  Tol       : Float;
                  var F_min : Float); external 'dmath';
{ Minimization of a Func of several variables (Simplex) }
Proc SaveSimplex(FileName :Str); external 'dmath';
{ Save Simplex iterations in a file }
{ ----------------------------------------------------------------
  Nonlinear equations
  ---------------------------------------------------------------- }
Proc RootBrack(Func              : TFunc;
                    var X, Y, FX, FY : Float); external 'dmath';
{ Brackets a root of Func Func between X and Y }
Proc Bisect(Func     : TFunc;
                 var X, Y : Float;
                 MaxIter  : Int;
                 Tol      : Float;
                 var F    : Float); external 'dmath';
{ Bisection method }
Proc Secant(Func     : TFunc;
                 var X, Y : Float;
                 MaxIter  : Int;
                 Tol      : Float;
                 var F    : Float); external 'dmath';
{ Secant method }
Proc NewtEq(Func, Deriv : TFunc;
                 var X       : Float;
                 MaxIter     : Int;
                 Tol         : Float;
                 var F       : Float); external 'dmath';
{ Newton-Raphson method for a single nonlinear equation }
Proc NewtEqs(Equations : TEquations;
                  Jacobian  : TJacobian;
                  X, F      : TVector;
                  Lb, Ub    : Int;
                  MaxIter   : Int;
                  Tol       : Float); external 'dmath';
{ Newton-Raphson method for a system of nonlinear equations }
Proc Broyden(Equations : TEquations;
                  X, F      : TVector;
                  Lb, Ub    : Int;
                  MaxIter   : Int;
                  Tol       : Float); external 'dmath';
{ Broyden's method for a system of nonlinear equations }
{ ----------------------------------------------------------------
  Polynomials and rational fractions
  ---------------------------------------------------------------- }
Func Poly(X    : Float;
              Coef : TVector;
              Deg  : Int) : Float; external 'dmath';
{ Evaluates a polynomial }
Func RFrac(X          : Float;
               Coef       : TVector;
               Deg1, Deg2 : Int) : Float; external 'dmath';
{ Evaluates a rational fraction }
Func RootPol1(A, B  : Float;
                  var X : Float) : Int; external 'dmath';
{ Solves the linear equation A + B * X = 0 }
Func RootPol2(Coef : TVector;
                  Z    : TCompVector) : Int; external 'dmath';
{ Solves a quadratic equation }
Func RootPol3(Coef : TVector;
                  Z    : TCompVector) : Int; external 'dmath';
{ Solves a cubic equation }
Func RootPol4(Coef : TVector;
                  Z    : TCompVector) : Int; external 'dmath';
{ Solves a quartic equation }
Func RootPol(Coef : TVector;
```

```
                  Deg  : Int;
                  Z    : TCompVector) : Int; external 'dmath';
{ Solves a polynomial equation }
Func SetRealRoots(Deg : Int;
                       Z   : TCompVector;
                       Tol : Float) : Int; external 'dmath';
{ Set the imaginary part of a root to zero }
Proc SortRoots(Deg : Int;
                     Z   : TCompVector); external 'dmath';
{ Sorts the roots of a polynomial }
{ ----------------------------------------------------------------
  Numerical integration and differential equations
  ---------------------------------------------------------------- }
Func TrapInt(X, Y : TVector; N : Int) : Float; external 'dmath';
{ Integration by trapezoidal rule }
Func GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
{ Integral from A to B }
Func GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
{ Integral from 0 to B }
Func Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
{ Convolution product at time T }
Proc ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Int);external 'dmath';
{ Convolution by trapezoidal rule }
Proc RKF45(F                     : TDiffEqs;
                Neqn                 : Int;
                Y, Yp                : TVector;
                var T                : Float;
                Tout, RelErr, AbsErr : Float;
                var Flag             : Int); external 'dmath';
{ Integration of a system of differential equations }
{ ----------------------------------------------------------------
  Fast Fourier Transform
  ---------------------------------------------------------------- }
Proc FFT(NumSamples        : Int;
               InArray, OutArray : TCompVector); external 'dmath';
{ Fast Fourier Transform }
Proc IFFT(NumSamples        : Int;
               InArray, OutArray : TCompVector); external 'dmath';
{ Inverse Fast Fourier Transform }
Proc FFT_Int(NumSamples     : Int;
                       RealIn, ImagIn : TIntVector;
                       OutArray       : TCompVector); external 'dmath';
{ Fast Fourier Transform for Int data }
Proc FFT_Int_Cleanup; external 'dmath';
{ Clear memory after a call to FFT_Int }
Proc CalcFrequency(NumSamples,
                        FrequencyIndex : Int;
                        InArray        : TCompVector;
                        var FFT        : Complex); external 'dmath';
{ Direct computation of Fourier transform }
{ ----------------------------------------------------------------
  Random numbers
  ---------------------------------------------------------------- }
Proc SetRNG(RNG : RNG_Type); external 'dmath';
{ Select generator }
Proc InitGen(Seed : RNG_IntType); external 'dmath';
{ Initialize generator }
Func IRanGen : RNG_IntType; external 'dmath';
{ 32-bit random Int in [-2^31 .. 2^31 - 1] }
Func IRanGen31 : RNG_IntType; external 'dmath';
{ 31-bit random Int in [0 .. 2^31 - 1] }
Func RanGen1 : Float; external 'dmath';
{ 32-bit random real in [0,1] }
Func RanGen2 : Float; external 'dmath';
{ 32-bit random real in [0,1) }
Func RanGen3 : Float; external 'dmath';
{ 32-bit random real in (0,1) }
Func RanGen53 : Float; external 'dmath';
{ 53-bit random real in [0,1) }
Proc InitMWC(Seed : RNG_IntType); external 'dmath';
{ Initializes the 'Multiply with carry' random number generator }
Func IRanMWC : RNG_IntType; external 'dmath';
{ Returns a 32 bit random number in [-2^31 ; 2^31-1] }
Proc InitMT(Seed : RNG_IntType); external 'dmath';
{ Initializes Mersenne Twister generator with a seed }
Proc InitMTbyArray(InitKey   : array of RNG_LongType;
                        KeyLength : Word); external 'dmath';
{ Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
Func IRanMT : RNG_IntType; external 'dmath';
{ Random Int from MT generator }
Proc InitUVAGbyString(KeyPhrase :Str); external 'dmath';
{ Initializes the UVAG generator with a string }
Proc InitUVAG(Seed : RNG_IntType); external 'dmath';
{ Initializes the UVAG generator with an Int }
Func IRanUVAG : RNG_IntType; external 'dmath';
{ Random Int from UVAG generator }
Func RanGaussStd : Float; external 'dmath';
{ Random number from standard normal distribution }
Func RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
```

```
9777: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9778: Proc RanMult(M      : TVector; L      : TMatrix;
9779:                  Lb, Ub : Int;
9780:                  X      : TVector); external 'dmath';
9781: { Random vector from multinormal distribution (correlated) }
9782: Proc RanMultIndep(M, S   : TVector;
9783:                       Lb, Ub : Int;
9784:                       X      : TVector); external 'dmath';
9785: { Random vector from multinormal distribution (uncorrelated) }
9786: Proc InitMHParams(NCycles, MaxSim, SavedSim : Int); external 'dmath';
9787: { Initializes Metropolis-Hastings parameters }
9788: Proc GetMHParams(var NCycles, MaxSim,SavedSim:Int); external 'dmath';
9789: { Returns Metropolis-Hastings parameters }
9790: Proc Hastings(Func      : TFuncNVar;
9791:                  T         : Float;
9792:                  X         : TVector;
9793:                  V         : TMatrix;
9794:                  Lb, Ub    : Int;
9795:                  Xmat      : TMatrix;
9796:                  X_min     : TVector;
9797:                  var F_min : Float); external 'dmath';
9798: { Simulation of a probability density Func by Metropolis-Hastings }
9799: Proc InitSAParams(NT, NS, NCycles : Int; RT : Float); external 'dmath';
9800: { Initializes Simulated Annealing parameters }
9801: Proc SA_CreateLogFile(FileName :Str); external 'dmath';
9802: { Initializes log file }
9803: Proc SimAnn(Func          : TFuncNVar;
9804:                X, Xmin, Xmax : TVector;
9805:                Lb, Ub        : Int;
9806:                var F_min     : Float); external 'dmath';
9807: { Minimization of a Func of several var. by simulated annealing }
9808: Proc InitGAParams(NP, NG : Int; SR, MR, HR : Float); external 'dmath';
9809: { Initializes Genetic Algorithm parameters }
9810: Proc GA_CreateLogFile(FileName :Str); external 'dmath';
9811: { Initializes log file }
9812: Proc GenAlg(Func          : TFuncNVar;
9813:                X, Xmin, Xmax : TVector;
9814:                Lb, Ub        : Int;
9815:                var F_min     : Float); external 'dmath';
9816: { Minimization of a Func of several var. by genetic algorithm }
9817: { ----------------------------------------------------------------
9818:   Statistics
9819:   ---------------------------------------------------------------- }
9820: Func Mean(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
9821: { Mean of sample X }
9822: Func Min(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
9823: { Minimum of sample X }
9824: Func Max(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
9825: { Maximum of sample X }
9826: Func Median(X : TVector; Lb, Ub : Int; Sorted :Bool) : Float; external 'dmath';
9827: { Median of sample X }
9828: Func StDev(X : TVector; Lb, Ub : Int; M : Float) : Float; external 'dmath';
9829: { Standard deviation estimated from sample X }
9830: Func StDevP(X : TVector; Lb, Ub : Int; M : Float) : Float; external 'dmath';
9831: { Standard deviation of population }
9832: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float; external 'dmath';
9833: { Correlation coefficient }
9834: Func Skewness(X : TVector; Lb, Ub : Int; M,Sigma: Float): Float; external 'dmath';
9835: { Skewness of sample X }
9836: Func Kurtosis(X : TVector; Lb, Ub : Int; M,Sigma: Float): Float; external 'dmath';
9837: { Kurtosis of sample X }
9838: Proc QSort(X : TVector; Lb, Ub : Int); external 'dmath';
9839: { Quick sort (ascending order) }
9840: Proc DQSort(X : TVector; Lb, Ub : Int); external 'dmath';
9841: { Quick sort (descending order) }
9842: Proc Interval(X1, X2              : Float;
9843:                  MinDiv, MaxDiv     : Int;
9844:                  var Min, Max, Step : Float); external 'dmath';
9845: { Determines an interval for a set of values }
9846: Proc AutoScale(X : TVector; Lb, Ub : Int; Scale : TScale;
9847:                   var XMin, XMax, XStep : Float); external 'dmath';
9848: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9849: Proc StudIndep(N1, N2          : Int;
9850:                   M1, M2, S1, S2 : Float;
9851:                   var T          : Float;
9852:                   var DoF        : Int); external 'dmath';
9853: { Student t-test for independent samples }
9854: Proc StudPaired(X, Y    : TVector;
9855:                    Lb, Ub   : Int;
9856:                    var T    : Float;
9857:                    var DoF  : Int); external 'dmath';
9858: { Student t-test for paired samples }
9859: Proc AnOVa1(Ns               : Int;
9860:                N                : TIntVector;
9861:                M, S             : TVector;
9862:                var V_f, V_r, F  : Float;
9863:                var DoF_f, DoF_r : Int); external 'dmath';
9864: { One-way analysis of variance }
9865: Proc AnOVa2(NA, NB, Nobs : Int;
```

```
                  M, S          : TMatrix;
                  V, F          : TVector;
                  DoF           : TIntVector); external 'dmath';
{ Two-way analysis of variance }
Proc Snedecor(N1, N2          : Int;
                    S1, S2          : Float;
                    var F           : Float;
                    var DoF1, DoF2 : Int); external 'dmath';
{ Snedecor's F-test (comparison of two variances) }
Proc Bartlett(Ns        : Int;
                    N         : TIntVector;
                    S         : TVector;
                    var Khi2 : Float;
                    var DoF  : Int); external 'dmath';
{ Bartlett's test (comparison of several variances) }
Proc Mann_Whitney(N1, N2     : Int;
                        X1, X2      : TVector;
                        var U, Eps : Float); external 'dmath';
{ Mann-Whitney test}
Proc Wilcoxon(X, Y       : TVector;
                    Lb, Ub      : Int;
                    var Ndiff  : Int;
                    var T, Eps : Float); external 'dmath';
{ Wilcoxon test }
Proc Kruskal_Wallis(Ns       : Int;
                          N        : TIntVector;
                          X        : TMatrix;
                          var H    : Float;
                          var DoF : Int); external 'dmath';
{ Kruskal-Wallis test }
Proc Khi2_Conform(N_cls     : Int;
                        N_estim  : Int;
                        Obs      : TIntVector;
                        Calc     : TVector;
                        var Khi2 : Float;
                        var DoF  : Int); external 'dmath';
{ Khi-2 test for conformity }
Proc Khi2_Indep(N_lin     : Int;
                      N_col     : Int;
                      Obs       : TIntMatrix;
                      var Khi2 : Float;
                      var DoF  : Int); external 'dmath';
{ Khi-2 test for independence }
Proc Woolf_Conform(N_cls    : Int;
                         N_estim : Int;
                         Obs     : TIntVector;
                         Calc    : TVector;
                         var G   : Float;
                         var DoF : Int); external 'dmath';
{ Woolf's test for conformity }
Proc Woolf_Indep(N_lin    : Int;
                       N_col    : Int;
                       Obs      : TIntMatrix;
                       var G    : Float;
                       var DoF : Int); external 'dmath';
{ Woolf's test for independence }
Proc DimStatClassVector(var C : TStatClassVector;
                                Ub     : Int); external 'dmath';
{ Allocates an array of statistical classes: C[0..Ub] }
Proc Distrib(X        : TVector;
                   Lb, Ub   : Int;
                   A, B, H : Float;
                   C        : TStatClassVector); external 'dmath';
{ Distributes an array X[Lb..Ub] into statistical classes }
{ ----------------------------------------------------------------
  Linear / polynomial regression
  ---------------------------------------------------------------- }
Proc LinFit(X, Y    : TVector;
                  Lb, Ub : Int;
                  B       : TVector;
                  V       : TMatrix); external 'dmath';
{ Linear regression : Y = B(0) + B(1) * X }
Proc WLinFit(X, Y, S : TVector;
                   Lb, Ub   : Int;
                   B        : TVector;
                   V        : TMatrix); external 'dmath';
{ Weighted linear regression : Y = B(0) + B(1) * X }
Proc SVDLinFit(X, Y    : TVector;
                     Lb, Ub : Int;
                     SVDTol : Float;
                     B       : TVector;
                     V       : TMatrix); external 'dmath';
{ Unweighted linear regression by singular value decomposition }
Proc WSVDLinFit(X, Y, S : TVector;
                      Lb, Ub   : Int;
                      SVDTol   : Float;
                      B        : TVector;
                      V        : TMatrix); external 'dmath';
{ Weighted linear regression by singular value decomposition }
```

```
 9955: Proc MulFit(X            : TMatrix;
 9956:                  Y            : TVector;
 9957:                  Lb, Ub, Nvar : Int;
 9958:                  ConsTerm     :Bool;
 9959:                  B            : TVector;
 9960:                  V            : TMatrix); external 'dmath';
 9961: { Multiple linear regression by Gauss-Jordan method }
 9962: Proc WMulFit(X            : TMatrix;
 9963:                   Y, S         : TVector;
 9964:                   Lb, Ub, Nvar : Int;
 9965:                   ConsTerm     :Bool;
 9966:                   B            : TVector;
 9967:                   V            : TMatrix); external 'dmath';
 9968: { Weighted multiple linear regression by Gauss-Jordan method }
 9969: Proc SVDFit(X            : TMatrix;
 9970:                  Y            : TVector;
 9971:                  Lb, Ub, Nvar : Int;
 9972:                  ConsTerm     :Bool;
 9973:                  SVDTol       : Float;
 9974:                  B            : TVector;
 9975:                  V            : TMatrix); external 'dmath';
 9976: { Multiple linear regression by singular value decomposition }
 9977: Proc WSVDFit(X            : TMatrix;
 9978:                   Y, S         : TVector;
 9979:                   Lb, Ub, Nvar : Int;
 9980:                   ConsTerm     :Bool;
 9981:                   SVDTol       : Float;
 9982:                   B            : TVector;
 9983:                   V            : TMatrix); external 'dmath';
 9984: { Weighted multiple linear regression by singular value decomposition }
 9985: Proc PolFit(X, Y        : TVector;
 9986:                  Lb, Ub, Deg : Int;
 9987:                  B           : TVector;
 9988:                  V           : TMatrix); external 'dmath';
 9989: { Polynomial regression by Gauss-Jordan method }
 9990: Proc WPolFit(X, Y, S     : TVector;
 9991:                   Lb, Ub, Deg : Int;
 9992:                   B           : TVector;
 9993:                   V           : TMatrix); external 'dmath';
 9994: { Weighted polynomial regression by Gauss-Jordan method }
 9995: Proc SVDPolFit(X, Y        : TVector;
 9996:                     Lb, Ub, Deg : Int;
 9997:                     SVDTol      : Float;
 9998:                     B           : TVector;
 9999:                     V           : TMatrix); external 'dmath';
10000: { Unweighted polynomial regression by singular value decomposition }
10001: Proc WSVDPolFit(X, Y, S     : TVector;
10002:                      Lb, Ub, Deg : Int;
10003:                      SVDTol      : Float;
10004:                      B           : TVector;
10005:                      V           : TMatrix); external 'dmath';
10006: { Weighted polynomial regression by singular value decomposition }
10007: Proc RegTest(Y, Ycalc : TVector;
10008:                   LbY, UbY : Int;
10009:                   V        : TMatrix;
10010:                   LbV, UbV : Int;
10011:                   var Test : TRegTest); external 'dmath';
10012: { Test of unweighted regression }
10013: Proc WRegTest(Y, Ycalc, S : TVector;
10014:                    LbY, UbY    : Int;
10015:                    V           : TMatrix;
10016:                    LbV, UbV    : Int;
10017:                    var Test    : TRegTest); external 'dmath';
10018: { Test of weighted regression }
10019: { ----------------------------------------------------------------
10020:   Nonlinear regression
10021:   ---------------------------------------------------------------- }
10022: Proc SetOptAlgo(Algo : TOptAlgo); external 'dmath';
10023: { Sets the optimization algorithm for nonlinear regression }
10024: Func GetOptAlgo : TOptAlgo; external 'dmath';
10025: { Returns the optimization algorithm }
10026: Proc SetMaxParam(N : Byte); external 'dmath';
10027: { Sets the maximum number of regression parameters for nonlinear regression }
10028: Func GetMaxParam : Byte; external 'dmath';
10029: { Returns the maximum number of regression parameters for nonlinear regression }
10030: Proc SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
10031: { Sets the bounds on the I-th regression parameter }
10032: Proc GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
10033: { Returns the bounds on the I-th regression parameter }
10034: Proc NLFit(RegFunc   : TRegFunc;
10035:                 DerivProc : TDerivProc;
10036:                 X, Y      : TVector;
10037:                 Lb, Ub    : Int;
10038:                 MaxIter   : Int;
10039:                 Tol       : Float;
10040:                 B         : TVector;
10041:                 FirstPar,
10042:                 LastPar   : Int;
10043:                 V         : TMatrix); external 'dmath';
```

```
10044: { Unweighted nonlinear regression }
10045: Proc WNLFit(RegFunc   : TRegFunc;
10046:                 DerivProc : TDerivProc;
10047:                 X, Y, S   : TVector;
10048:                 Lb, Ub    : Int;
10049:                 MaxIter   : Int;
10050:                 Tol       : Float;
10051:                 B         : TVector;
10052:                 FirstPar, LastPar   : Int;
10053:                 V         : TMatrix); external 'dmath';
10054: { Weighted nonlinear regression }
10055: Proc SetMCFile(FileName :Str); external 'dmath';
10056: { Set file for saving MCMC simulations }
10057: Proc SimFit(RegFunc   : TRegFunc;
10058:                 X, Y      : TVector;
10059:                 Lb, Ub    : Int;
10060:                 B         : TVector;
10061:                 FirstPar, LastPar   : Int;
10062:                 V         : TMatrix); external 'dmath';
10063: { Simulation of unweighted nonlinear regression by MCMC }
10064: Proc WSimFit(RegFunc   : TRegFunc;
10065:                  X, Y, S   : TVector;
10066:                  Lb, Ub    : Int;
10067:                  B         : TVector;
10068:                  FirstPar, LastPar   : Int;
10069:                  V         : TMatrix); external 'dmath';
10070: { Simulation of weighted nonlinear regression by MCMC }
10071: { ----------------------------------------------------------------
10072:   Nonlinear regression models
10073:   ---------------------------------------------------------------- }
10074: Proc FracFit(X, Y       : TVector;
10075:                  Lb, Ub     : Int;
10076:                  Deg1, Deg2 : Int;
10077:                  ConsTerm   :Bool;
10078:                  MaxIter    : Int;
10079:                  Tol        : Float;
10080:                  B          : TVector;
10081:                  V          : TMatrix); external 'dmath';
10082: { Unweighted fit of rational fraction }
10083: Proc WFracFit(X, Y, S    : TVector;
10084:                   Lb, Ub     : Int;
10085:                   Deg1, Deg2 : Int;
10086:                   ConsTerm   :Bool;
10087:                   MaxIter    : Int;
10088:                   Tol        : Float;
10089:                   B          : TVector;
10090:                   V          : TMatrix); external 'dmath';
10091: { Weighted fit of rational fraction }
10092:
10093: Func FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10094: { Returns the value of the rational fraction at point X }
10095: Proc ExpFit(X, Y         : TVector;
10096:                 Lb, Ub, Nexp : Int;
10097:                 ConsTerm     :Bool;
10098:                 MaxIter      : Int;
10099:                 Tol          : Float;
10100:                 B            : TVector;
10101:                 V            : TMatrix); external 'dmath';
10102: { Unweighted fit of sum of exponentials }
10103: Proc WExpFit(X, Y, S      : TVector;
10104:                  Lb, Ub, Nexp : Int;
10105:                  ConsTerm     :Bool;
10106:                  MaxIter      : Int;
10107:                  Tol          : Float;
10108:                  B            : TVector;
10109:                  V            : TMatrix); external 'dmath';
10110: { Weighted fit of sum of exponentials }
10111: Func ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10112: { Returns the value of the regression Func at point X }
10113: Proc IncExpFit(X, Y     : TVector;
10114:                    Lb, Ub   : Int;
10115:                    ConsTerm :Bool;
10116:                    MaxIter  : Int;
10117:                    Tol      : Float;
10118:                    B        : TVector;
10119:                    V        : TMatrix); external 'dmath';
10120: { Unweighted fit of model of increasing exponential }
10121: Proc WIncExpFit(X, Y, S  : TVector;
10122:                     Lb, Ub   : Int;
10123:                     ConsTerm :Bool;
10124:                     MaxIter  : Int;
10125:                     Tol      : Float;
10126:                     B        : TVector;
10127:                     V        : TMatrix); external 'dmath';
10128: { Weighted fit of increasing exponential }
10129: Func IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10130: { Returns the value of the regression Func at point X }
10131: Proc ExpLinFit(X, Y    : TVector;
10132:                    Lb, Ub  : Int;
```

```
10133:                     MaxIter : Int;
10134:                     Tol     : Float;
10135:                     B       : TVector;
10136:                     V       : TMatrix); external 'dmath';
10137: { Unweighted fit of the "exponential + linear" model }
10138: Proc WExpLinFit(X, Y, S : TVector;
10139:                      Lb, Ub  : Int;
10140:                      MaxIter : Int;
10141:                      Tol     : Float;
10142:                      B       : TVector;
10143:                      V       : TMatrix); external 'dmath';
10144: { Weighted fit of the "exponential + linear" model }
10145: Func ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10146: { Returns the value of the regression Func at point X }
10147: Proc MichFit(X, Y    : TVector;
10148:                  Lb, Ub  : Int;
10149:                  MaxIter : Int;
10150:                  Tol     : Float;
10151:                  B       : TVector;
10152:                  V       : TMatrix); external 'dmath';
10153: { Unweighted fit of Michaelis equation }
10154: Proc WMichFit(X, Y, S : TVector;
10155:                   Lb, Ub  : Int;
10156:                   MaxIter : Int;
10157:                   Tol     : Float;
10158:                   B       : TVector;
10159:                   V       : TMatrix); external 'dmath';
10160: { Weighted fit of Michaelis equation }
10161: Func MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10162: { Returns the value of the Michaelis equation at point X }
10163: Proc MintFit(X, Y    : TVector;
10164:                  Lb, Ub  : Int;
10165:                  MintVar : TMintVar;
10166:                  Fit_S0  :Bool;
10167:                  MaxIter : Int;
10168:                  Tol     : Float;
10169:                  B       : TVector;
10170:                  V       : TMatrix); external 'dmath';
10171: { Unweighted fit of the integrated Michaelis equation }
10172: Proc WMintFit(X, Y, S : TVector;
10173:                   Lb, Ub  : Int;
10174:                   MintVar : TMintVar;
10175:                   Fit_S0  :Bool;
10176:                   MaxIter : Int;
10177:                   Tol     : Float;
10178:                   B       : TVector;
10179:                   V       : TMatrix); external 'dmath';
10180: { Weighted fit of the integrated Michaelis equation }
10181: Func MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10182: { Returns the value of the integrated Michaelis equation at point X }
10183: Proc HillFit(X, Y    : TVector;
10184:                  Lb, Ub  : Int;
10185:                  MaxIter : Int;
10186:                  Tol     : Float;
10187:                  B       : TVector;
10188:                  V       : TMatrix); external 'dmath';
10189: { Unweighted fit of Hill equation }
10190: Proc WHillFit(X, Y, S : TVector;
10191:                   Lb, Ub  : Int;
10192:                   MaxIter : Int;
10193:                   Tol     : Float;
10194:                   B       : TVector;
10195:                   V       : TMatrix); external 'dmath';
10196: { Weighted fit of Hill equation }
10197: Func HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10198: { Returns the value of the Hill equation at point X }
10199: Proc LogiFit(X, Y     : TVector;
10200:                  Lb, Ub   : Int;
10201:                  ConsTerm :Bool;
10202:                  General  :Bool;
10203:                  MaxIter  : Int;
10204:                  Tol      : Float;
10205:                  B        : TVector;
10206:                  V        : TMatrix); external 'dmath';
10207: { Unweighted fit of logistic Func }
10208: Proc WLogiFit(X, Y, S  : TVector;
10209:                   Lb, Ub   : Int;
10210:                   ConsTerm :Bool;
10211:                   General  :Bool;
10212:                   MaxIter  : Int;
10213:                   Tol      : Float;
10214:                   B        : TVector;
10215:                   V        : TMatrix); external 'dmath';
10216: { Weighted fit of logistic Func }
10217: Func LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10218: { Returns the value of the logistic Func at point X }
10219: Proc PKFit(X, Y    : TVector;
10220:                Lb, Ub  : Int;
10221:                MaxIter : Int;
```

```
10222:                 Tol      : Float;
10223:                 B        : TVector;
10224:                 V        : TMatrix); external 'dmath';
10225: { Unweighted fit of the acid-base titration curve }
10226: Proc WPKFit(X, Y, S : TVector;
10227:                  Lb, Ub   : Int; MaxIter : Int;
10228:                  Tol      : Float;
10229:                  B        : TVector;
10230:                  V        : TMatrix); external 'dmath';
10231: { Weighted fit of the acid-base titration curve }
10232: Func PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10233: { Returns the value of the acid-base titration Func at point X }
10234: Proc PowFit(X, Y    : TVector;
10235:                  Lb, Ub   : Int;
10236:                  MaxIter  : Int;
10237:                  Tol      : Float;
10238:                  B        : TVector;
10239:                  V        : TMatrix); external 'dmath';
10240: { Unweighted fit of power Func }
10241: Proc WPowFit(X, Y, S : TVector;
10242:                   Lb, Ub   : Int;
10243:                   MaxIter  : Int;
10244:                   Tol      : Float;
10245:                   B        : TVector;
10246:                   V        : TMatrix); external 'dmath';
10247: { Weighted fit of power Func }
10248:
10249: Func PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10250: { Returns the value of the power Func at point X }
10251: Proc GammaFit(X, Y    : TVector;
10252:                    Lb, Ub   : Int;
10253:                    MaxIter  : Int;
10254:                    Tol      : Float;
10255:                    B        : TVector;
10256:                    V        : TMatrix); external 'dmath';
10257: { Unweighted fit of gamma distribution Func }
10258: Proc WGammaFit(X, Y, S : TVector;
10259:                     Lb, Ub   : Int;
10260:                     MaxIter  : Int;
10261:                     Tol      : Float;
10262:                     B        : TVector;
10263:                     V        : TMatrix); external 'dmath';
10264: { Weighted fit of gamma distribution Func }
10265: Func GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10266: { Returns the value of the gamma distribution Func at point X }
10267: { ----------------------------------------------------------------
10268:   Principal component analysis
10269:   ---------------------------------------------------------------- }
10270: Proc VecMean(X            : TMatrix;
10271:                   Lb, Ub, Nvar : Int;
10272:                   M            : TVector); external 'dmath';
10273: { Computes the mean vector M from matrix X }
10274: Proc VecSD(X            : TMatrix;
10275:                 Lb, Ub, Nvar : Int;
10276:                 M, S         : TVector); external 'dmath';
10277: { Computes the vector of standard deviations S from matrix X }
10278: Proc MatVarCov(X            : TMatrix;
10279:                     Lb, Ub, Nvar : Int;
10280:                     M            : TVector;
10281:                     V            : TMatrix); external 'dmath';
10282: { Computes the variance-covariance matrix V from matrix X }
10283: Proc MatCorrel(V    : TMatrix;
10284:                     Nvar : Int;
10285:                     R    : TMatrix); external 'dmath';
10286: { Computes the correlation matrix R from the var-cov matrix V }
10287: Proc PCA(R      : TMatrix;
10288:               Nvar   : Int;
10289:               Lambda : TVector;
10290:               C, Rc  : TMatrix); external 'dmath';
10291: { Performs a principal component analysis of the correlation matrix R }
10292: Proc ScaleVar(X            : TMatrix;
10293:                    Lb, Ub, Nvar : Int;
10294:                    M, S         : TVector;
10295:                    Z            : TMatrix); external 'dmath';
10296: { Scales a set of variables by subtracting means and dividing by SD's }
10297: Proc PrinFac(Z            : TMatrix;
10298:                   Lb, Ub, Nvar : Int;
10299:                   C, F         : TMatrix); external 'dmath';
10300: { Computes principal factors }
10301: { ----------------------------------------------------------------
10302:   Strings
10303:   ---------------------------------------------------------------- }
10304: Func LTrim(S :Str) :Str; external 'dmath';
10305: { Removes leading blanks }
10306: Func RTrim(S :Str) :Str; external 'dmath';
10307: { Removes trailing blanks }
10308: Func Trim(S :Str) :Str; external 'dmath';
10309: { Removes leading and trailing blanks }
10310: Func StrChar(N : Byte; C : Char) :Str; external 'dmath';
```

```
10311: { Returns a string made of character C repeated N times }
10312: Func RFill(S :Str; L : Byte) :Str; external 'dmath';
10313: { Completes string S with trailing blanks for a total length L }
10314: Func LFill(S :Str; L : Byte) :Str; external 'dmath';
10315: { Completes string S with leading blanks for a total length L }
10316: Func CFill(S :Str; L : Byte) :Str; external 'dmath';
10317: { Centers string S on a total length L }
10318: Func Replace(S :Str; C1, C2 : Char) :Str; external 'dmath';
10319: { Replaces in string S all the occurences of C1 by C2 }
10320: Func Extract(S:str; var Index:Byte; Delim:Char):str; external 'dmath';
10321: { Extracts a field from a string }
10322: Proc Parse(S:str;Delim:Char;Field:TStrVector;var N:Byte);external 'dmath';
10323: { Parses a string into its constitutive fields }
10324: Proc SetFormat(NumLength,MaxDec:Int;FloatPoint,NSZero:Bool); external 'dmath';
10325: { Sets the numeric format }
10326: Func FloatStr(X : Float) :Str; external 'dmath';
10327: { Converts a real to a string according to the numeric format }
10328: Func IntStr(N : LongInt) :Str; external 'dmath';
10329: { Converts an Int to a string }
10330: Func CompStr(Z : Complex) :Str; external 'dmath';
10331: { Converts a complex number to a string }
10332: {$IFDEF DELPHI}
10333: Func StrDec(S :Str) :Str; external 'dmath';
10334: { Set decimal separator to the symbol defined in SysUtils }
10335: Func IsNumeric(var S :Str; var X : Float) :Bool; external 'dmath';
10336: { Test if a string represents a number and returns it in X }
10337: Func ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10338: { Reads a floating point number from an Edit control }
10339: Proc WriteNumToFile(var F : Text; X : Float); external 'dmath';
10340: { Writes a floating point number in a text file }
10341: {$ENDIF}
10342: { ----------------------------------------------------------------
10343:   BGI / Delphi graphics
10344:   ---------------------------------------------------------------- }
10345: Func InitGraphics
10346: {$IFDEF DELPHI}
10347: (Width, Height : Int) :Bool;
10348: {$ELSE}
10349: (Pilot, Mode : Int; BGIPath :Str) :Bool; {$ENDIF} external 'dmath';
10350: { Enters graphic mode }
10351: Proc SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10352:                     X1, X2, Y1,Y2 : Int; GraphBorder:Boolean); external 'dmath';
10353: { Sets the graphic window }
10354: Proc SetOxScale(Scale                : TScale;
10355:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10356: { Sets the scale on the Ox axis }
10357: Proc SetOyScale(Scale                : TScale;
10358:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10359: { Sets the scale on the Oy axis }
10360: Proc GetOxScale(var Scale                : TScale;
10361:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10362: { Returns the scale on the Ox axis }
10363: Proc GetOyScale(var Scale                : TScale;
10364:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10365: { Returns the scale on the Oy axis }
10366: Proc SetGraphTitle(Title:str);external 'dmath';{Sets title for the graph }
10367: Proc SetOxTitle(Title String);external'dmath';{ Sets title for the Ox axis }
10368: Proc SetOyTitle(Title:str);external'dmath';{Sets the title for Oy axis }
10369: Func GetGraphTitle:str;external 'dmath';{ Returns the title for the graph }
10370: Func GetOxTitle:str; external 'dmath'; { Returns the title for the Ox axis }
10371: Func GetOyTitle:str; external 'dmath'; { Returns the title for the Oy axis }
10372: {$IFNDEF DELPHI}
10373: Proc SetTitleFont(FontIndex, Width, Height : Int); external 'dmath';
10374: { Sets the font for the main graph title }
10375: Proc SetOxFont(FontIndex, Width, Height : Int); external 'dmath';
10376: { Sets the font for the Ox axis (title and labels) }
10377: Proc SetOyFont(FontIndex, Width, Height : Int); external 'dmath';
10378: { Sets the font for the Oy axis (title and labels) }
10379: Proc SetLgdFont(FontIndex, Width, Height : Int); external 'dmath';
10380: { Sets the font for the legends }
10381: Proc SetClipping(Clip :Bool); external 'dmath';
10382: { Determines whether drawings are clipped at the current viewport
10383:   boundaries, according to the value of the Boolean parameter Clip }
10384: {$ENDIF}
10385: Proc PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10386: { Plots the horizontal axis }
10387: Proc PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10388: { Plots the vertical axis }
10389: Proc PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF}Grid:TGrid);external 'dmath';
10390: { Plots a grid on the graph }
10391: Proc WriteGraphTitle{$IFDEF DELPHI}(Canvas:TCanvas){$ENDIF}; external 'dmath';
10392: { Writes the title of the graph }
10393: Proc SetMaxCurv(NCurv : Byte); external 'dmath';
10394: { Sets the maximum number of curves and re-initializes their parameters }
10395: Proc SetPointParam
10396: {$IFDEF DELPHI}
10397: (CurvIndex, Symbol, Size : Int; Color : TColor);
10398: {$ELSE}
10399: (CurvIndex, Symbol, Size, Color : Int); {$ENDIF} external 'dmath';
```

```
10400: { Sets the point parameters for curve # CurvIndex }
10401: Proc SetLineParam
10402: {$IFDEF DELPHI}
10403: (CurvIndex : Int; Style : TPenStyle; Width : Int; Color : TColor);
10404: {$ELSE}
10405: (CurvIndex, Style, Width, Color : Int); {$ENDIF} external 'dmath';
10406: { Sets the line parameters for curve # CurvIndex }
10407: Proc SetCurvLegend(CurvIndex : Int; Legend :Str); external 'dmath';
10408: { Sets the legend for curve # CurvIndex }
10409: Proc SetCurvStep(CurvIndex, Step : Int); external 'dmath';
10410: { Sets the step for curve # CurvIndex }
10411: Func GetMaxCurv: Byte;external 'dmath';{ Returns the maximum number of curves }
10412: Proc GetPointParam
10413: {$IFDEF DELPHI}
10414: (CurvIndex : Int; var Symbol, Size : Int; var Color : TColor);
10415: {$ELSE}
10416: (CurvIndex : Int; var Symbol, Size, Color : Int); {$ENDIF} external 'dmath';
10417: { Returns the point parameters for curve # CurvIndex }
10418: Proc GetLineParam
10419: {$IFDEF DELPHI}
10420: (CurvIndex : Int; var Style : TPenStyle; var Width : Int; var Color : TColor);
10421: {$ELSE}
10422: (CurvIndex : Int; var Style, Width, Color : Int); {$ENDIF} external 'dmath';
10423: { Returns the line parameters for curve # CurvIndex }
10424: Func GetCurvLegend(CurvIndex : Int) :Str; external 'dmath';
10425: { Returns the legend for curve # CurvIndex }
10426: Func GetCurvStep(CurvIndex : Int) : Int; external 'dmath';
10427: { Returns the step for curve # CurvIndex }
10428: {$IFDEF DELPHI}
10429: Proc PlotPoint(Canvas    : TCanvas;
10430:                    X, Y      : Float; CurvIndex : Int); external 'dmath';
10431: {$ELSE}
10432: Proc PlotPoint(Xp, Yp, CurvIndex : Int); external 'dmath';
10433: {$ENDIF}
10434: { Plots a point on the screen }
10435: Proc PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10436:                    X, Y                 : TVector;
10437:                   Lb, Ub, CurvIndex     : Int); external 'dmath';
10438: { Plots a curve }
10439: Proc PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10440:                                X, Y, S               : TVector;
10441:                                Ns, Lb, Ub, CurvIndex : Int); external 'dmath';
10442: { Plots a curve with error bars }
10443: Proc PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10444:                   Func                  : TFunc;
10445:                   Xmin, Xmax            : Float;
10446:                   {$IFDEF DELPHI}Npt    : Int;{$ENDIF}
10447:                   CurvIndex             : Int); external 'dmath';
10448: { Plots a Func }
10449: Proc WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10450:                      NCurv                 : Int;
10451:                      ShowPoints, ShowLines :Bool); external 'dmath';
10452: { Writes the legends for the plotted curves }
10453: Proc ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10454:                 Nx, Ny, Nc            : Int;
10455:                 X, Y, Z               : TVector;
10456:                 F                     : TMatrix); external 'dmath';
10457: { Contour plot }
10458: Func Xpixel(X:Float):Int; external 'dmath';{Converts user abscissa X to screen coordinate }
10459: Func Ypixel(Y:Float):Int; external 'dmath';{Converts user ordinate Y to screen coordinate }
10460: Func Xuser(X:Int):Float; external 'dmath';{Converts screen coordinate X to user abscissa }
10461: Func Yuser(Y:Int):Float; external 'dmath';{Converts screen coordinate Y to user ordinate }
10462: {$IFNDEF DELPHI}
10463: Proc LeaveGraphics; external 'dmath';
10464: { Quits graphic mode }
10465: {$ENDIF}
10466: { ----------------------------------------------------------------
10467:   LaTeX graphics
10468:   ---------------------------------------------------------------- }
10469: Func TeX_InitGraphics(FileName          :Str; PgWidth, PgHeight : Int;
10470:                         Header            :Bool) :Bool; external 'dmath';
10471: { Initializes the LaTeX file }
10472: Proc TeX_SetWindow(X1,X2,Y1 Y2:Int; GraphBorder :Bool); external 'dmath';
10473: { Sets the graphic window }
10474: Proc TeX_LeaveGraphics(Footer:Boolean);external 'dmath';{ Close LaTeX file }
10475: Proc TeX_SetOxScale(Scale:TScale;OxMin,OxMax,OxStep:Float);external 'dmath';
10476: { Sets the scale on the Ox axis }
10477: Proc TeX_SetOyScale(Scale:TScale;OyMin,OyMax,OyStep:Float); external 'dmath';
10478: { Sets the scale on the Oy axis }
10479: Proc TeX_SetGraphTitle(Title:str);external 'dmath';{ Sets title for graph }
10480: Proc TeX_SetOxTitle(Title:str);external 'dmath';{ Sets title for Ox axis }
10481: Proc TeX_SetOyTitle(Title:str);external 'dmath';{ Sets title for Oy axis }
10482: Proc TeX_PlotOxAxis; external 'dmath'; { Plots  horizontal axis }
10483: Proc TeX_PlotOyAxis; external 'dmath'; { Plots  vertical axis }
10484: Proc TeX_PlotGrid(Grid:TGrid);external 'dmath'; { Plots a grid on the graph }
10485: Proc TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10486: Proc TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10487: { Sets the maximum number of curves and re-initializes their parameters }
10488: Proc TeX_SetPointParam(CurvIndex, Symbol, Size : Int); external 'dmath';
```

```
10489: { Sets the point parameters for curve # CurvIndex }
10490: Proc TeX_SetLineParam(CurvIndex, Style : Int;
10491:                 Width : Float; Smooth :Bool); external 'dmath';
10492: { Sets the line parameters for curve # CurvIndex }
10493: Proc TeX_SetCurvLegend(CurvIndex : Int; Legend :Str); external 'dmath';
10494: { Sets the legend for curve # CurvIndex }
10495: Proc TeX_SetCurvStep(CurvIndex, Step : Int); external 'dmath';
10496: { Sets the step for curve # CurvIndex }
10497: Proc TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Int); external 'dmath';
10498: { Plots a curve }
10499: Proc TeX_PlotCurveWithErrorBars(X,Y,S:TVector;Ns,Lb,Ub,CurvIndex Int);external 'dmath';
10500: { Plots a curve with error bars }
10501: Proc TeX_PlotFunc(Func:TFunc;X1 X2:Float;Npt:Int;CurvIndex:Int);external 'dmath';
10502: { Plots a Func }
10503: Proc TeX_WriteLegend(NCurv:Int;ShowPoints,ShowLines:Bool); external 'dmath';
10504: { Writes the legends for the plotted curves }
10505: Proc TeX_ConRec(Nx,Ny Nc:Int; X,Y,Z :TVector; F : TMatrix); external 'dmath';
10506: { Contour plot }
10507: Func Xcm(X:Float):Float; external 'dmath'; { Converts user coordinate X to cm }
10508: Func Ycm(Y:Float):Float; external 'dmath'; { Converts user coordinate Y to cm }
10509:
10510: //*******************************************************unit uPSI_SynPdf;
10511:  Func RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10512:  Func _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10513:  Func _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10514:  Func PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10515:  Func PdfRect1( const Box : TPdfBox) : TPdfRect;
10516:  Func PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10517:  //Func _GetCharCount( Text : PAnsiChar) : Int
10518:  //Proc L2R( W : PWideChar; L : Int)
10519:  Func PdfCoord( MM : single) : Int
10520:  Func CurrentPrinterPaperSize : TPDFPaperSize
10521:  Func CurrentPrinterRes : TPoint
10522:  Proc GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10523:  Proc GDICommentOutline(MetaHandle:HDC; const aTitle : RawUTF8; aLevel : Int)
10524:  Proc GDICommentLink(MetaHandle:HDC;const aBookmarkName:RawUTF8;const aRect: TRect)
10525:  Const('Usp10','String 'usp10.dll
10526:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10527:   'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10528:  TScriptState_set', 'set of TScriptState_enum
10529: //*********************************************************************
10530:
10531: Proc SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10532: begin
10533:  Proc PMrandomize( I : word)
10534:  Func PMrandom : longint
10535:  Func Rrand : extended
10536:  Func Irand( N : word) : word
10537:  Func Brand( P : extended) :Bool
10538:  Func Nrand : extended
10539: end;
10540:
10541: Proc SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10542: begin
10543:   Func Endian( x : LongWord) : LongWord
10544:   Func Endian64( x : Int64) : Int64
10545:   Func spRol( x : LongWord; y : Byte) : LongWord
10546:   Func spRor( x : LongWord; y : Byte) : LongWord
10547:   Func Ror64( x : Int64; y : Byte) : Int64
10548: end;
10549:
10550: Proc SIRegister_MapReader(CL: TPSPascalCompiler);
10551: begin
10552:  Proc ClearModules
10553:  Proc ReadMapFile( Fname :Str)
10554:  Func AddressInfo( Address : dword) :Str
10555: end;
10556:
10557: Proc SIRegister_LibTar(CL: TPSPascalCompiler);
10558: begin
10559:  TTarPermission','(tpReadByOwner,tpWriteByOwner,tpExecuteByOwner,tpReadByGroup,tpWriteByGroup,
       tpExecuteByGroup,tpReadByOther,tpWriteByOther, tpExecuteByOther)
10560:   TTarPermissions', 'set of TTarPermission
10561:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10562:    +'Block,ftDirectory,ftFifo,ftContiguous,ftDumpDir,ftMultiVolume,ftVolumeHeader;
10563:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10564:   TTarModes', 'set of TTarMode
10565:   TTarDirRec', 'record Name :Str; Size : INT64; DateTime : TDa'
10566:    +teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10567:    +RING; UID : Int; GID : Int; UserName :Str; GroupName :Str;'
10568:    + ChecksumOK :Bool; Mode : TTarModes; Magic :Str; MajorDevNo : INTE'
10569:    +GER; MinorDevNo : Int; FilePos : INT64; end
10570:   SIRegister_TTarArchive(CL);
10571:   SIRegister_TTarWriter(CL);
10572:  Func PermissionString( Permissions : TTarPermissions) :Str
10573:  Func ConvertFilename( Filename :Str) :Str
10574:  Func FileTimeGMT( FileName :Str) : TDateTime;
10575:  Func FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10576:  Proc ClearDirRec( var DirRec : TTarDirRec)
```

```
10577: end;
10578:
10579: //*****************************************************unit uPSI_TlHelp32;
10580: Proc SIRegister_TlHelp32(CL: TPSPascalCompiler);
10581: begin
10582:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10583:  Func CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10584:  Const('TH32CS_SNAPHEAPLIST',LongWord( $00000001);
10585:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10586:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10587:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10588:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10589:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10590:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10591:   AddTypeS('THeapList32', 'tagHEAPLIST32
10592:  Const('HF32_DEFAULT','LongInt'( 1);
10593:  Const('HF32_SHARED','LongInt'( 2);
10594:  Func Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10595:  Func Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10596:  AddTypeS(tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10597:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10598:    +dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10599:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10600:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10601:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10602:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10603:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10604:  Func Heap32First(var lphe:THeapEntry32;th32ProcessID,th32HeapID:DWORD):BOOL
10605:  Func Heap32Next( var lphe : THeapEntry32) : BOOL
10606:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10607:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10608:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10609:    +'aPri : Longint; dwFlags : DWORD; end
10610:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10611:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10612:  Func Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10613:  Func Thread32Next( hSnapshot: THandle; var lpte : TThreadENtry32):BOOL end;
10614:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10615:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10616:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10617:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10618:  Const('EWX_LOGOFF','LongInt'( 0);
10619:  Const('EWX_SHUTDOWN','LongInt'( 1);
10620:  Const('EWX_REBOOT','LongInt'( 2);
10621:  Const('EWX_FORCE','LongInt'( 4);
10622:  Const('EWX_POWEROFF','LongInt'( 8);
10623:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10624:  Func GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10625:  Func GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10626:  Func GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10627:  Func GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10628:  Func GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10629:  Func GetWindowWord( hWnd : HWND; nIndex : Int) : Word
10630:  Func SetWindowWord( hWnd : HWND; nIndex : Int; wNewWord : Word) : Word
10631:  Func GetWindowLong( hWnd : HWND; nIndex : Int) : Longint
10632:  Func SetWindowLong( hWnd : HWND; nIndex : Int; dwNewLong:Longint) : Longint
10633:  Func GetClassWord( hWnd : HWND; nIndex : Int) : Word
10634:  Func SetClassWord( hWnd : HWND; nIndex : Int; wNewWord : Word) : Word
10635:  Func GetClassLong( hWnd : HWND; nIndex : Int) : DWORD
10636:  Func SetClassLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint) : DWORD
10637:  Func GetDesktopWindow : HWND
10638:  Func GetParent( hWnd : HWND) : HWND
10639:  Func SetParent( hWndChild, hWndNewParent : HWND) : HWND
10640:  Func GetTopWindow( hWnd : HWND) : HWND
10641:  Func GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10642:  Func GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10643:  //Delphi DFM
10644:  Func LoadDFMFile2Strings(const AFile:str; AStrings:TStrings; var WasText:bool):Int
10645:  Func SaveStrings2DFMFile( AStrings : TStrings; const AFile :Str) : Int
10646:  Proc GetHighlighters(AOwner:TComponent; AHighlighters:TStringList;AppendToList:Bool);
10647:  Func GetHighlightersFilter(AHighlighters: TStringList):Str;
10648:  Func GetHighlighterFromFileExt(AHighlighters: TStringList;Extension:Str):TSynCustomHighlighter;
10649:  Func ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10650:  Func OpenIcon( hWnd : HWND) : BOOL
10651:  Func CloseWindow( hWnd : HWND) : BOOL
10652:  Func MoveWindow(hWnd:HWND;X Y,nWidth, nHeight : Int; bRepaint : BOOL) : BOOL
10653:  Func SetWindowPos(hWnd:HWND;hWndInsertAfter:HWND;X,Y,cx,cy:Int;uFlags:UINT): BOOL
10654:  Func IsWindowVisible( hWnd : HWND) : BOOL
10655:  Func IsIconic( hWnd : HWND) : BOOL
10656:  Func AnyPopup : BOOL
10657:  Func BringWindowToTop( hWnd : HWND) : BOOL
10658:  Func IsZoomed( hWnd : HWND) : BOOL
10659:  Func IsWindow( hWnd : HWND) : BOOL
10660:  Func IsMenu( hMenu : HMENU) : BOOL
10661:  Func IsChild( hWndParent, hWnd : HWND) : BOOL
10662:  Func DestroyWindow( hWnd : HWND) : BOOL
10663:  Func ShowWindow( hWnd : HWND; nCmdShow : Int) : BOOL
10664:  Func AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10665:  Func ShowWindowAsync( hWnd : HWND; nCmdShow : Int) : BOOL
```

```
10666:  Func FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10667:  Func IsWindowUnicode( hWnd : HWND) : BOOL
10668:  Func EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10669:  Func IsWindowEnabled( hWnd : HWND) : BOOL
10670:
10671: Proc SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10672: begin
10673:  const('ShowSetupDialogOptLong','String '--setup
10674:  PrimaryConfPathOptLong','String '--primary-config-path=
10675:  PrimaryConfPathOptShort','String '--pcp=
10676:  SecondaryConfPathOptLong','String '--secondary-config-path=
10677:  SecondaryConfPathOptShort','String '--scp=
10678:  NoSplashScreenOptLong','String '--no-splash-screen
10679:  NoSplashScreenOptShort','String '--nsc
10680:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10681:  SkipLastProjectOpt','String '--skip-last-project
10682:  DebugLogOpt','String '--debug-log=
10683:  DebugLogOptEnable','String '--debug-enable=
10684:  LanguageOpt','String '--language=
10685:  LazarusDirOpt','String '--lazarusdir=
10686:  Proc ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:bool);
10687:  Func GetCommandLineParameters(aCmdLineParams:TStrings;isStartLazarus:Bool):Str
10688:  Func ExtractPrimaryConfigPath( aCmdLineParams : TStrings) :Str
10689:  Func IsHelpRequested :Bool
10690:  Func IsVersionRequested :Bool
10691:  Func GetLanguageSpecified :Str
10692:  Func ParamIsOption( ParamIndex : Int; const Option :Str) :Bool
10693:  Func ParamIsOptionPlusValue(ParamIndex:int;const Option:str;out AValue:str):bool;
10694:  Proc ParseNoGuiCmdLineParams
10695:  Func ExtractCmdLineFilenames : TStrings
10696: end;
10697:
10698: Proc SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10699: begin
10700:  Func CompareFilenames( const Filename1, Filename2 :Str) : Int
10701:  Func CompareFilenamesIgnoreCase( const Filename1, Filename2 :Str) : Int
10702:  Func CompareFileExt( const Filename, Ext :Str; CaseSensitive :Bool) : Int;
10703:  Func CompareFileExt1( const Filename, Ext :Str) : Int;
10704:  Func CompareFilenameStarts( const Filename1, Filename2 :Str) : Int
10705:  Func CompareFilenames(Filename1:PChar;Len1:Int; Filename2:PChar;Len2:Int):Int
10706:  Func CompareFilenamesP(Filename1,Filename2:PChar;IgnoreCase :Bool) : Int
10707:  Func DirPathExists( DirectoryName :Str) :Bool
10708:  Func DirectoryIsWritable( const DirectoryName :Str) :Bool
10709:  Func ExtractFileNameOnly( const AFilename :Str) :Str
10710:  Func FilenameIsAbsolute( const TheFilename :Str) :Bool
10711:  Func FilenameIsWinAbsolute( const TheFilename :Str) :Bool
10712:  Func FilenameIsUnixAbsolute( const TheFilename :Str) :Bool
10713:  Func ForceDirectory( DirectoryName :Str) :Bool
10714:  Proc CheckIfFileIsExecutable( const AFilename :Str)
10715:  Proc CheckIfFileIsSymlink( const AFilename :Str)
10716:  Func FileIsText( const AFilename :Str) :Bool
10717:  Func FileIsText2(const AFilename:str; out FileReadable: bool): bool
10718:  Func FilenameIsTrimmed( const TheFilename :Str) :Bool
10719:  Func FilenameIsTrimmed2( StartPos : PChar; NameLen : Int) :Bool
10720:  Func TrimFilename( const AFilename :Str) :Str
10721:  Func ResolveDots( const AFilename :Str) :Str
10722:  Proc ForcePathDelims( var FileName :Str)
10723:  Func GetForcedPathDelims( const FileName :Str) :Str
10724:  Func CleanAndExpandFilename( const Filename :Str) :Str
10725:  Func CleanAndExpandDirectory( const Filename :Str) :Str
10726:  Func TrimAndExpandFilename(const Filename:str;const BaseDir:str):str
10727:  Func TrimAndExpandDirectory(const Filename:str;const BaseDir:str):Str
10728:  Func TryCreateRelativePath(const Dest,Source:str; UsePointDirectory:bool;
        AlwaysRequireSharedBaseFolder:Bool; out RelPath :Str) :Bool
10729:  Func CreateRelativePath(const Filename,
        BaseDirectory:str;UsePointDirectory:boolean;AlwaysRequireSharedBaseFolder:Bool) :Str
10730:  Func FileIsInPath( const Filename, Path :Str) :Bool
10731:  Func AppendPathDelim( const Path :Str) :Str
10732:  Func ChompPathDelim( const Path :Str) :Str
10733:  Func CreateAbsoluteSearchPath(const SearchPath,BaseDirectory:str):str
10734:  Func CreateRelativeSearchPath(const SearchPath,BaseDirectory:str):Str
10735:  Func MinimizeSearchPath( const SearchPath :Str) :Str
10736:  Func FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10737:  (*Func FileExistsUTF8( const Filename :Str) :Bool
10738:  Func FileAgeUTF8( const FileName :Str) : Longint
10739:  Func DirectoryExistsUTF8( const Directory :Str) :Bool
10740:  Func ExpandFileNameUTF8( const FileName :Str; BaseDir :Str) :Str
10741:  Func FindFirstUTF8(const Path:str;Attr:Longint;out Rslt:TSearchRec): Longint
10742:  Func FindNextUTF8( var Rslt : TSearchRec) : Longint
10743:  Proc FindCloseUTF8( var F : TSearchrec)
10744:  Func FileSetDateUTF8( const FileName :Str; Age : Longint) : Longint
10745:  Func FileGetAttrUTF8( const FileName :Str) : Longint
10746:  Func FileSetAttrUTF8( const Filename :Str; Attr : longint) : Longint
10747:  Func DeleteFileUTF8( const FileName :Str) :Bool
10748:  Func RenameFileUTF8( const OldName, NewName :Str) :Bool
10749:  Func FileSearchUTF8(const Name,DirList:str;ImplicitCurrentDir:Bool):str
10750:  Func FileIsReadOnlyUTF8( const FileName :Str) :Bool
10751:  Func GetCurrentDirUTF8 :Str
10752:  Func SetCurrentDirUTF8( const NewDir :Str) :Bool
```

```
10753:  Func CreateDirUTF8( const NewDir :Str) :Bool
10754:  Func RemoveDirUTF8( const Dir :Str) :Bool
10755:  Func ForceDirectoriesUTF8( const Dir :Str) :Bool
10756:  Func FileOpenUTF8( const FileName :Str; Mode : Int) : THandle
10757:  Func FileCreateUTF8( const FileName :Str) : THandle;
10758:  Func FileCreateUTF81( const FileName :Str; Rights :Card): THandle;
10759:  Func FileCreateUtf82(const FileName:str;ShareMode:Int;Rights:Card): THandle;
10760:  Func FileSizeUtf8( const Filename :Str) : int64
10761:  Func GetFileDescription( const AFilename :Str) :Str
10762:  Func GetAppConfigDirUTF8( Global :Bool; Create :Bool) :Str
10763:  Func GetAppConfigFileUTF8(Global:Bool;SubDir:bool;CreateDir: bool):Str
10764:  Func GetTempFileNameUTF8( const Dir, Prefix :Str) :Str*)
10765:  Func IsUNCPath( const Path :Str) :Bool
10766:  Func ExtractUNCVolume( const Path :Str) :Str
10767:  Func ExtractFileRoot( FileName :Str) :Str
10768:  Func GetDarwinSystemFilename( Filename :Str) :Str
10769:  Proc SplitCmdLineParams( const Params:str; ParamList:TStrings; ReadBackslash:bool)
10770:  Func StrToCmdLineParam( const Param :Str) :Str
10771:  Func MergeCmdLineParams( ParamList : TStrings) :Str
10772:  Proc InvalidateFileStateCache( const Filename :Str)
10773:  Func FindAllFiles(const SearchPath:str;SearchMask:Str;SearchSubDirs:Bool):TStringList);
10774:  Func FindAllDirectories(const SearchPath:str;SearchSubDirs:Bool):TStringList
10775:  Func FindAllDocs(const Root, extmask:Str): TStringlist;
10776:  Func ReadFileToString( const Filename:Str):Str
10777:  Proc Incl(var X: longint; N: Longint);
10778:  type TCopyFileFlag = ( cffOverwriteFile,
10779:     cffCreateDestDirectory, cffPreserveTime );
10780:   TCopyFileFlags = set of TCopyFileFlag;*)
10781:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10782:   TCopyFileFlags', 'set of TCopyFileFlag
10783:   Func CopyDirTree(const SourceDir,TargetDir:str;Flags:TCopyFileFlags):Bool
10784: end;
10785:
10786: Proc SIRegister_lazMasks(CL: TPSPascalCompiler);
10787: begin
10788:   TMaskCharType','( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10789:   SIRegister_TMask(CL);
10790:   SIRegister_TParseStringList(CL);
10791:   SIRegister_TMaskList(CL);
10792:  Func MatchesMask(const FileName,Mask:str;const CaseSensitive:Bool):Bool
10793:  Func MatchesWindowsMask(const FileName,Mask:str;const CaseSensitive:Bool): Bool;
10794:  Func MatchesMaskList(const FileName,Mask:Str;Separatr:Char;const CaseSensitive:Bool):Bool;
10795:  Func MatchesWindowsMaskList(const FileName,Mask:str;Separat:Char;const CaseSensitive:Bool):Bool;
10796: end;
10797:
10798: Proc SIRegister_JvShellHook(CL: TPSPascalCompiler);
10799: begin
10800: //PShellHookInfo', '^TShellHookInfo // will not work
10801:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10802:   SHELLHOOKINFO', 'TShellHookInfo
10803:   LPSHELLHOOKINFO', 'PShellHookInfo
10804:   TJvShellHookEvent', 'Proc ( Sender : TObject; var Msg : TMessage)
10805:   SIRegister_TJvShellHook(CL);
10806:  Func InitJvShellHooks :Bool
10807:  Proc UnInitJvShellHooks
10808: end;
10809:
10810: Proc SIRegister_JvExControls(CL: TPSPascalCompiler);
10811: begin
10812:  TDlgCode','(dcWantAllKeys,dcWantArrows,dcWantChars,dcButton,dcHasSetSel,dcWantTab,dcNative)
10813:   TDlgCodes', 'set of TDlgCodedcWantMessage',' dcWantAllKeys);
10814:   SIRegister_IJvExControl(CL);
10815:   SIRegister_IJvDenySubClassing(CL);
10816:   SIRegister_TStructPtrMessage(CL);
10817:  Proc SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10818:  Proc DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl :Bool);
10819:  Proc DrawDotNetControl1(DC:HDC; R:TRect;AColor:TColor;UseFocusedColor:Bool);
10820:  Proc HandleDotNetHighlighting(Cntrl:TWinControl;const Msg:TMessage;MouseOver:Bool;Color:TColor);
10821:  Func CreateWMMessage( Msg : Int; WParam : Int; LParam : Longint) : TMessage;
10822:  Func CreateWMMessage1( Msg : Int; WParam : Int; LParam : TControl) : TMessage;
10823:  Func SmallPointToLong( const Pt : TSmallPoint) : Longint
10824:  Func ShiftStateToKeyData( Shift : TShiftState) : Longint
10825:  Func GetFocusedControl( AControl : TControl) : TWinControl
10826:  Func DlgcToDlgCodes( Value : Longint) : TDlgCodes
10827:  Func DlgCodesToDlgc( Value : TDlgCodes) : Longint
10828:  Proc GetHintColor(var HintInfo:THintInfo AControl:TControl;HintColor:TColor)
10829:  Func DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) :Bool
10830:   SIRegister_TJvExControl(CL);
10831:   SIRegister_TJvExWinControl(CL);
10832:   SIRegister_TJvExCustomControl(CL);
10833:   SIRegister_TJvExGraphicControl(CL);
10834:   SIRegister_TJvExHintWindow(CL);
10835:   SIRegister_TJvExPubGraphicControl(CL);
10836: end;
10837:
10838: (*----------------------------------------------------------------------------*)
10839: Proc SIRegister_EncdDecd(CL: TPSPascalCompiler);
10840: begin
10841:  Proc EncodeStream(Input, Output : TStream)
```

```
10842:  Proc DecodeStream(Input, Output : TStream)
10843:  Func EncodeString1(const Input :Str) :Str
10844:  Func DecodeString1(const Input :Str) :Str
10845: end;
10846:
10847: (*----------------------------------------------------------------------------*)
10848: Proc SIRegister_SockAppReg(CL: TPSPascalCompiler);
10849: begin
10850:   SIRegister_TWebAppRegInfo(CL);
10851:   SIRegister_TWebAppRegList(CL);
10852:  Proc GetRegisteredWebApps( AList : TWebAppRegList)
10853:  Proc RegisterWebApp( const AFileName, AProgID :Str)
10854:  Proc UnregisterWebApp( const AProgID :Str)
10855:  Func FindRegisteredWebApp( const AProgID :Str) :Str
10856:  Func CreateRegistry(InitializeNewFile:Bool):TCustomIniFilesUDPPort',String 'UDPPort
10857: end;
10858:
10859: Proc SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10860: begin
10861:  // TStringDynArray', 'array of string
10862:  Func GetEnvVarValue( const VarName :Str) :Str
10863:  Func SetEnvVarValue( const VarName, VarValue :Str) : Int
10864:  Func DeleteEnvVar( const VarName :Str) : Int
10865:  Func CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:str;const
        BufSize:Int):Int;
10866:  Func ExpandEnvVars( const Str :Str) :Str
10867:  Func GetAllEnvVars( const Vars : TStrings) : Int
10868:  Proc GetAllEnvVarNames( const Names : TStrings);
10869:  Func GetAllEnvVarNames1 : TStringDynArray;
10870:  Func EnvBlockSize : Int
10871:   TPJEnvVarsEnum', 'Proc ( const VarName :Str; Data : TObject)
10872:   SIRegister_TPJEnvVarsEnumerator(CL);
10873:   SIRegister_TPJEnvVars(CL);
10874:   FindClass('TOBJECT'),'EPJEnvVars
10875:   FindClass('TOBJECT'),'EPJEnvVars
10876:  //Proc Register
10877: end;
10878:
10879: (*----------------------------------------------------------------------------*)
10880: Proc SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10881: begin
10882:  'cOneSecInMS','LongInt'( 1000);
10883:  //'cDefTimeSlice','LongInt'( 50);
10884:  //'cDefMaxExecTime',' cOneMinInMS);
10885:  'cAppErrorMask','LongInt'( 1 shl 29);
10886:  Func IsApplicationError( const ErrCode : LongWord) :Bool
10887:   TPJConsoleAppPriority','( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10888:   TPJConsoleColors','record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10889:  Func MakeConsoleColors(const AForeground,ABackground:TPJConsoleColor):TPJConsoleColors;
10890:  Func MakeConsoleColors1(const AForeground,ABackground:TColor): TPJConsoleColors;
10891:  Func MakeConsoleColors2(const AForeground ABackground:TAlphaColor): TPJConsoleColors;
10892:  Func MakeSize( const ACX, ACY : LongInt) : TSize
10893:   SIRegister_TPJCustomConsoleApp(CL);
10894:   SIRegister_TPJConsoleApp(CL);
10895: end;
10896:
10897: Proc SIRegister_ip_misc(CL: TPSPascalCompiler);
10898: begin
10899:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10900:   t_encoding', '( uuencode, base64, mime )
10901:  Func internet_date( date : TDateTime) :Str
10902:  Func lookup_hostname( const hostname :Str) : longint
10903:  Func my_hostname :Str
10904:  Func my_ip_address : longint
10905:  Func ip2string( ip_address : longint) :Str
10906:  Func resolve_hostname( ip : longint) :Str
10907:  Func address_from( const s :Str; count : Int) :Str
10908:  Func encode_base64( data : TStream) : TStringList
10909:  Func decode_base64( source : TStringList) : TMemoryStream
10910:  Func posn( const s, t :Str; count : Int) : Int
10911:  Func poscn( c : char; const s :Str; n : Int) : Int
10912:  Func filename_of( const s :Str) :Str
10913:  //Func trim( const s :Str) :Str
10914:  //Proc setlength( var s :Str; l : byte)
10915:  Func TimeZoneBias : longint
10916:  Func eight2seven_quoteprint( const s :Str) :Str
10917:  Func eight2seven_german( const s :Str) :Str
10918:  Func seven2eight_quoteprint( const s :Str) :Str end;
10919:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10920:  Func socketerror : cint
10921:  Func fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10922:  Func fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint): ssize_t
10923:  Func fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint): ssize_t
10924:  //Func fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10925:  Func fplisten( s : cint; backlog : cint) : cint
10926:  //Func fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10927:  //Func fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10928:  //Func fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10929:  Func NetAddrToStr( Entry : in_addr) :Str
```

```
10930:  Func HostAddrToStr( Entry : in_addr) :Str
10931:  Func StrToHostAddr( IP :Str) : in_addr
10932:  Func StrToNetAddr( IP :Str) : in_addr
10933:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10934:   cint8', 'shortint
10935:   cuint8', 'byte
10936:   cchar', 'cint8
10937:   cschar', 'cint8
10938:   cuchar', 'cuint8
10939:   cint16', 'smallint
10940:   cuint16', 'word
10941:   cshort', 'cint16
10942:   csshort', 'cint16
10943:   cushort', 'cuint16
10944:   cint32', 'longint
10945:   cuint32', 'longword
10946:   cint', 'cint32
10947:   csint', 'cint32
10948:   cuint', 'cuint32
10949:   csigned', 'cint
10950:   cunsigned', 'cuint
10951:   cint64', 'int64
10952:   clonglong', 'cint64
10953:   cslonglong', 'cint64
10954:   cbool', 'longbool
10955:   cfloat', 'single
10956:   cdouble', 'double
10957:   clongdouble', 'extended
10958:
10959: Proc SIRegister_uLkJSON(CL: TPSPascalCompiler);
10960: begin
10961:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10962:   SIRegister_TlkJSONdotnetclass(CL);
10963:   SIRegister_TlkJSONbase(CL);
10964:   SIRegister_TlkJSONnumber(CL);
10965:   SIRegister_TlkJSONstring(CL);
10966:   SIRegister_TlkJSONboolean(CL);
10967:   SIRegister_TlkJSONnull(CL);
10968:   TlkJSONFuncEnum','Procedure(ElName:str;Elem:TlkJSONbase;data:TObject;var Continue:Boolean)
10969:   SIRegister_TlkJSONcustomlist(CL);
10970:   SIRegister_TlkJSONlist(CL);
10971:   SIRegister_TlkJSONobjectmethod(CL);
10972:   TlkHashItem', 'record hash :Card; index : Int; end
10973:   TlkHashFunction', 'Func ( const ws : WideString) :Card
10974:   SIRegister_TlkHashTable(CL);
10975:   SIRegister_TlkBalTree(CL);
10976:   SIRegister_TlkJSONobject(CL);
10977:   SIRegister_TlkJSON(CL);
10978:   SIRegister_TlkJSONstreamed(CL);
10979:  Func GenerateReadableText( vObj : TlkJSONbase; var vLevel : Int):Str
10980: end;
10981:
10982: Ex: js:= TlkJSON.create;
10983:     jsonObject:= js.ParseText(jsonstring) as TlkJSONobject;
10984:   try
10985:     writeln('get message: '+jsonObject.Field['amessage'].value);
10986:     writeln(vartostr(jsonobject.Field['book'].Child[3].field['ISBN'].value));
10987:
10988: Proc SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10989: begin
10990:   TZListSortCompare', 'Func (Item1, Item2 : TObject): Int
10991:   SIRegister_TZSortedList(CL);
10992:  Func zFirstDelimiter( const Delimiters, Str :Str) : Int
10993:  Func zLastDelimiter( const Delimiters, Str :Str) : Int
10994:  //Func MemLCompUnicode( P1, P2 : PWideChar; Len : Int) :Bool
10995:  //Func MemLCompAnsi( P1, P2 : PAnsiChar; Len : Int) :Bool
10996:  Func zStartsWith( const Str, SubStr : WideString) :Bool;
10997:  Func StartsWith1( const Str, SubStr : RawByteString) :Bool;
10998:  Func EndsWith( const Str, SubStr : WideString) :Bool;
10999:  Func EndsWith1( const Str, SubStr : RawByteString) :Bool;
11000:  Func SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
11001:  Func SQLStrToFloatDef1( Str :Str; Def : Extended) : Extended;
11002:  Func SQLStrToFloat( const Str : Ansistr) : Extended
11003:  //Func BufferToStr( Buffer : PWideChar; Length : LongInt) :Str;
11004:  //Func BufferToStr1( Buffer : PAnsiChar; Length : LongInt) :Str;
11005:  Func BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
11006:  Func StrToBoolEx( Str :Str) :Bool
11007:  Func BoolToStrEx( Bool :Bool) :Str
11008:  Func IsIpAddr( const Str :Str) :Bool  //IsIP()
11009:  Func zSplitString( const Str, Delimiters :Str) : TStrings
11010:  Proc PutSplitString( List : TStrings; const Str, Delimiters :Str)
11011:  Proc AppendSplitString( List : TStrings; const Str, Delimiters :Str)
11012:  Func ComposeString( List : TStrings; const Delimiter :Str) :Str
11013:  Func FloatToSQLStr( Value : Extended) :Str
11014:  Proc PutSplitStringEx( List : TStrings; const Str, Delimiter :Str)
11015:  Func SplitStringEx( const Str, Delimiter :Str) : TStrings
11016:  Proc AppendSplitStringEx( List : TStrings; const Str, Delimiter:Str)
11017:  Func zBytesToStr( const Value : TByteDynArray) : Ansistr
11018:  Func zStrToBytes( const Value : Ansistr) : TByteDynArray;
```

```
11019:  Func StrToBytes1( const Value : UTF8String) : TByteDynArray;
11020:  Func StrToBytes2( const Value : RawByteString) : TByteDynArray;
11021:  Func StrToBytes3( const Value : WideString) : TByteDynArray;
11022:  Func StrToBytes4( const Value : UnicodeString) : TByteDynArray;
11023:  Func BytesToVar( const Value : TByteDynArray) : Variant
11024:  Func VarToBytes( const Value : Variant) : TByteDynArray
11025:  Func AnsiSQLDateToDateTime( const Value :Str) : TDateTime
11026:  Func TimestampStrToDateTime( const Value :Str) : TDateTime
11027:  Func DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec:Bool):Str
11028:  Func EncodeCString( const Value :Str) :Str
11029:  Func DecodeCString( const Value :Str) :Str
11030:  Func zReplaceChar( const Source, Target : Char; const Str:Str):Str
11031:  Func MemPas( Buffer : PChar; Length : LongInt) :Str
11032:  Proc DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
        SubVersion:Int);
11033:  Func EncodeSQLVersioning(const MajorVersion:Int;const MinorVersion:Int;const SubVersion:Int):Int;
11034:  Func FormatSQLVersion( const SQLVersion : Int) :Str
11035:  Func ZStrToFloat( Value : AnsiChar) : Extended;
11036:  Func ZStrToFloat1( Value : Ansistr) : Extended;
11037:  Proc ZSetString( const Src : AnsiChar; var Dest : Ansistr);
11038:  Proc ZSetString1(const Src:AnsiChar;const Len:Card;var Dest:Ansistr);
11039:  Proc ZSetString2(const Src : AnsiChar; var Dest : UTF8String);
11040:  Proc ZSetString3(const Src:AnsiChar;const Len:Card;var Dest:UTF8String);
11041:  Proc ZSetString4(const Src:AnsiChar;const Len:Card;var Dest:WideString);
11042:  Proc ZSetString5(const Src : AnsiChar; var Dest : RawByteString);
11043:  Proc ZSetString6(const Src:AnsiChar;const Len:Card;var Dest:RawByteString);
11044: end;
11045:
11046: unit uPSI_ZEncoding;
11047:  Func StringToAnsiEx( const s :Str; const FromCP, ToCP : Word) : RawByteString
11048:  Func AnsiToStringEx(const s:RawByteString;const FromCP, ToCP : Word :Str
11049:  Func ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
11050:  Func ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
11051:  Func ZConvertAnsiToRaw( const Src : Ansistr; const RawCP : Word) : RawByteString
11052:  Func ZConvertRawToAnsi(const Src:RawByteString;const RawCP:Word):Ansistr
11053:  Func ZConvertAnsiToUTF8( const Src : Ansistr) : UTF8String
11054:  Func ZConvertUTF8ToAnsi( const Src : UTF8String) : Ansistr
11055:  Func ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
11056:  Func ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
11057:  Func ZConvertRawToString(const Src:RawByteString;const RawCP,StringCP:Word):str
11058:  Func ZConvertStringToRaw( const Src :Str; const StringCP, RawCP : Word) : RawByteString
11059:  Func ZConvertStringToRawWithAutoEncode(const Src:str;const StringCP,RawCP:Word):RawByteString;
11060:  Func ZConvertUTF8ToString(const Src:UTF8String;const StringCP: Word):Str
11061:  Func ZConvertStringToUTF8(const Src:Str;const StringCP:Word):UTF8String
11062:  Func ZConvertStringToUTF8WithAutoEncode(const Src:str;const StringCP:Word): UTF8String
11063:  Func ZConvertStringToAnsi( const Src :Str; const StringCP : Word) : Ansistr
11064:  Func ZConvertStringToAnsiWithAutoEncode(const Src:str;const StringCP:Word): Ansistr
11065:  Func ZConvertAnsiToString(const Src:Ansistr;const StringCP:Word):str
11066:  Func ZConvertUnicodeToString(const Src:WideString;const StringCP:Word):str
11067:  Func ZConvertUnicodeToString_CPUTF8(const Src:WideString; const StringCP : Word):Str
11068:  Func ZConvertStringToUnicode( const Src :Str; const StringCP : Word) : WideString
11069:  Func ZConvertString_CPUTF8ToUnicode(const Src:str;const StringCP:Word): WideString
11070:  Func ZConvertStringToUnicodeWithAutoEncode(const Src:str;const StringCP:Word):WideString
11071:  Func ZMoveAnsiToRaw( const Src : Ansistr; const RawCP : Word) : RawByteString
11072:  Func ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : Ansistr
11073:  Func ZMoveAnsiToUTF8( const Src : Ansistr) : UTF8String
11074:  Func ZMoveUTF8ToAnsi( const Src : UTF8String) : Ansistr
11075:  Func ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
11076:  Func ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
11077:  Func ZMoveStringToAnsi( const Src :Str; const StringCP : Word) : Ansistr
11078:  Func ZMoveAnsiToString( const Src : Ansistr; const StringCP : Word) :Str
11079:  Func ZMoveRawToString(const Src:RawByteString;const RawCP,StringCP:Word):Str
11080:  Func ZMoveStringToRaw( const Src :Str; const StringCP, RawCP : Word) : RawByteString
11081:  Func ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) :Str
11082:  Func ZMoveStringToUTF8( const Src :Str; const StringCP : Word) : UTF8String
11083:  Func ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
11084:  Func ZUnknownRawToUnicodeWithAutoEncode(const S:RawByteString;const CP:Word): WideString
11085:  Func ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
11086:  Func ZDefaultSystemCodePage : Word
11087:  Func ZCompatibleCodePages( const CP1, CP2 : Word) :Bool
11088:  Func MPing(const AHost:Str;const ATimes:Int; out AvgMS:Double):Boolean;
11089:
11090: Proc SIRegister_BoldComUtils(CL: TPSPascalCompiler);
11091: begin
11092:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
11093:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
11094:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
11095:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
11096:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
11097:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
11098:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
11099:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
11100:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
11101:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
11102:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
11103:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
11104:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
11105:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
11106:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
```

```
11107:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
11108:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
11109:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
11110:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
11111:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
11112:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
11113:  ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
11114:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
11115:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
11116:  ('EOAC_NONE','LongWord').SetUInt( $0);
11117:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
11118:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
11119:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
11120:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
11121:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
11122:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
11123:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
11124:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
11125:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
11126:   FindClass('TOBJECT'),'EBoldCom
11127:  Func BoldVariantIsType( V : OleVariant; TypeCode : Int) :Bool
11128:  Func BoldMemoryToVariant( const Buffer, BufSize : Int) : OleVariant
11129:  Func BoldStreamToVariant( Stream : TStream) : OleVariant
11130:  Func BoldStringsToVariant( Strings : TStrings) : OleVariant
11131:  Func BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Int) : Int
11132:  Func BoldVariantToStream( V : OleVariant; Stream : TStream) : Int
11133:  Func BoldVariantArrayOfArraysOfStringToStrings(V:OleVariant;Strings: TStrings) : Int
11134:  Func BoldVariantIsNamedValues( V : OleVariant) :Bool
11135:  Func BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11136:  Func BoldGetNamedValue( Data : OleVariant; const Name :Str) : OleVariant
11137:  Proc BoldSetNamedValue( Data : OleVariant; const Name :Str; Value : OleVariant)
11138:  Func BoldCreateGUID : TGUID
11139:  Func BoldCreateComObject( const ClsId,IId:TGUID; out Obj:variant;out Res:HResult):Bool
11140:  Func BoldCreateRemoteComObject(const HostName:str;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
11141:  Proc BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11142:  Proc BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11143: end;
11144:
11145: (*----------------------------------------------------------------------------*)
11146: Proc SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11147: begin
11148:  Func ParseISODate( s :Str) : TDateTime
11149:  Func ParseISODateTime( s :Str) : TDateTime
11150:  Func ParseISOTime( str :Str) : TDateTime
11151: end;
11152:
11153: (*----------------------------------------------------------------------------*)
11154: Proc SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11155: begin
11156:  Func BoldCreateGUIDAsString( StripBrackets :Bool) :Str
11157:  Func BoldCreateGUIDWithBracketsAsString :Str
11158: end;
11159:
11160: Proc SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11161: begin
11162:   FindClass('TOBJECT'),'TBoldFileHandler
11163:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11164:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11165:   TBoldInitializeFileContents', 'Proc ( StringList : TStringList)
11166:   SIRegister_TBoldFileHandler(CL);
11167:   SIRegister_TBoldDiskFileHandler(CL);
11168:  Proc BoldCloseAllFilehandlers
11169:  Proc BoldRemoveUnchangedFilesFromEditor
11170:  Func BoldFileHandlerList : TBoldObjectArray
11171:  Func BoldFileHandlerForFile(path,
       FileName:str;ModuleType:TBoldModuleType;ShowInEditor:Bool;OnInitializeFileContents:TBoldInitializeFileContents):
       TBoldFileHandler
11172: end;
11173:
11174: Proc SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11175: begin
11176:   PCharArr', 'array of PChar
11177:  Func BoldInternetOpen(Agent:Str;AccessType:Int;Proxy:str;ProxyByPass:Str;Flags:Int):ptr);
11178:  Func BoldInternetOpenUrl(iNet:Poin;URL:str;Headers:str;Flags,Context:Card):Point
11179:  Func BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11180:  Func BoldInternetCloseHandle( HINet : Pointer) : LongBool
11181:  Func BoldHttpQueryInfo(hRequest:Pointer;InfoLevel:Card;Buffer:Pointer;BufferLength:Card;Reserved:Card) :
       LongBool
11182:  Func BoldInternetQueryDataAvailable(hFile:Pointer;var NumberOfBytesAvailable:Card; flags:Card;
       Context:Card): LongBool
11183:  Func BoldHttpOpenRequest(hConnect:Pointer;Verb,ObjectName,Version,Referrer:str;AcceptTypes:PCharArr;Flags,
       Context:Card):Pointer
11184:  Func BoldHttpSendRequest(hRequest:Ptr;Headers:str;Optional:Ptr;OptionalLength:Cardi): LongBool
11185:  Func BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11186:  Func BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11187:  Func BoldInternetConnect(hInet: HINTERNET;ServerName:str; nServerPort:INTERNET_PORT; Username:str;
       Password :Str; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
```

```
11188:  Func BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11189: end;
11190:
11191: Proc SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11192: begin
11193:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11194:   SIRegister_TfrmBoldQueryUser(CL);
11195:  Func QueryUser( const Title, Query :Str) : TBoldQueryResult
11196: end;
11197:
11198: (*----------------------------------------------------------------------------*)
11199: Proc SIRegister_BoldQueue(CL: TPSPascalCompiler);
11200: begin
11201:  //('befIsInDisplayList',' BoldElementFlag0);
11202:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11203:  //('befFollowerSelected',' BoldElementFlag2);
11204:   FindClass('TOBJECT'),'TBoldQueue
11205:   FindClass('TOBJECT'),'TBoldQueueable
11206:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11207:   SIRegister_TBoldQueueable(CL);
11208:   SIRegister_TBoldQueue(CL);
11209:  Func BoldQueueFinalized :Bool
11210:  Func BoldInstalledQueue : TBoldQueue
11211: end;
11212:
11213: Proc SIRegister_Barcode(CL: TPSPascalCompiler);
11214: begin
11215:  const mmPerInch','Extended).setExtended( 25.4);
11216:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11217:    + bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11218:    +Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11219:    +bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11220:    +odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11221:   TBarLineType', '( white, black, black_half )
11222:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11223:   TShowTextPosition','(stpTopLeft,stpTopRight,stpTopCenter,stpBottomLeft,stpBottomRight, stpBottomCenter )
11224:   TCheckSumMethod', '( csmNone, csmModulo10 )
11225:   SIRegister_TAsBarcode(CL);
11226:  Func CheckSumModulo10( const data :Str) :Str
11227:  Func ConvertMmToPixelsX( const Value : Double) : Int
11228:  Func ConvertMmToPixelsY( const Value : Double) : Int
11229:  Func ConvertInchToPixelsX( const Value : Double) : Int
11230:  Func ConvertInchToPixelsY( const Value : Double) : Int
11231: end;
11232:
11233: Proc SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11234: begin
11235:   TVector3f = array[0..2] of single;
11236:   CL.AddTypeS('TAffineVector', 'TVector3f
11237:   CL.AddTypeS('TVertex', 'TAffineVector
11238:   THomogeneousByteVector', 'array[0..3] of Byte
11239:   THomogeneousWordVector', 'array[0..3] of Word
11240:   THomogeneousIntVector', 'array[0..3] of Int
11241:   THomogeneousFltVector', 'array[0..3] of single
11242:   THomogeneousDblVector', 'array[0..3] of double
11243:   THomogeneousExtVector', 'array[0..3] of extended
11244:   TAffineByteVector', 'array[0..2] of Byte
11245:   TAffineWordVector', 'array[0..2] of Word
11246:   TAffineIntVector', 'array[0..2] of Int
11247:   TAffineFltVector', 'array[0..2] of single
11248:   TAffineDblVector', 'array[0..2] of double
11249:   TAffineExtVector', 'array[0..2] of extended
11250:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11251:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11252:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11253:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11254:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11255:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11256:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11257:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11258:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11259:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11260:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11261:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11262:   TMatrix4b', 'THomogeneousByteMatrix
11263:   TMatrix4w', 'THomogeneousWordMatrix
11264:   TMatrix4i', 'THomogeneousIntMatrix
11265:   TMatrix4f', 'THomogeneousFltMatrix
11266:   TMatrix4d', 'THomogeneousDblMatrix
11267:   TMatrix4e', 'THomogeneousExtMatrix
11268:   TMatrix3b', 'TAffineByteMatrix
11269:   TMatrix3w', 'TAffineWordMatrix
11270:   TMatrix3i', 'TAffineIntMatrix
11271:   TMatrix3f', 'TAffineFltMatrix
11272:   TMatrix3d', 'TAffineDblMatrix
11273:   TMatrix3e', 'TAffineExtMatrix
11274:   //'PMatrix', '^TMatrix // will not work
11275:   TMatrixGL', 'THomogeneousFltMatrix
11276:   THomogeneousMatrix', 'THomogeneousFltMatrix
```

```
11277:   TAffineMatrix', 'TAffineFltMatrix
11278:   TQuaternion', 'record Vector : TVector4f; end
11279:   TRectangle', 'record Left : Int; Top : Int; Width : integer; Height : Int; end
11280:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11281:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11282:    +',ttTranslateZ,ttPerspectiveX,ttPerspectiveY,ttPerspectiveZ, ttPerspectiveW )
11283:  'EPSILON','Extended').setExtended( 1E-100);
11284:  'EPSILON2','Extended').setExtended( 1E-50);
11285:  Func VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11286:  Func VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11287:  Func VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11288:  Func VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11289:  Func VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11290:  Func VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11291:  Func VectorAngle( V1, V2 : TAffineVector) : Single
11292:  Func VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11293:  Func VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11294:  Func VectorDotProduct( V1, V2 : TVectorGL) : Single
11295:  Func VectorLength( V : array of Single) : Single
11296:  Func VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11297:  Proc VectorNegate( V : array of Single)
11298:  Func VectorNorm( V : array of Single) : Single
11299:  Func VectorNormalize( V : array of Single) : Single
11300:  Func VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11301:  Func VectorReflect( V, N : TAffineVector) : TAffineVector
11302:  Proc VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11303:  Proc VectorScale( V : array of Single; Factor : Single)
11304:  Func VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11305:  Func CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11306:  Func CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11307:  Func CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11308:  Func CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11309:  Func CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11310:  Proc MatrixAdjoint( var M : TMatrixGL)
11311:  Func MatrixAffineDeterminant( M : TAffineMatrix) : Single
11312:  Proc MatrixAffineTranspose( var M : TAffineMatrix)
11313:  Func MatrixDeterminant( M : TMatrixGL) : Single
11314:  Proc MatrixInvert( var M : TMatrixGL)
11315:  Func MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11316:  Proc MatrixScale( var M : TMatrixGL; Factor : Single)
11317:  Proc MatrixTranspose( var M : TMatrixGL)
11318:  Func QuaternionConjugate( Q : TQuaternion) : TQuaternion
11319:  Func QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11320:  Func QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11321:  Func QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Int; t:Single):TQuaternion
11322:  Func QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11323:  Proc QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11324:  Func ConvertRotation( Angles : TAffineVector) : TVectorGL
11325:  Func CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11326:  //Func MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) :Bool
11327:  Func VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11328:  Func VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11329:  Func VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11330:  Func MakeAffineDblVector( V : array of Double) : TAffineDblVector
11331:  Func MakeDblVector( V : array of Double) : THomogeneousDblVector
11332:  Func MakeAffineVector( V : array of Single) : TAffineVector
11333:  Func MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11334:  Func MakeVector( V : array of Single) : TVectorGL
11335:  Func PointInPolygonGL( xp, yp : array of Single; x, y : Single) :Bool
11336:  Func VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11337:  Func VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11338:  Func VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11339:  Func VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11340:  Func ArcCosGL( X : Extended) : Extended
11341:  Func ArcSinGL( X : Extended) : Extended
11342:  Func ArcTan2GL( Y, X : Extended) : Extended
11343:  Func CoTanGL( X : Extended) : Extended
11344:  Func DegToRadGL( Degrees : Extended) : Extended
11345:  Func RadToDegGL( Radians : Extended) : Extended
11346:  Proc SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11347:  Func TanGL( X : Extended) : Extended
11348:  Func Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11349:  Func Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11350:  Func Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11351:  Func Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11352:  Func Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11353:  Func Roll1(Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11354: end;
11355:
11356: Proc SIRegister_JclRegistry(CL: TPSPascalCompiler);
11357: begin
11358:  Func RegCreateKey( const RootKey : HKEY; const Key, Value :Str) : Longint
11359:  Func RegDeleteEntry( const RootKey : HKEY; const Key, Name :Str) :Bool
11360:  Func RegDeleteKeyTree( const RootKey : HKEY; const Key :Str) :Bool
11361:  Func RegReadBool( const RootKey : HKEY; const Key, Name :Str) :Bool
11362:  Func RegReadBoolDef(const RootKey:HKEY; const Key, Name:Str; Def:Bool) :Bool
11363:  Func RegReadInt( const RootKey : HKEY; const Key, Name :Str) : Int
11364:  Func RegReadIntDef( const RootKey : HKEY; const Key, Name :Str; Def : Int) : Int
11365:  Func RegReadString( const RootKey : HKEY; const Key, Name :Str) :Str
```

```
11366:  Func RegReadStringDef( const RootKey : HKEY; const Key, Name, Def :Str) :Str
11367:  Func RegReadDWORD( const RootKey : HKEY; const Key, Name :Str) : Int64
11368:  Func RegReadDWORDDef(const RootKey: HKEY; const Key, Name :Str; Def:int64) : Int64
11369:  Proc RegWriteBool( const RootKey : HKEY; const Key, Name :Str; Value :Bool)
11370:  Proc RegWriteInt( const RootKey : HKEY; const Key, Name :Str; Value : Int)
11371:  Proc RegWriteString( const RootKey : HKEY; const Key, Name, Value :Str)
11372:  Proc RegWriteDWORD( const RootKey : HKEY; const Key, Name :Str; Value : Int64)
11373:  Func RegGetValueNames(const RootKey:HKEY;const Key:Str; const List : TStrings): Bool
11374:  Func RegGetKeyNames( const RootKey: HKEY;const Key:Str; const List:TStrings):Bool
11375:  Func RegHasSubKeys( const RootKey : HKEY; const Key :Str) :Bool
11376:  Func RegKeyExists( const RootKey : HKEY; const Key :Str) :Bool
11377:   AddTypeS('TExecKind',(ekMachineRun,ekMachineRunOnce,ekUserRun,ekUserRunOnce,ekServiceRun,
      ekServiceRunOnce )
11378:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11379:  Func UnregisterAutoExec( ExecKind : TExecKind; const Name :Str) :Bool
11380:  Func RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline :Str) :Bool
11381:  Func RegSaveList(const RootKey:HKEY;const Key:str;const ListName:str;const Items:TStrings):Bool;
11382:  Func RegLoadList(const RootKey:HKEY;const Key:str;const ListName:str;const SaveTo:TStrings):Bool;
11383:  Func RegDelList( const RootKey:HKEY;const Key:str;const ListName:str): Bool
11384: end;
11385:
11386: Proc SIRegister_JclCOM(CL: TPSPascalCompiler);
11387: begin
11388:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11389:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11390:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11391:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11392:   FindClass('TOBJECT'),'EInvalidParam
11393:  Func IsDCOMInstalled :Bool
11394:  Func IsDCOMEnabled :Bool
11395:  Func GetDCOMVersion :Str
11396:  Func GetMDACVersion :Str
11397:  Func GetMDACVersion2:Str
11398:  Func MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11399:  Func MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11400:  Func MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11401:  Func MarshalInterMachineInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11402:  Func MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11403:  Func CreateComponentCategory( const CatID:TGUID; const sDescription:str): HResult
11404:  Func RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID): HResult
11405:  Func UnRegisterCLSIDInCategory(const ClassID: TGUID; const CatID : TGUID): HResult
11406:  Func ResetIStreamToStart( Stream : IStream) :Bool
11407:  Func SizeOfIStreamContents( Stream : IStream) : Largeint
11408:  Func StreamToVariantArray( Stream : TStream) : OleVariant;
11409:  Func StreamToVariantArray1( Stream : IStream) : OleVariant;
11410:  Proc VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11411:  Proc VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11412: end;
11413:
11414: Proc SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11415: begin
11416:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11417:  FahrenheitFreezingPoint','Extended).setExtended( 32.0);
11418:  KelvinFreezingPoint','Extended).setExtended( 273.15);
11419:  CelsiusAbsoluteZero','Extended).setExtended( - 273.15);
11420:  FahrenheitAbsoluteZero','Extended).setExtended( - 459.67);
11421:  KelvinAbsoluteZero','Extended).setExtended( 0.0);
11422:  DegPerCycle','Extended).setExtended( 360.0);
11423:  DegPerGrad','Extended).setExtended( 0.9);
11424:  DegPerRad','Extended).setExtended( 57.295779513082320876798154814105);
11425:  GradPerCycle','Extended).setExtended( 400.0);
11426:  GradPerDeg','Extended).setExtended( 1.1111111111111111111111111111111);
11427:  GradPerRad','Extended).setExtended( 63.661977236758134307553505349006);
11428:  RadPerCycle','Extended).setExtended( 6.2831853071795864769252867665590);
11429:  RadPerDeg','Extended).setExtended( 0.017453292519943295769236907684886);
11430:  RadPerGrad','Extended).setExtended( 0.015707963267948966192313216916398);
11431:  CyclePerDeg','Extended).setExtended( 0.0027777777777777777777777777777778);
11432:  CyclePerGrad','Extended).setExtended( 0.0025);
11433:  CyclePerRad','Extended).setExtended( 0.159154943091895335768883763372 51);
11434:  ArcMinutesPerDeg','Extended).setExtended( 60.0);
11435:  ArcSecondsPerArcMinute',Extended').setExtended( 60.0);
11436:  Func HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11437:  Func MakePercentage( const Step, Max : Longint) : Longint
11438:  Func CelsiusToKelvin( const T : double) : double
11439:  Func CelsiusToFahrenheit( const T : double) : double
11440:  Func KelvinToCelsius( const T : double) : double
11441:  Func KelvinToFahrenheit( const T : double) : double
11442:  Func FahrenheitToCelsius( const T : double) : double
11443:  Func FahrenheitToKelvin( const T : double) : double
11444:  Func CycleToDeg( const Cycles : double) : double
11445:  Func CycleToGrad( const Cycles : double) : double
11446:  Func CycleToRad( const Cycles : double) : double
11447:  Func DegToCycle( const Degrees : double) : double
11448:  Func DegToGrad( const Degrees : double) : double
11449:  Func DegToRad( const Degrees : double) : double
11450:  Func GradToCycle( const Grads : double) : double
11451:  Func GradToDeg( const Grads : double) : double
11452:  Func GradToRad( const Grads : double) : double
11453:  Func RadToCycle( const Radians : double) : double
```

```
11454:  Func RadToDeg( const Radians : double) : double
11455:  Func RadToGrad( const Radians : double) : double
11456:  Func DmsToDeg( const D, M : Int; const S : double) : double
11457:  Func DmsToRad( const D, M : Int; const S : double) : double
11458:  Proc DegToDms( const Degrees : double; out D, M : Int; out S : double)
11459:  Func DegToDmsStr( const Degrees : double; const SecondPrecision :Card):Str
11460:  Proc CartesianToPolar( const X, Y : double; out R, Phi : double)
11461:  Proc PolarToCartesian( const R, Phi : double; out X, Y : double)
11462:  Proc CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11463:  Proc CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11464:  Proc CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11465:  Proc SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11466:  Func CmToInch( const Cm : double) : double
11467:  Func InchToCm( const Inch : double) : double
11468:  Func FeetToMetre(const Feet : double) : double
11469:  Func MetreToFeet(const Metre : double) : double
11470:  Func YardToMetre(const Yard : double) : double
11471:  Func MetreToYard(const Metre : double) : double
11472:  Func NmToKm(const Nm : double) : double
11473:  Func KmToNm(const Km : double) : double
11474:  Func KmToSm(const Km : double) : double
11475:  Func SmToKm(const Sm : double) : double
11476:  Func LitreToGalUs( const Litre : double) : double
11477:  Func GalUsToLitre( const GalUs : double) : double
11478:  Func GalUsToGalCan( const GalUs : double) : double
11479:  Func GalCanToGalUs( const GalCan : double) : double
11480:  Func GalUsToGalUk( const GalUs : double) : double
11481:  Func GalUkToGalUs( const GalUk : double) : double
11482:  Func LitreToGalCan( const Litre : double) : double
11483:  Func GalCanToLitre( const GalCan : double) : double
11484:  Func LitreToGalUk( const Litre : double) : double
11485:  Func GalUkToLitre( const GalUk : double) : double
11486:  Func KgToLb( const Kg : double) : double
11487:  Func LbToKg( const Lb : double) : double
11488:  Func KgToOz( const Kg : double) : double
11489:  Func OzToKg( const Oz : double) : double
11490:  Func CwtUsToKg(const Cwt : double) : double
11491:  Func CwtUkToKg(const Cwt : double) : double
11492:  Func KaratToKg(const Karat : double) : double
11493:  Func KgToCwtUs(const Kg : double) : double
11494:  Func KgToCwtUk(const Kg : double) : double
11495:  Func KgToKarat(const Kg : double) : double
11496:  Func KgToSton(const Kg : double) : double
11497:  Func KgToLton(const Kg : double) : double
11498:  Func StonToKg(const STon : double) : double
11499:  Func LtonToKg(const Lton : double) : double
11500:  Func QrUsToKg(const Qr : double) : double
11501:  Func QrUkToKg(const Qr : double) : double
11502:  Func KgToQrUs(const Kg : double) : double
11503:  Func KgToQrUk(const Kg : double) : double
11504:  Func PascalToBar( const Pa : double) : double
11505:  Func PascalToAt( const Pa : double) : double
11506:  Func PascalToTorr( const Pa : double) : double
11507:  Func BarToPascal( const Bar : double) : double
11508:  Func AtToPascal( const At : double) : double
11509:  Func TorrToPascal( const Torr : double) : double
11510:  Func KnotToMs( const Knot : double) : double
11511:  Func HpElectricToWatt( const HpE : double) : double
11512:  Func HpMetricToWatt( const HpM : double) : double
11513:  Func MsToKnot( const ms : double) : double
11514:  Func WattToHpElectric( const W : double) : double
11515:  Func WattToHpMetric( const W : double) : double
11516:  Func getBigPI:Str; //PI of 1000 numbers
11517:
11518: Proc SIRegister_devcutils(CL: TPSPascalCompiler);
11519: begin
11520:  Func CDExecuteFile( const FileName, Params, DefaultDir :Str; ShowCmd : Int) : THandle
11521:  Proc CDCopyFile( const FileName, DestName :Str)
11522:  Proc CDMoveFile( const FileName, DestName :Str)
11523:  Func MakeCommaTextToColor( Text :Str; Index : Int; DefaultColor : TColor) : TColor
11524:  Proc CDDeleteFiles( Sender : TObject; s :Str)
11525:  Func CDGetTempDir :Str
11526:  Func CDGetFileSize( FileName :Str) : longint
11527:  Func GetFileTime( FileName :Str) : longint
11528:  Func GetShortName( FileName :Str) :Str
11529:  Func GetFullName( FileName :Str) :Str
11530:  Func WinReboot :Bool
11531:  Func WinDir :Str
11532:  Func RunFile( FileToRun :Str; Params :Str; Dir :Str; Wait :Bool) :Card
11533:  Func RunFile_( Cmd, WorkDir :Str; Wait :Bool) :Bool
11534:  Func devExecutor : TdevExecutor
11535: end;
11536:
11537: Proc SIRegister_FileAssocs(CL: TPSPascalCompiler);
11538: begin
11539: Proc CheckAssociations // AssociationsCount','LongInt'( 7);
11540: Proc Associate( Index : Int)
11541: Proc UnAssociate( Index : Int)
11542: Func IsAssociated( Index : Int) :Bool
```

```
11543: Func CheckFiletype(const extension,filetype,description, verb,serverapp:str):Bool
11544: Proc RegisterFiletype(const extension,filetype,description,verb,serverapp,IcoNum:str)
11545: Proc RegisterDDEServer( const filetype, verb, topic, servername, macro :Str)
11546: Proc RefreshIcons;
11547: Func GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11548: Func MergColor(Colors: Array of TColor): TColor;
11549: Func NewColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11550: Proc DimBitmap(ABitmap: TBitmap; Value: Int);
11551: Func GrayColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11552: Func GetInverseColor(AColor: TColor): TColor;
11553: Proc GrayBitmap(ABitmap: TBitmap; Value: Int);
11554: Proc DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: Int;ShadowColor: TColor);
11555: Proc DrawCheckMark(ACanvas: TCanvas; X, Y: Int);
11556: Proc GetSystemMenuFont(Font: TFont);
11557: end;
11558:
11559: //***************************unit uPSI_JvHLParser;******************************
11560: Func IsStringConstant(const St:Str):Bool;
11561: Func IsIntConstant(const St:Str):Bool;
11562: Func IsRealConstant(const St:Str):Bool;
11563: Func IsIdentifier(const ID:Str):Bool;
11564: Func GetStringValue(const St:Str):Str;
11565: Proc ParseString(const S:Str; Ss: TStrings);
11566: Func IsStringConstantW(const St: WideString):Bool;
11567: Func IsIntConstantW(const St: WideString):Bool;
11568: Func IsRealConstantW(const St: WideString):Bool;
11569: Func IsIdentifierW(const ID: WideString):Bool;
11570: Func GetStringValueW(const St: WideString): WideString;
11571: Proc ParseStringW(const S: WideString; Ss: TStrings);
11572:
11573:
11574: //****************************unit uPSI_JclMapi;******************************
11575: Func JclSimpleSendMail( const ARecipient,AName,ASubject, ABody :Str; const AAttachment : TFileName;
       ShowDialog :Bool; AParentWND : HWND) :Bool
11576: Func JclSimpleSendFax( const ARecipient, AName,ASubject,ABody:str;const AAttachment: TFileName; ShowDialog
       :Bool; AParentWND : HWND) :Bool
11577: Func JclSimpleBringUpSendMailDialog(const ASubject,ABody:str;const AAttach:TFileName;AParentWND:HWND):Bool
11578: Func MapiCheck( const Res : DWORD; IgnoreUserAbort :Bool) : DWORD
11579: Func MapiErrorMessage( const ErrorCode : DWORD) :Str
11580:
11581: Proc SIRegister_IdNTLM(CL: TPSPascalCompiler);
11582: begin
11583:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11584:  Func BuildType1Message( ADomain, AHost :Str) :Str
11585:  Func BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:str):str
11586:  Proc RegisterAuthenticationMethod(MethodName:str; AuthClass:TIdAuthenticationClass)
11587:  Func FindAuthClass( AuthName :Str) : TIdAuthenticationClass
11588: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11589: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11590: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11591: end;
11592:
11593: Proc SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11594: begin
11595: ('IpAny','LongWord').SetUInt( $00000000);
11596:  IpLoopBack','LongWord').SetUInt( $7F000001);
11597:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11598:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11599:  PortAny','LongWord( $0000);
11600:  SocketMaxConnections','LongInt'( 5);
11601:  TIpAddr', 'LongWord
11602:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4:Byte; end
11603:  Func HostToNetLong( HostLong : LongWord) : LongWord
11604:  Func HostToNetShort( HostShort : Word) : Word
11605:  Func NetToHostLong( NetLong : LongWord) : LongWord
11606:  Func NetToHostShort( NetShort : Word) : Word
11607:  Func StrToIp( Ip :Str) : TIpAddr
11608:  Func IpToStr( Ip : TIpAddr) :Str
11609: end;
11610:
11611: (*----------------------------------------------------------------------------*)
11612: Proc SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11613: begin
11614:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11615:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11616:    +'amSha1, AlsmtpClientAuthAutoSelect )
11617:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11618:   SIRegister_TAlSmtpClient(CL);
11619: end;
11620:
11621: Proc SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11622: begin
11623: TBitNo', 'IntTStByteNo', 'IntTStationNo','IntTInOutNo','IntTIo','(EE,AA,NE,NA )
11624: TBitSet', 'set of TBitNo
11625: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11626: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11627: TBitAddr', 'LongInt
11628: TByteAddrRec', 'record Kind : TAddrKind; ByteNo: Byte; end
11629: TByteAddr', 'SmallInt
```

```
11630: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11631:  Func BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11632:  Func BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11633:  Proc BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11634:  Func BitAddrToStr( Value : TBitAddr) :Str
11635:  Func StrToBitAddr( const Value :Str) : TBitAddr
11636:  Func ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11637:  Func BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11638:  Proc ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11639:  Func ByteAddrToStr( Value : TByteAddr) :Str
11640:  Func StrToByteAddr( const Value :Str) : TByteAddr
11641:  Proc IncByteAddr( var ByteAddr : TByteAddr; Increment : Int)
11642:  Proc DecByteAddr( var ByteAddr : TByteAddr; Decrement : Int)
11643:  Func InOutStateToStr( State : TInOutState) :Str
11644:  Func MasterErrorToStr( ErrorCode : TErrorCode) :Str
11645:  Func SlaveErrorToStr( ErrorCode : TErrorCode) :Str
11646: end;
11647:
11648: Proc SIRegister_WDosTimers(CL: TPSPascalCompiler);
11649: begin
11650: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11651:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4,if2 )
11652: DpmiPmVector', 'Int64
11653:  'DInterval','LongInt'( 1000);
11654:  //'DEnabled','Boolean')BoolToStr( True);
11655:  'DIntFreq','string' if64
11656:  //'DMessages','Boolean if64);
11657:  SIRegister_TwdxCustomTimer(CL);
11658:  SIRegister_TwdxTimer(CL);
11659:  SIRegister_TwdxRtcTimer(CL);
11660:  SIRegister_TCustomIntTimer(CL);
11661:  SIRegister_TIntTimer(CL);
11662:  SIRegister_TRtcIntTimer(CL);
11663:  Func RealNow : TDateTime
11664:  Func MsToDateTime( MilliSecond : LongInt) : TDateTime
11665:  Func DateTimeToMs( Time : TDateTime) : LongInt
11666: end;
11667:
11668: Proc SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11669: begin
11670: TIdSyslogPRI', 'Int
11671: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11672:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11673:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11674:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11675:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11676: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11677:   SIRegister_TIdSysLogMsgPart(CL);
11678:   SIRegister_TIdSysLogMessage(CL);
11679:  Func FacilityToString( AFac : TIdSyslogFacility) :Str
11680:  Func SeverityToString( ASec : TIdsyslogSeverity) :Str
11681:  Func NoToSeverity( ASev : Word) : TIdSyslogSeverity
11682:  Func logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11683:  Func NoToFacility( AFac : Word) : TIdSyslogFacility
11684:  Func logFacilityToNo( AFac : TIdSyslogFacility) : Word
11685: end;
11686:
11687: Proc SIRegister_TextUtils(CL: TPSPascalCompiler);
11688: begin
11689:  'UWhitespace','String '(?:\s*)
11690:  Func StripSpaces( const AText :Str) :Str
11691:  Func CharCount( const AText :Str; Ch : Char) : Int
11692:  Func BalancedText(const AText:Str; const Ch1,Ch2: Char; const Count: Int) :Str
11693:  Func BalancedTextReg( const AText:str;const Ch1, Ch2 : Char; const Count :Int):str
11694: end;
11695:
11696: Proc SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11697: begin
11698:  ExtPascalVersion','String '0.9.8
11699:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11700:    +'Opera, brKonqueror, brMobileSafari )
11701:   AddTypeS('TCSSUnit',(cssPX,cssPerc,cssEM,cssEX,cssIN,cssCM,cssMM,cssPT, cssPC, cssnone )
11702:   AddTypeS('TExtProcedure', 'Procedure
11703:  Func DetermineBrowser( const UserAgentStr :Str) : TBrowser
11704:  Func ExtExtract(const Delims:array of string;var S:str;var Matches:TStringList;Remove:bool):bool;
11705:  Func ExtExplode( Delim : char; const S :Str; Separator : char) : TStringList
11706:  Func FirstDelimiter( const Delimiters, S :Str; Offset : Int) : Int
11707:  Func RPosEx( const Substr, Str :Str; Offset : Int) : Int
11708:  Func CountStr( const Substr, Str :Str; UntilStr :Str) : Int
11709:  Func StrToJS( const S :Str; UseBR :Bool) :Str
11710:  Func CaseOf( const S :Str; const Cases : array of string) : Int
11711:  Func RCaseOf( const S :Str; const Cases : array of string) : Int
11712:  Func EnumToJSString( TypeInfo : PTypeInfo; Value : Int) :Str
11713:  Func SetPaddings(Top:Int;Right:int;Bottom:intr;Left:int;CSSUnit:TCSSUnit;Header:bool):str;
11714:  Func SetMargins(Top:Int;Right:int;Bottom:int;Left:Int;CSSUnit:TCSSUnit;Header:bool):str;
11715:  Func ExtBefore( const BeforeS, AfterS, S :Str) :Bool
11716:  Func IsUpperCase( S :Str) :Bool
11717:  Func BeautifyJS(const AScript:str;const StartingLevel:Int;SplitHTMLNewLine:bool):str;
```

```
11718:  Func BeautifyCSS( const AStyle :Str) :Str
11719:  Func LengthRegExp( Rex :Str; CountAll :Bool) : Int
11720:  Func JSDateToDateTime( JSDate :Str) : TDateTime
11721: end;
11722:
11723: Proc SIRegister_JclShell(CL: TPSPascalCompiler);
11724: begin
11725:   TSHDeleteOption','( doSilent, doAllowUndo, doFilesOnly )
11726:   TSHDeleteOptions','set of TSHDeleteOption
11727:   TSHRenameOption','( roSilent, roRenameOnCollision )
11728:   TSHRenameOptions','set of TSHRenameOption
11729:  Func SHDeleteFiles(Parent:HWND;const Files:str;Options:TSHDeleteOptions):Boolean
11730:  Func SHDeleteFolder(Parent:HWND; const Folder:str; Options: TSHDeleteOptions): Bool
11731:  Func SHRenameFile( const Src, Dest :Str; Options : TSHRenameOptions) :Bool
11732:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11733:   TEnumFolderFlags', 'set of TEnumFolderFlag
11734:   TEnumFolderRec','record DisplayName:str; Attributes:DWORD;IconLarge:HICON;IconSmall:HICON;
       Item:PItemIdList;EnumIdList:IEnumIdList;Folder:IShellFolder;end
11735:  Func SHEnumFolderFirst(const Folder:str;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11736:  Func SHEnumSpecialFolderFirst(SpecFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11737:  Proc SHEnumFolderClose( var F : TEnumFolderRec)
11738:  Func SHEnumFolderNext( var F : TEnumFolderRec) :Bool
11739:  Func GetSpecialFolderLocation( const Folder : Int) :Str
11740:  Func DisplayPropDialog( const Handle : HWND; const FileName :Str) :Bool;
11741:  Func DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) :Bool;
11742:  Func DisplayContextMenu(const Handle:HWND; const FileName:Str; Pos:TPoint):Bool
11743:  Func OpenFolder( const Path :Str; Parent : HWND) :Bool
11744:  Func OpenSpecialFolder( FolderID : Int; Parent : HWND) :Bool
11745:  Func SHReallocMem( var P : Pointer; Count : Int) :Bool
11746:  Func SHAllocMem( out P : Pointer; Count : Int) :Bool
11747:  Func SHGetMem( var P : Pointer; Count : Int) :Bool
11748:  Func SHFreeMem( var P : Pointer) :Bool
11749:  Func DriveToPidlBind( const DriveName :Str; out Folder : IShellFolder) : PItemIdList
11750:  Func PathToPidl( const Path :Str; Folder : IShellFolder) : PItemIdList
11751:  Func PathToPidlBind( const FileName :Str; out Folder : IShellFolder) : PItemIdList
11752:  Func PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11753:  Func PidlCompare( const Pidl1, Pidl2 : PItemIdList) :Bool
11754:  Func PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) :Bool
11755:  Func PidlFree( var IdList : PItemIdList) :Bool
11756:  Func PidlGetDepth( const Pidl : PItemIdList) : Int
11757:  Func PidlGetLength( const Pidl : PItemIdList) : Int
11758:  Func PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11759:  Func PidlToPath( IdList : PItemIdList) :Str
11760:  Func StrRetFreeMem( StrRet : TStrRet) :Bool
11761:  Func StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free :Bool) :Str
11762:   PShellLink', '^TShellLink // will not work
11763:   TShellLink', 'record Arguments :Str; ShowCmd : Int; Work'
11764:    +ingDirectory :Str; IdList : PItemIDList; Target :Str; Description '
11765:    +:Str; IconLocation :Str; IconIndex : Int; HotKey : Word; end
11766:  Proc ShellLinkFree( var Link : TShellLink)
11767:  Func ShellLinkResolve( const FileName :Str; var Link : TShellLink) : HRESULT
11768:  Func ShellLinkCreate( const Link : TShellLink; const FileName :Str) : HRESULT
11769:  Func ShellLinkCreateSystem(const Link:TShellLink;const Folder:Int;const FileName:str):HRESULT;
11770:  Func ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) :Bool
11771:  Func SHDllGetVersion( const FileName :Str; var Version : TDllVersionInfo) :Bool
11772:  Func GetSystemIcon( IconIndex : Int; Flags :Card) : HICON
11773:  Func OverlayIcon( var Icon : HICON; Overlay : HICON; Large :Bool) :Bool
11774:  Func OverlayIconShortCut( var Large, Small : HICON) :Bool
11775:  Func OverlayIconShared( var Large, Small : HICON) :Bool
11776:  Func SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) :Str
11777:  Func ShellExecEx(const FileName:str;const Paramtrs:str;const Verb:str;CmdShow:Int):Bool;
11778:  Func ShellExec(Wnd:Int;const Operati,FileNam,Paramtrs,Directy:str;ShowCommand:Int):Bool;
11779:  Func ShellExecAndWait(const FileName:str;const Paramets:str;const Verb:str;CmdShow:Int):Bool;
11780:  Func ShellOpenAs( const FileName :Str) :Bool
11781:  Func ShellRasDial( const EntryName :Str) :Bool
11782:  Func ShellRunControlPanel( const NameOrFileName:str; AppletNumber:Int):Boolean
11783:  Func GetFileNameIcon( const FileName :Str; Flags :Card) : HICON
11784:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11785:  Func GetFileExeType( const FileName : TFileName) : TJclFileExeType
11786:  Func ShellFindExecutable( const FileName, DefaultDir :Str) :Str
11787:  Proc keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11788:  Func OemKeyScan( wOemChar : Word) : DWORD
11789:  Proc mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11790: end;
11791:
11792: Proc SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11793: begin
11794:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11795:  //Func xmlValidChar( const Ch : AnsiChar) :Bool;
11796:  Func xmlValidChar1( const Ch : UCS4Char) :Bool;
11797:  Func xmlValidChar2( const Ch : WideChar) :Bool;
11798:  Func xmlIsSpaceChar( const Ch : WideChar) :Bool
11799:  Func xmlIsLetter( const Ch : WideChar) :Bool
11800:  Func xmlIsDigit( const Ch : WideChar) :Bool
11801:  Func xmlIsNameStartChar( const Ch : WideChar) :Bool
11802:  Func xmlIsNameChar( const Ch : WideChar) :Bool
11803:  Func xmlIsPubidChar( const Ch : WideChar) :Bool
11804:  Func xmlValidName( const Text : UnicodeString) :Bool
11805:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
```

```
11806:  //Func xmlSkipSpace( var P : PWideChar) :Bool
11807:  //Func xmlSkipEq( var P : PWideChar) :Bool
11808:  //Func xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) :Bool
11809:  //Func xmlGetEntityEncoding(const Buf:Pointer;const BufSize:Int;out HeaderSize:Int):TUnicodeCodecClass
11810:  Func xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11811:  Func xmlTag( const Tag : UnicodeString) : UnicodeString
11812:  Func xmlEndTag( const Tag : UnicodeString) : UnicodeString
11813:  Func xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11814:  Func xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11815:  Proc xmlSafeTextInPlace( var Txt : UnicodeString)
11816:  Func xmlSafeText( const Txt : UnicodeString) : UnicodeString
11817:  Func xmlSpaceIndent( const IndentLength : Int; const IndentLevel : Int):UnicodeString
11818:  Func xmlTabIndent( const IndentLevel : Int) : UnicodeString
11819:  Func xmlComment( const Comment : UnicodeString) : UnicodeString
11820:  Proc SelfTestcXMLFunctions
11821: end;
11822:
11823: (*----------------------------------------------------------------------------*)
11824: Proc SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11825: begin
11826:  Func AWaitCursor : IUnknown
11827:  Func ChangeCursor( NewCursor : TCursor) : IUnknown
11828:  Proc SuspendRedraw( AControl : TWinControl; Suspend :Bool)
11829:  Func YesNo( const ACaption, AMsg :Str) :Bool
11830:  Proc strTokenize( const S :Str; Delims : TSysCharSet; Results : TStrings)
11831:  Func GetBorlandLibPath( Version : Int; ForDelphi :Bool) :Str
11832:  Func GetExpandedLibRoot( Version : Int; ForDelphi :Bool) :Str
11833:  Proc GetPathList( Version : Int; ForDelphi :Bool; Strings : TStrings)
11834:  Proc GetSystemPaths( Strings : TStrings)
11835:  Proc MakeEditNumeric( EditHandle : Int)
11836: end;
11837:
11838: Proc SIRegister_yuvconverts(CL: TPSPascalCompiler);
11839: begin
11840:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11841:  'BI_YUY2','LongWord( $32595559);
11842:  'BI_UYVY','LongWord').SetUInt( $59565955);
11843:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11844:  'BI_YVU9','LongWord').SetUInt( $39555659);
11845:  'BI_YUV12','LongWord( $30323449);
11846:  'BI_Y8','LongWord').SetUInt( $20203859);
11847:  'BI_Y211','LongWord').SetUInt( $31313259);
11848:  Func BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11849:  Func ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Int):Bool;
11850: end;
11851:
11852: (*----------------------------------------------------------------------------*)
11853: Proc SIRegister_AviCap(CL: TPSPascalCompiler);
11854: begin
11855:  'WM_USER','LongWord').SetUInt( $0400);
11856:  'WM_CAP_START','LongWord').SetUint($0400);
11857:  'WM_CAP_END','longword').SetUint($0400+85);
11858:  //WM_CAP_START+  85
11859:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11860:  Func capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11861:  Func capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11862:  Func capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11863:  Func capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11864:  Func capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11865:  Func capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11866:  Func capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11867:  Func capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11868:  Func capGetUserData( hwnd : THandle) : LongInt
11869:  Func capDriverConnect( hwnd : THandle; I : Word) : LongInt
11870:  Func capDriverDisconnect( hwnd : THandle) : LongInt
11871:  Func capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11872:  Func capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11873:  Func capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11874:  Func capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11875:  Func capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11876:  Func capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11877:  Func capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11878:  Func capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11879:  Func capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11880:  Func capEditCopy( hwnd : THandle) : LongInt
11881:  Func capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11882:  Func capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11883:  Func capGetAudioFormatSize( hwnd : THandle) : LongInt
11884:  Func capDlgVideoFormat( hwnd : THandle) : LongInt
11885:  Func capDlgVideoSource( hwnd : THandle) : LongInt
11886:  Func capDlgVideoDisplay( hwnd : THandle) : LongInt
11887:  Func capDlgVideoCompression( hwnd : THandle) : LongInt
11888:  Func capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11889:  Func capGetVideoFormatSize( hwnd : THandle) : LongInt
11890:  Func capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11891:  Func capPreview( hwnd : THandle; f : Word) : LongInt
11892:  Func capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11893:  Func capOverlay( hwnd : THandle; f : Word) : LongInt
11894:  Func capPreviewScale( hwnd : THandle; f : Word) : LongInt
```

```
11895:  Func capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11896:  Func capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11897:  Func capGrabFrame( hwnd : THandle) : LongInt
11898:  Func capGrabFrameNoStop( hwnd : THandle) : LongInt
11899:  Func capCaptureSequence( hwnd : THandle) : LongInt
11900:  Func capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11901:  Func capCaptureStop( hwnd : THandle) : LongInt
11902:  Func capCaptureAbort( hwnd : THandle) : LongInt
11903:  Func capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11904:  Func capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11905:  Func capCaptureSingleFrame( hwnd : THandle) : LongInt
11906:  Func capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11907:  Func capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11908:  Func capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11909:  Func capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11910:  Func capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11911:  Func capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11912:  Func capPalettePaste( hwnd : THandle) : LongInt
11913:  Func capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11914:  Func capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11915:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11916:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11917:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11918:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11919:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11920:   //PCapStatus', '^TCapStatus // will not work
11921:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11922:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11923:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11924:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11925:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11926:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11927:    +' wNumAudioAllocated : WORD; end
11928:   //PCaptureParms', '^TCaptureParms // will not work
11929:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11930:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11931:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11932:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11933:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11934:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11935:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11936:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11937:    +he : BOOL; AVStreamMaster : WORD; end
11938:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11939:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11940:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11941:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11942:  Func capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Int; nWidth, nHeight : Int;
       hwndParent : THandle; nID : Int) : THandle
11943:  Func capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Int;lpszVer:PChar;cbVer:Int):Bool;
11944:  'IDS_CAP_BEGIN','LongInt'( 300);
11945:  'IDS_CAP_END','LongInt'( 301);
11946:  'IDS_CAP_INFO','LongInt'( 401);
11947:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11948:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11949:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11950:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11951:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11952:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11953:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11954:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11955:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11956:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11957:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11958:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11959:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11960:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11961:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11962:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11963:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11964:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11965:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11966:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11967:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11968:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11969:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11970:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11971:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11972:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11973:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11974:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11975:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11976:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11977:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11978:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11979:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11980:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11981:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11982:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
```

```
11983:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11984:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11985:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11986:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11987:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11988:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11989:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11990:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11991:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11992:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11993:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11994:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11995:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11996:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11997:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11998:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11999:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
12000:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
12001:  'AVICAP32','String 'AVICAP32.dll
12002: end;
12003:
12004: Proc SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
12005: begin
12006:  Func AlBoolToInt( Value :Bool) : Int
12007:  Func ALMediumPos( LTotal, LBorder, LObject : Int) : Int
12008:  Func AlIsValidEmail( const Value : Ansistr) :Bool
12009:  Func AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
12010:  Func ALInc( var x : Int; Count : Int) : Int
12011:  Func ALCopyStr(const aSourceString: Ansistr; aStart, aLength: Int): Ansistr
12012:  Func ALGetStringFromFile(filename:AnsiStr;const ShareMode:Word= fmShareDenyWrite):AnsiStr;
12013:  Proc ALSaveStringtoFile(Str: Ansistr; filename: Ansistr);
12014:  Func ALIsInt(const S: Ansistr):Bool;
12015:  Func ALIsDecimal(const S: Ansistr):Bool;
12016:  Func ALStringToWideString(const S: Ansistr; const aCodePage: Word): WideString;
12017:  Func AlWideStringToString(const WS: WideString; const aCodePage: Word): Ansistr;
12018:  Func ALQuotedStr(const S: Ansistr; const Quote: AnsiChar = ''''): Ansistr;
12019:  Func ALDequotedStr(const S: Ansistr; AQuote: AnsiChar): Ansistr;
12020:  Func AlUTF8removeBOM(const S: Ansistr): Ansistr;
12021:  Func ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): Ansistr;
12022:  Func ALRandomStr(const aLength: Longint): Ansistr;
12023:  Func ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char):Str;
12024:  Func ALRandomStrU(const aLength: Longint):Str;
12025: end;
12026:
12027: Proc SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
12028: begin
12029:  Proc ALJSONToTStrings(const AJsonStr:Ansistr;aLst:TALStrings; const aNullStr:AnsiStr;const
       aTrueStr:Ansistr; const aFalseStr: Ansistr)
12030: end;
12031:
12032: Proc SIRegister_ALWindows(CL: TPSPascalCompiler);
12033: begin
12034:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
12035:    +RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
12036:    +ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
12037:    +; ullAvailExtendedVirtual : Int64; end
12038:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
12039:  Func ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
12040:  Func ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
12041:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
12042:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
12043:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
12044:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
12045:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
12046: end;
12047:
12048: Proc SIRegister_IPCThrd(CL: TPSPascalCompiler);
12049: begin
12050:   SIRegister_THandledObject(CL);
12051:   SIRegister_TEvent(CL);
12052:   SIRegister_TMutex(CL);
12053:   SIRegister_TSharedMem(CL);
12054:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
12055:  'TRACE_BUFFER','String 'TRACE_BUFFER
12056:  'TRACE_MUTEX','String 'TRACE_MUTEX
12057:   //PTraceEntry', '^TTraceEntry // will not work
12058:   SIRegister_TIPCTracer(CL);
12059:  'MAX_CLIENTS','LongInt'( 6);
12060:  'IPCTIMEOUT','LongInt'( 2000);
12061:  'IPCBUFFER_NAME','String 'BUFFER_NAME
12062:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
12063:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
12064:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
12065:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
12066:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
12067:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
12068:   FindClass('TOBJECT'),'EMonitorActive
12069:   FindClass('TOBJECT'),'TIPCThread
12070:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
```

```
12071:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
12072:    +'ach, evClientSwitch, evClientSignal, evClientExit )
12073:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
12074:   TClientFlags', 'set of TClientFlag
12075:   //PEventData', '^TEventData // will not work
12076:   TEventData', 'record X:SmallInt; Y:SmallInt;Flag: TClientFlag; Flags:TClientFlags; end
12077:   TConnectEvent', 'Proc ( Sender : TIPCThread; Connecting :Bool)
12078:   TDirUpdateEvent', 'Proc ( Sender : TIPCThread)
12079:   TIPCNotifyEvent', 'Proc ( Sender : TIPCThread; Data : TEventData)
12080:   //PIPCEventInfo', '^TIPCEventInfo // will not work
12081:   TIPCEventInfo','record FID:Int;FKind:TEventKind;FData:TEventData;end
12082:   SIRegister_TIPCEvent(CL);
12083:   //PClientDirRecords', '^TClientDirRecords // will not work
12084:   SIRegister_TClientDirectory(CL);
12085:   TIPCState', '( stInActive, stDisconnected, stConnected )
12086:   SIRegister_TIPCThread(CL);
12087:   SIRegister_TIPCMonitor(CL);
12088:   SIRegister_TIPCClient(CL);
12089:  Func IsMonitorRunning( var Hndl : THandle) :Bool
12090: end;
12091:
12092: (*----------------------------------------------------------------------------*)
12093: Proc SIRegister_ALGSMComm(CL: TPSPascalCompiler);
12094: begin
12095:   SIRegister_TAlGSMComm(CL);
12096:  Func AlGSMComm_BuildPDUMessage( aSMSCenter,aSMSAddress,aMessage: Ansistr) : Ansistr
12097:  Proc AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiStr;var aSMSCenter,aSMSAddress,AMessage:AnsiStr);
12098:  Func AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : Ansistr
12099:  Func AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:Ansistr;const UseGreekAlphabet:Bool):Widestring;
12100:  Func ALMatchesMask(const Filename, Mask: Ansistr):Bool;
12101: end;
12102:
12103: Proc SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
12104: begin
12105:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Int;
12106:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
12107:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
12108:   TInternetScheme', 'Int
12109:   TALIPv6Binary', 'array[1..16] of Char;
12110:  // TALIPv6Binary = array[1..16] of ansiChar;
12111:  //   TInternetScheme = Int;
12112:   SIRegister_TALHTTPRequestHeader(CL);
12113:   SIRegister_TALHTTPCookie(CL);
12114:   SIRegister_TALHTTPCookieCollection(CL);
12115:   SIRegister_TALHTTPResponseHeader(CL);
12116:  Func ALHTTPDecode( const AStr : Ansistr) : Ansistr
12117:  Proc ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
12118: //Proc ALExtractHTTPFields(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings:TALStrings;StripQuotes:Bool;
12119: //Proc ALExtractHeaderFields(Separators,WhiteSpace,Quotes:TSysCharSet;Content:PAnsiChar;
       Strings:TALStrings;Decode:Bool;StripQuotes:Bool)
12120: //Proc ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings:TALStrings;Decode:Bool;StripQuotes:Bool)
12121:  Func AlRemoveShemeFromUrl( aUrl : Ansistr) : Ansistr
12122:  Func AlExtractShemeFromUrl( aUrl : Ansistr) : TInternetScheme
12123:  Func AlExtractHostNameFromUrl( aUrl : Ansistr) : Ansistr
12124:  Func AlExtractDomainNameFromUrl( aUrl : Ansistr) : Ansistr
12125:  Func AlExtractUrlPathFromUrl( aUrl : Ansistr) : Ansistr
12126:  Func AlInternetCrackUrl(aUrl:Ansistr;var SchemeName,HostName,UserName,Password,UrlPath,ExtraInfo:
       Ansistr; var PortNumber:Int):Bool;
12127:  Func AlInternetCrackUrl1(aUrl:Ansistr;var SchemeNam,HostName,UserName,Password,UrlPath,
       Anchor:Ansistr;Query:TALStrings;var PortNumber:Int):Bool
12128:  Func AlInternetCrackUrl2(var Url:Ansistr;var Anchor:Ansistr;Query:TALStrings):Bool;
12129:  Func AlRemoveAnchorFromUrl( aUrl : Ansistr; var aAnchor : Ansistr) : Ansistr;
12130:  Func AlRemoveAnchorFromUrl1( aUrl : Ansistr) : Ansistr;
12131:  Func AlCombineUrl( RelativeUrl, BaseUrl : Ansistr) : Ansistr;
12132:  Func AlCombineUrl1(RelativeUrl,BaseUrl,Anchor:Ansistr; Query:TALStrings): Ansistr;
12133:  Func ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : Ansistr
12134:  Func ALDateTimeToRfc822Str( const aValue : TDateTime) : Ansistr
12135:  Func ALTryRfc822StrToGMTDateTime( const S : Ansistr; out Value : TDateTime) :Bool
12136:  Func ALRfc822StrToGMTDateTime( const s : Ansistr) : TDateTime
12137:  Func ALTryIPV4StrToNumeric( aIPv4Str : Ansistr; var aIPv4Num :Card) :Bool
12138:  Func ALIPV4StrToNumeric( aIPv4 : Ansistr) :Card
12139:  Func ALNumericToIPv4Str( aIPv4 :Card) : Ansistr
12140:  Func ALZeroIpV6 : TALIPv6Binary
12141:  Func ALTryIPV6StrToBinary( aIPv6Str : Ansistr; var aIPv6Bin : TALIPv6Binary) :Bool
12142:  Func ALIPV6StrTobinary( aIPv6 : Ansistr) : TALIPv6Binary
12143:  Func ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : Ansistr
12144:  Func ALBinaryStrToIPv6Binary( aIPV6BinaryStr : Ansistr) : TALIPv6Binary
12145: end;
12146:
12147: Proc SIRegister_ALFcnHTML(CL: TPSPascalCompiler);  //JavaScript
12148: begin
12149:  Proc ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const DecodeHTMLText:Bool;
12150:  Func ALUTF8ExtractHTMLText1(HtmlContent:Ansistr;const DecodeHTMLText:Bool):Ansistr;
12151:  Func ALXMLCDataElementEncode( Src : Ansistr) : Ansistr
12152:  Func ALXMLTextElementEncode(Src:Ansistr;const useNumericReference:bool):Ansistr
12153:  Func ALUTF8XMLTextElementDecode( const Src : Ansistr) : Ansistr
12154:  Func ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const useNumRef:bool):Ansistr);
```

```
12155:  Func ALUTF8HTMLDecode( const Src : Ansistr) : Ansistr
12156:  Func ALJavascriptEncode(const Src:Ansistr;const useNumericReference:bool):Ansistr
12157:  Func ALUTF8JavascriptDecode( const Src : Ansistr) : Ansistr
12158:  Proc ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:Ansistr;const
        DeleteBodyOfUnwantedTag:Bool;const ReplaceUnwantedTagCharBy:AnsiChar)
12159:  Proc ALCompactHtmlTagParams( TagParams : TALStrings)
12160: Proc ALUTF8JavascriptDecodeV( var Str : Ansistr)
12161: //'Func ALUTF8JavascriptDecode( const Src : Ansistr) : Ansistr
12162:
12163: {This Func evaluates the Javascript code given in the
12164:  parameter "aCode" and returns result. The Func works
12165:  similar to browser's console, so you can send even the code
12166:  like this "2+2" => returns "4".}
12167:
12168: Example: writeln('runJScript: '+runJS('Math.pow(2, 64)'))
12169:
12170:  Func ALRunJavascript( const aCode : Ansistr) : Ansistr
12171:  Func RunJavascript( const aCode : Ansistr) : Ansistr
12172:  Func RunJavascript2(const aCode:Ansistr):Ansistr //no CoInitialize of COM-Server
12173:  Func RunJS( const aCode : Ansistr) : Ansistr  //no CoInitialize of COM-Server
12174:  //CL.AddDelphiFunction('Proc ALHideHtmlUnwantedTagForHTMLHandleTagfunct( var HtmlContent : Ansistr; const
        DeleteBodyOfUnwantedTag:Boolean;const ReplaceUnwantedTagCharBy:AnsiChar)
12175:  //CL.AddDelphiFunction('Proc ALCompactHtmlTagParams( TagParams : TALStrings)
12176:  Func ALJavascriptEncodeU(const Src:str; const useNumericReference:boolean) :Str
12177:  Proc ALJavascriptDecodeVU( var Str :Str)
12178:  Func ALJavascriptDecodeU( const Src :Str):Str
12179: end;
12180:
12181: Proc SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12182: begin
12183:   SIRegister_TALEMailHeader(CL);
12184:   SIRegister_TALNewsArticleHeader(CL);
12185:  Func AlParseEmailAddress(FriendlyEmail:AnsiStr;var RealName:AString;const decodeRealName:Bool):Ansistr;
12186:  Func AlExtractEmailAddress( FriendlyEmail : Ansistr) : Ansistr
12187:  Func ALMakeFriendlyEmailAddress( aRealName, aEmail : Ansistr) : Ansistr
12188:  Func ALEncodeRealName4FriendlyEmailAddress( aRealName : Ansistr) : Ansistr
12189:  Func AlGenerateInternetMessageID : Ansistr;
12190:  Func AlGenerateInternetMessageID1( ahostname : Ansistr) : Ansistr;
12191:  Func ALDecodeQuotedPrintableString( src : Ansistr) : Ansistr
12192:  Func AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:Ansistr;aDefaultCodePage:Int):Ansistr;
12193: end;
12194:
12195: (*----------------------------------------------------------------------------*)
12196: Proc SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12197: begin
12198:  Func ALHostToIP( HostName : Ansistr; var Ip : Ansistr):Boolean
12199:  Func ALIPAddrToName( IPAddr : Ansistr) : Ansistr
12200:  Func ALgetLocalIPs : TALStrings
12201:  Func ALgetLocalHostName : Ansistr
12202: end;
12203:
12204: Proc SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12205: begin
12206:  Proc AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest : TALWebRequest;ServerVariables:TALStrings);
12207:  Proc AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
        TALStrings; ScriptName, ScriptFileName : Ansistr; Url : Ansistr);
12208:  Proc ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12209:  Proc AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,ScriptFileName:Ansistr;Url:AnsiStr;
12210:  Proc
        AlCGIInitServerVariablesFromWebRequest(WebRequest:TALWebRequest;ServerVariables:TALStrings;ScriptName,ScriptFileName:A
        Ansistr);
12211:  Proc AlCGIExec( InterpreterFilename: Ansistr; ServerVariables: TALStrings; RequestContentStream: Tstream;
        ResponseContentStream: Tstream; ResponseHeader: TALHTTPResponseHeader);
12212:  Proc AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:Ansistr; WebRequest :
        TALIsapiRequest; overloadedCookies:Ansistr;overloadedQueryString:Ansistr;overloadedReferer: Ansistr;'
12213:  +'overloadedRequestContentStream:Tstream;var
        ResponseContentStr:Ansistr;ResponseHeader:TALHTTPResponseHeader;
12214:  Proc AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,InterpreterFilename:Ansistr;WebRequest:
        TALIsapiRequest; var ResponseContentString : Ansistr; ResponseHeader : TALHTTPResponseHeader);
12215: end;
12216:
12217: Proc SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12218: begin
12219:   TStartupInfoA', 'TStartupInfo
12220:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12221:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12222:  SE_LOCK_MEMORY_NAME','String')( 'SeLockMemoryPrivilege
12223:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12224:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12225:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12226:  SE_TCB_NAME','String 'SeTcbPrivilege
12227:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12228:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12229:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12230:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12231:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12232:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12233:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12234:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
```

```
12235:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12236:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12237:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12238:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12239:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12240:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12241:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12242:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12243:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12244:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12245:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12246:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12247:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12248:  Func AlGetEnvironmentString : Ansistr
12249:  Func ALWinExec32(const FileName,CurrentDir,Environment:Ansistr;InStream:Tstream;OutStream:TStream):Dword;
12250:  Func ALWinExec321(const FileName:AnsiStr;InputStream:Tstream;OutputStream:TStream):Dword;
12251:  Func ALWinExecAndWait32( FileName : Ansistr; Visibility : Int) : DWORD
12252:  Func ALWinExecAndWait32V2( FileName : Ansistr; Visibility : Int) : DWORD
12253:  Func ALNTSetPrivilege( sPrivilege : Ansistr; bEnabled :Bool) :Bool
12254: end;
12255:
12256: Proc SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12257: begin
12258:  Func AlEmptyDirectory(Directory:Ansistr;SubDirectory:Bool;IgnoreFiles:array of Ansistr; const
       RemoveEmptySubDirectory :Bool; const FileNameMask : Ansistr; const MinFileAge : TdateTime):Boolean;
12259:  Func AlEmptyDirectory1( Directory : Ansistr; SubDirectory :Bool; const RemoveEmptySubDirectory:Bool;const
       FileNameMask:ansiStr;const MinFileAge:TdateTime):Bool;
12260:  Func AlCopyDirectory(SrcDirectory,DestDirectory:Ansistr;SubDirectory:Boolean; const FileNameMask:
       Ansistr; const FailIfExists:Boolean):Bool
12261:  Func ALGetModuleName : Ansistr
12262:  Func ALGetModuleFileNameWithoutExtension : Ansistr
12263:  Func ALGetModulePath : Ansistr
12264:  Func AlGetFileSize( const AFileName : Ansistr) : int64
12265:  Func AlGetFileVersion( const AFileName : Ansistr) : Ansistr
12266:  Func ALGetFileCreationDateTime( const aFileName : Ansistr) : TDateTime
12267:  Func ALGetFileLastWriteDateTime( const aFileName : Ansistr) : TDateTime
12268:  Func ALGetFileLastAccessDateTime( const aFileName : Ansistr) : TDateTime
12269:  Proc ALSetFileCreationDateTime(const aFileName:Ansistr;const aCreationDateTime:TDateTime)
12270:  Func ALIsDirectoryEmpty( const directory : Ansistr) :Bool
12271:  Func ALFileExists( const Path : Ansistr) :Bool
12272:  Func ALDirectoryExists( const Directory : Ansistr) :Bool
12273:  Func ALCreateDir( const Dir : Ansistr) :Bool
12274:  Func ALRemoveDir( const Dir : Ansistr) :Bool
12275:  Func ALDeleteFile( const FileName : Ansistr) :Bool
12276:  Func ALRenameFile( const OldName, NewName : Ansistr) :Bool
12277: end;
12278:
12279: Proc SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12280: begin
12281:   NativeInt', 'Int NativeUInt', 'Cardinal
12282:  Func ALMimeBase64EncodeString( const S : Ansistr) : Ansistr
12283:  Func ALMimeBase64EncodeStringNoCRLF( const S : Ansistr) : Ansistr
12284:  Func ALMimeBase64DecodeString( const S : Ansistr) : Ansistr
12285:  Func ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12286:  Func ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12287:  Func ALMimeBase64DecodedSize( const InputSize : NativeInt): NativeInt
12288:  Proc ALMimeBase64Encode(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12289:  Proc ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray;InputOffset:NativeInt; const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12290:  Proc ALMimeBase64EncodeFullLines( const InputBuffer: TByteDynArray; InputOffset : NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12291:  Func ALMimeBase64Decode(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt): NativeInt;
12292:  Func ALMimeBase64DecodePartial(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt;'
12293:  + var ByteBuffer:Card;var ByteBufferSpace:Card):NativeInt;
12294:  Func ALMimeBase64DecodePartialEnd(out OutputBuffer:TByteDynArray;OutputOffset:NativeInt;const
       ByteBuffer:Card;const ByteBufferSpace:Card):NativeInt;
12295:  Proc ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12296:  Proc ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12297:  Proc ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12298:  Func ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12299:  Func ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray;var ByteBuffer:Card;var ByteBufferSpace:Card) : NativeInt;
12300:  Func ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Card;const
       ByteBufferSpace:Card):NativeInt;
12301:  Proc ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12302:  Proc ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12303:  Proc ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12304:  Proc ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12305:  Proc ALMimeBase64EncodeStreamNoCRLF(const InputStream:TStream;const OutputStream:TStream)
12306:  Proc ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12307:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12308:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'(cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
```

```
12309:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12310:  Proc ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12311:  Proc ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12312:  Func ALGetDefaultFileExtFromMimeContentType( aContentType : Ansistr) : Ansistr
12313:  Func ALGetDefaultMIMEContentTypeFromExt( aExt : Ansistr) : Ansistr
12314: end;
12315:
12316: Proc SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12317: begin
12318:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12319:   FindClass('TOBJECT'),TALXMLNode
12320:   FindClass('TOBJECT'),TALXMLNodeList
12321:   FindClass('TOBJECT'),TALXMLDocument
12322:   TAlXMLParseProcessingInstructionEvent','Proc (Sender:TObject;const Target,Data:AnsiStr)
12323:   TAlXMLParseTextEvent','Proc (Sender: TObject; const str: Ansistr)
12324:   TAlXMLParseStartElementEvent','Procedure(Sender:TObject;const Name:Ansistr;const Attributes:TALStrings)
12325:   TAlXMLParseEndElementEvent','Procedure(Sender : TObject;const Name: Ansistr)
12326:   TALXmlNodeType','(ntReserved,ntElement,ntAttribute,ntText,ntCData,ntEntityRef,ntEntity,ntProcessingInstr,
       ntComment,ntDocumentntDocType,ntDocFragment,ntNotation)
12327:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12328:   TALXMLDocOptions', 'set of TALXMLDocOption
12329:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12330:   TALXMLParseOptions', 'set of TALXMLParseOption
12331:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12332:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12333:   SIRegister_EALXMLDocError(CL);
12334:   SIRegister_TALXMLNodeList(CL);
12335:   SIRegister_TALXMLNode(CL);
12336:   SIRegister_TALXmlElementNode(CL);
12337:   SIRegister_TALXmlAttributeNode(CL);
12338:   SIRegister_TALXmlTextNode(CL);
12339:   SIRegister_TALXmlDocumentNode(CL);
12340:   SIRegister_TALXmlCommentNode(CL);
12341:   SIRegister_TALXmlProcessingInstrNode(CL);
12342:   SIRegister_TALXmlCDataNode(CL);
12343:   SIRegister_TALXmlEntityRefNode(CL);
12344:   SIRegister_TALXmlEntityNode(CL);
12345:   SIRegister_TALXmlDocTypeNode(CL);
12346:   SIRegister_TALXmlDocFragmentNode(CL);
12347:   SIRegister_TALXmlNotationNode(CL);
12348:   SIRegister_TALXMLDocument(CL);
12349:  cAlXMLUTF8EncodingStr','String 'UTF-8
12350:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12351:  CALNSDelim','String ':
12352:  CALXML','String 'xml
12353:  CALVersion','String 'version
12354:  CALEncoding','String 'encoding
12355:  CALStandalone','String 'standalone
12356:  CALDefaultNodeIndent','String '
12357:  CALXmlDocument','String 'DOCUMENT
12358:  Func ALCreateEmptyXMLDocument(const Rootname: Ansistr): TalXMLDocument
12359:  Proc ALClearXMLDocument(const rootname:Ansistr;xmldoc:TalXMLDocument;const EncodingStr:AnsiStr);
12360:  Func ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,ChildNodeValue:Ansistr;const
       Recurse:Bool):TalxmlNode
12361:  Func ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : Ansistr; const
       ChildNodeName, ChildNodeValue : Ansistr; const Recurse :Bool) : TalxmlNode
12362:  Func ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:Ansistr;const
       Recurse:Bool):TalxmlNode
12363:  Func ALFindXmlNodeByNameAndAttribute(xmlrec:TalxmlNode;const NodeName:Ansistr;const AttributeName,
       AttributeValue:Ansistr;const Recurse:Bool):TalxmlNode
12364:  Func ALExtractAttrValue(const AttrName,AttrLine:Ansistr;const Default:Ansistr): Ansistr
12365: end;
12366:
12367: Proc SIRegister_TProcess(CL: TPSPascalCompiler);
12368: begin
12369:   //with RegClassS(CL,'TComponent', 'TProcess') do
12370:   with CL.AddClassN(CL.FindClass('TComponent'),'TProcess') do begin
12371:     Constructor Create( AOwner : TComponent)
12372:     Proc Free
12373:     Proc Execute
12374:     Func Resume : Integer
12375:     Func Suspend : Integer
12376:     Func Terminate( AExitCode : Integer) :Bool
12377:     Func WaitOnExit : DWord
12378:     WindowRect', 'Trect', iptrw);
12379:     StartupInfo', 'TStartupInfo', iptr);
12380:     ProcessAttributes', 'TSecurityAttributes', iptrw);
12381:     ProcessInformation', 'TProcessInformation', iptr);
12382:     Handle', 'THandle', iptr);
12383:     ThreadHandle', 'THandle', iptr);
12384:     Input', 'TOutPutPipeStream', iptr);
12385:     OutPut', 'TInputPipeStream', iptr);
12386:     StdErr', 'TinputPipeStream', iptr);
12387:     ExitStatus', 'Integer', iptr);
12388:     Active', 'Boolean', iptrw);
12389:     ApplicationName', 'String', iptrw);
12390:     CommandLine', 'String', iptrw);
12391:     ConsoleTitle', 'String', iptrw);
12392:     CurrentDirectory', 'String', iptrw);
```

```
12393:     DeskTop', 'String', iptrw);
12394:     Environment', 'TStrings', iptrw);
12395:     FillAttribute', 'Cardinal', iptrw);
12396:     InheritHandles', 'LongBool', iptrw);
12397:     Options', 'TProcessOptions', iptrw);
12398:     Priority', 'TProcessPriority', iptrw);
12399:     StartUpOptions', 'TStartUpOptions', iptrw);
12400:     Running', 'Boolean', iptr);
12401:     ShowWindow', 'TShowWindowOptions', iptrw);
12402:     ThreadAttributes', 'TSecurityAttributes', iptrw);
12403:     WindowColumns', 'Cardinal', iptrw);
12404:     WindowHeight', 'Cardinal', iptrw);
12405:     WindowLeft', 'Cardinal', iptrw);
12406:     WindowRows', 'Cardinal', iptrw);
12407:     WindowTop', 'Cardinal', iptrw);
12408:     WindowWidth', 'Cardinal', iptrw);
12409:   end;
12410: end;
12411:
12412: (*----------------------------------------------------------------------------*)
12413: Proc SIRegister_TAlWebSpider(CL: TPSPascalCompiler);
12414: begin
12415:   //with RegClassS(CL,'TObject', 'TAlWebSpider') do
12416:   with CL.AddClassN(CL.FindClass('TObject'),'TAlWebSpider') do begin
12417:     RegisterMethod('Proc Crawl
12418:     RegisterMethod('Proc UpdateLinkToLocalPath
12419:     OnCrawlBeforeDownload', 'TAlWebSpiderCrawlBeforeDownloadEvent', iptrw);
12420:     OnCrawlAfterDownload', 'TAlWebSpiderCrawlAfterDownloadEvent', iptrw);
12421:     OnCrawlDownloadSuccess', 'TAlWebSpiderCrawlDownloadSuccessEvent', iptrw);
12422:     OnCrawlDownloadRedirect', 'TAlWebSpiderCrawlDownloadRedirectEvent', iptrw);
12423:     OnCrawlDownloadError', 'TAlWebSpiderCrawlDownloadErrorEvent', iptrw);
12424:     OnCrawlGetNextLink', 'TAlWebSpiderCrawlGetNextLinkEvent', iptrw);
12425:     OnCrawlFindLink', 'TAlWebSpiderCrawlFindLinkEvent', iptrw);
12426:     OnCrawlEnd', 'TAlWebSpiderCrawlEndEvent', iptrw);
12427:     OnUpdateLinkToLocalPathGetNextFile','TAlWebSpiderUpdateLinkToLocalPathGetNextFileEvent'rw);
12428:     OnUpdateLinkToLocalPathFindLink', 'TAlWebSpiderUpdateLinkToLocalPathFindLinkEvent', iptrw);
12429:     OnUpdateLinkToLocalPathEnd', 'TAlWebSpiderUpdateLinkToLocalPathEndEvent', iptrw);
12430:     RegisterProperty('HttpClient', 'TalHttpClient', iptrw);
12431:   end;
12432: end;
12433:
12434: https://sourceforge.net/p/alcinoe/code/HEAD/tree/source/ALWebSpider.pas#1184
12435:    {----------------------------------}
12436: TAlTrivialWebSpider = Class(Tobject)
12437: Private
12438:   FWebSpider: TalWebSpider;
12439:   fStartUrl: Ansistr;
12440:   fLstUrlCrawled: TALStrings;
12441:   fLstErrorEncountered: TALStrings;
12442:   FPageDownloadedBinTree: TAlStringKeyAVLBinaryTree;
12443:   FPageNotYetDownloadedBinTree: TAlStringKeyAVLBinaryTree;
12444:   FCurrentDeepLevel: Integer;
12445:   FCurrentLocalFileNameIndex: Integer;
12446:   fMaxDeepLevel: Integer;
12447:   fOnCrawlBeforeDownload: TAlWebSpiderCrawlBeforeDownloadEvent;
12448:   fUpdateLinkToLocalPath:Bool;
12449:   fExcludeMask: Ansistr;
12450:   fStayInStartDomain:Bool;
12451:   fSaveDirectory: Ansistr;
12452:   fSplitDirectoryAmount: integer;
12453:   FHttpClient: TalHttpClient;
12454:   fIncludeMask: Ansistr;
12455:   fOnCrawlAfterDownload: TAlWebSpiderCrawlAfterDownloadEvent;
12456:   fOnCrawlFindLink: TAlTrivialWebSpiderCrawlFindLinkEvent;
12457:   fDownloadImage:Bool;
12458:   fOnUpdateLinkToLocalPathProgress: TAlTrivialWebSpiderUpdateLinkToLocalPathProgressEvent;
12459:   fOnCrawlProgress: TAlTrivialWebSpiderCrawlProgressEvent;
12460:   Proc WebSpiderCrawlDownloadError(Sender:TObject;const URL,ErrorMessage:Ansistr; HTTPResponseHeader:
       TALHTTPResponseHeader; var StopCrawling: Bool);
12461:   Proc WebSpiderCrawlDownloadRedirect(Sender:TObject;const Url,RedirectedTo:Ansistr;
12462:                        HTTPResponseHeader: TALHTTPResponseHeader; var StopCrawling:Bool);
12463:   Proc WebSpiderCrawlDownloadSuccess(Sender: TObject; const Url: Ansistr; HTTPResponseHeader:
       TALHTTPResponseHeader;HttpResponseContentTStream;var StopCrawling:Bool);
12464:  Proc WebSpiderCrawlFindLink(Sender:TObject;const HtmlTagString:AnsiStr;HtmlTagParams:
12465:                         TALStrings; const URL: Ansistr);
12466:  Proc WebSpiderCrawlGetNextLink(Sender: TObject; var Url: Ansistr);
12467:  Proc WebSpiderUpdateLinkToLocalPathFindLink(Sender:TObject;const HtmlTagString:AnsiStr;
12468:             HtmlTagParams: TALStrings;const URL: Ansistr;var LocalPath: Ansistr);
12469:  Proc WebSpiderUpdateLinkToLocalPathGetNextFile(Sender:TObject;var FileN,BaseHref:AnsiStr);
12470:     Func GetNextLocalFileName(const aContentType: Ansistr): Ansistr;
12471:   Protected
12472:   Public
12473:     Constructor Create;
12474:     Destructor Destroy; override;
12475:     Proc Crawl(const aUrl: Ansistr); overload; {Launch the Crawling of the page}
12476:   Proc Crawl(const aUrl:AnsiStr;LstUrlCrawled:TALStrings;LstErrorEncountered:TALStrings);
12477:     Property HttpClient: TalHttpClient Read FHttpClient write FHttpClient;
12478:     Property DownloadImage:Bool read fDownloadImage write fDownloadImage default false;
12479:  Property StayInStartDomain:Bool read fStayInStartDomain write fStayInStartDomain default true;
```

```
12480:     Property UpdateLinkToLocalPath:Bool read fUpdateLinkToLocalPath write fUpdateLinkToLocalPath default
       True;
12481:     Property MaxDeepLevel: Integer read fMaxDeepLevel write fMaxDeepLevel default -1;
12482:     Property ExcludeMask: Ansistr read fExcludeMask write fExcludeMask;
12483:     Property IncludeMask: Ansistr read fIncludeMask write fIncludeMask;
12484:     Property SaveDirectory: Ansistr read fSaveDirectory write fSaveDirectory;
12485:     Property SplitDirectoryAmount: integer read fSplitDirectoryAmount write fSplitDirectoryAmount default
       5000;
12486:     Property OnCrawlBeforeDownload: TAlWebSpiderCrawlBeforeDownloadEvent read fOnCrawlBeforeDownload write
       fOnCrawlBeforeDownload; {When a page is successfully downloaded}
12487:     Property OnCrawlAfterDownload: TAlWebSpiderCrawlAfterDownloadEvent read fOnCrawlAfterDownload write
       fOnCrawlAfterDownload; {When a page is successfully downloaded}
12488:     Property OnCrawlFindLink: TAlTrivialWebSpiderCrawlFindLinkEvent read fOnCrawlFindLink write
       fOnCrawlFindLink;  {When a a link is found}
12489:     Property OnCrawlProgress: TAlTrivialWebSpiderCrawlProgressEvent read fOnCrawlProgress write
       fOnCrawlProgress;
12490:     Property OnUpdateLinkToLocalPathProgress: TAlTrivialWebSpiderUpdateLinkToLocalPathProgressEvent read
       fOnUpdateLinkToLocalPathProgress write fOnUpdateLinkToLocalPathProgress;
12491:   end;
12492:
12493: Proc SIRegister_ALOpenOffice(CL: TPSPascalCompiler);
12494: begin
12495:  CL.AddClassN(CL.FindClass('TOBJECT'),'EALOpenOfficeException
12496:  Proc ConnectOpenOffice
12497:  Proc DisconnectOpenOffice( aTerminateOpenOffice :Bool)
12498:  Func IsOpenOfficeConnected :Bool
12499:  Func CreateUnoService( const aServiceName : Ansistr) : Variant
12500:  Func CreateUnoStruct(const aStructureName:Ansistr;const aMaxIndex:integer): Variant
12501:  Func HasUnoInterfaces( aObject: Variant;aInterfaceList: array of Ansistr):Bool
12502:  Func CreateOOProperties( aPropertyList : array of Variant) : Variant
12503:  Func MakeOOPropertyValue(aPropertyName: Ansistr;aPropertyValue: Variant): Variant
12504:  Func CreateOOCalcDocument : Variant
12505:  Func CreateOOWordDocument: Variant;
12506:  Func CreateOOImpressDocument : Variant
12507:  Func CreateOODrawDocument : Variant
12508:  Proc SaveOODocument(aDocument: Variant; aFileName: Ansistr;aFileType : Ansistr)
12509:  Proc CreateOOSheet( aDocument : Variant; const aSheetName : Ansistr)
12510:  Proc SetColumnWidth(aSheet:Variant;const aColumnIndex:int;const aWidthInCentimetres:int);
12511:  Proc SetColumnWidth1(aSheet:Variant;aColumn:Variant;const aWidthInCentimetres:integer);
12512:  Proc SetCellBold( aCell : Variant)
12513:  Proc SetCellBorder( aCellRange : Variant; const aBorderColor : Longword)
12514:  Func IsVariantNullOrEmpty( aVariant : Variant) :Bool
12515:  Func VarDummyArray : Variant
12516:  Func ConvertToURL( aWinAddress : Ansistr) : Ansistr
12517:  Func ConvertFromURL( aUrlAddress : Ansistr) : Ansistr
12518:  Func OORGB( aRedByte, aGreenByte, aBlueByte : byte) : Longword
12519:  Func ALCopyStr2(const aSourceString: Ansistr; aStart, aLength: Integer): Ansistr;
12520: end;
12521:
12522: Proc SIRegister_ALExecute2(CL: TPSPascalCompiler);
12523: begin
12524:   //CL.AddTypeS('TStartupInfoA', 'TStartupInfo
12525:  CL.AddConstantN('SE_CREATE_TOKEN_NAME','String').SetString( 'SeCreateTokenPrivilege
12526:  SE_ASSIGNPRIMARYTOKEN_NAME','String').SetString( 'SeAssignPrimaryTokenPrivilege
12527:  SE_LOCK_MEMORY_NAME','String').SetString( 'SeLockMemoryPrivilege
12528:  SE_INCREASE_QUOTA_NAME','String').SetString( 'SeIncreaseQuotaPrivilege
12529:  SE_UNSOLICITED_INPUT_NAME','String').SetString( 'SeUnsolicitedInputPrivilege
12530:  SE_MACHINE_ACCOUNT_NAME','String').SetString( 'SeMachineAccountPrivilege
12531:  Func AlGetEnvironmentStringX : Ansistr
12532:  Func ALWinExec0X(const aCommandLine:Ansistr;const aCurrentDirectory:Ansistr;const
       aEnvironment:Ansistr;const aInputStream:Tstream;const aOutputStream Stream;const aOwnerThread
       TThread):Dword;
12533:  Func ALWinExec1X(const aCommandLine: Ansistr; const aInputStream: Tstream; const aOutputStream: TStream;
       const aOwnerThread: TThread): Dword;
12534: Proc ALWinExec2X( const aCommandLine:Ansistr; const aCurrentDirectory : Ansistr);
12535: Proc ALWinExec3X(const aUserName:Ansistr;const aPassword:Ansistr;const aCommandLine:Ansistr;const
       aCurrentDirectory:Ansistr;const aLogonFlags:dword);
12536: Func ALWinExecAndWait4X(const aCommandLine: Ansistr;const aCurrentDirectory:Ansistr;const
       aEnvironment:Ansistr;const aVisibility:integer): DWORD;
12537:  Func ALWinExecAndWait5X(const aCommandLine:Ansistr; const aVisibility:integer):DWORD;
12538:  Func ALWinExecAndWaitV2X(const aCommandLine:Ansistr;const aVisibility integer):DWORD
12539:  Func ALNTSetPrivilegeX( const sPrivilege : Ansistr; bEnabled :Bool) :Bool
12540:  Func ALStartServiceX(const aServiceName:Ansistr;const aComputerName:Ansistr;const aTimeOut:integer):bool;
12541:  Func ALStopServiceX(const aServiceName:Ansistr;const aComputerName:Ansistr;const aTimeOut:integer):bool;
12542:  Func ALMakeServiceAutorestartingX( const aServiceName : Ansistr; const aComputerName : Ansistr; const
       aTimeToRestartInSec:integer; const aTimeToResetInSec: integer):Bool
12543: end;
12544:
12545: Proc SIRegister_ALIsapiHTTP(CL: TPSPascalCompiler);
12546: begin
12547: CL.AddClassN(CL.FindClass('TOBJECT'),'EALIsapiRequestContentSizeTooBig
12548: CL.AddClassN(CL.FindClass('TOBJECT'),'EALIsapiRequestConnectionDropped
12549: SIRegister_TALWebRequest(CL);
12550: SIRegister_TALISAPIRequest(CL);
12551: SIRegister_TALWebResponse(CL);
12552: SIRegister_TALISAPIResponse(CL);
12553:  Func ALIsapiHttpStatusString( StatusCode : Integer) : Ansistr
12554:  CL.AddConstantN('HSE_IO_SYNC','LongWord').SetUInt( $00000001);
12555:  'HSE_IO_ASYNC','LongWord').SetUInt( $00000002);
```

```
12556:  'HSE_IO_DISCONNECT_AFTER_SEND','LongWord').SetUInt( $00000004);
12557:  'HSE_IO_SEND_HEADERS','LongWord').SetUInt( $00000008);
12558: end;
12559:
12560: Proc SIRegister_uUsb(CL: TPSPascalCompiler);
12561: begin
12562:   //CL.AddTypeS('PDevBroadcastHdr', '^DEV_BROADCAST_HDR // will not work
12563:   CL.AddTypeS(DEV_BROADCAST_HDR','record dbch_size:DWORD;dbch_devicetype:DWORD;dbch_reserved:DWORD;end;
12564:   CL.AddTypeS('TDevBroadcastHdr', 'DEV_BROADCAST_HDR
12565:   //CL.AddTypeS('PDevBroadcastDeviceInterface','^DEV_BROADCAST_DEVICEINTERFACE//will not work
12566:   CL.AddTypeS(DEV_BROADCAST_DEVICEINTERFACE', 'record dbcc_size : DWORD; dbcc_'
12567:    +devicetype: DWORD;dbcc_reserved:DWORD; dbcc_classguid: TGUID; dbcc_name:Char; end
12568:   CL.AddTypeS('TDevBroadcastDeviceInterface', 'DEV_BROADCAST_DEVICEINTERFACE
12569:  //CL.AddConstantN('GUID_DEVINTERFACE_USB_DEVICE','TGUID').SetString( '{A5DCBF10-6530-11D2-901F-
      00C04FB951ED}
12570:  'DBT_DEVICEARRIVAL','LongWord').SetUInt( $8000);
12571:  'DBT_DEVICEREMOVECOMPLETE','LongWord').SetUInt( $8004);
12572:  'DBT_DEVTYP_DEVICEINTERFACE','LongWord').SetUInt( $00000005);
12573:   CL.AddTypeS('TUsbNotifyProc', 'Proc ( Sender : TObject; const DeviceName:Str)
12574:   SIRegister_TUsbNotifier(CL);
12575: end;
12576:
12577: Proc SIRegister_uWebcam(CL: TPSPascalCompiler);
12578: begin
12579:  CL.AddConstantN('WM_CAP_START','LongWord').SetUInt( $0400);
12580:  'WM_CAP_DRIVER_CONNECT','LongInt').SetInt( $0400 + 10);
12581:  'WM_CAP_DRIVER_DISCONNECT','LongInt').SetInt( $0400 + 11);
12582:  'WM_CAP_SAVEDIB','LongInt').SetInt( $0400 + 25);
12583:  'WM_CAP_GRAB_FRAME','LongInt').SetInt( $0400 + 60);
12584:  'WM_CAP_STOP','LongInt').SetInt( $0400 + 68);
12585:  Func Connectwebcam( WebcamID : integer) :Bool
12586:  Proc CaptureWebCam( FilePath :Str)
12587:  Proc CloseWebcam( );
12588:  Proc WebcamInit;
12589:  Func WebcamList: TStringlist;
12590: end;
12591:
12592: Proc WebcamInit;
12593: begin
12594:   LibHandle := LoadLibrary('avicap32.dll
12595:   CapGetDriverDescriptionA:= GetProcAddress(LibHandle,'capGetDriverDescriptionA
12596:   CapCreateCaptureWindowA := GetProcAddress(LibHandle,'capCreateCaptureWindowA
12597: end;
12598: Example: webcaminit;
12599:          writeln(webcamlist.text)
12600:          if Connectwebcam(0) then CaptureWebCam(Exepath+'foto4.png
12601:          CloseWebcam();
12602:
12603: Top5 Functions mX4.7.1.80
12604: Proc LoadResourceFile3HTML(aFile:str; ms:TMemoryStream);
12605: Func VarArrayToStr(const vArray: variant):Str;
12606: Func VarStrNull(const V:OleVariant):str; //avoid problems with null strings
12607: Func GetWMIObject(const objectName:Str): IDispatch; //create the Wmi instance
12608: Func GetAntiVirusProductInfo: TStringlist;
12609: Func EnDeCrypt(const Value :Str) :Str;
12610: //http://www.delphibasics.info/home/delphibasicssnippets/executingpreparedshellcodeindelphi
12611:
12612: Proc SIRegister_TeCanvas(CL: TPSPascalCompiler);
12613: //based on TEEProc, TeCanvas, TEEngine, TChart
12614: begin
12615:  'TeePiStep','Double').setExtended( Pi / 180.0);
12616:  'TeeDefaultPerspective','LongInt'( 100);
12617:  'TeeMinAngle','LongInt'( 270);
12618:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12619:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12620:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12621:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12622:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12623:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12624:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12625:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12626:  'TA_LEFT','LongInt'( 0);
12627:  'TA_RIGHT','LongInt'( 2);
12628:  'TA_CENTER','LongInt'( 6);
12629:  'TA_TOP','LongInt'( 0);
12630:  'TA_BOTTOM','LongInt'( 8);
12631:  'teePATCOPY','LongInt'( 0);
12632:  'NumCirclePoints','LongInt'( 64);
12633:  'teeDEFAULT_CHARSET','LongInt'( 1);
12634:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12635:  'TA_LEFT','LongInt'( 0);
12636:  'bs_Solid','LongInt'( 0);
12637:  'teepf24Bit','LongInt'( 0);
12638:  'teepfDevice','LongInt'( 1);
12639:  'CM_MOUSELEAVE','LongInt'( 10000);
12640:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12641:  'DC_BRUSH','LongInt'( 18);
12642:  'DC_PEN','LongInt'( 19);
12643:   teeCOLORREF', 'LongWord
```

```
12644:  TLogBrush', 'record lbStyle : Int; lbColor : TColor; lbHatch: Int; end
12645:  //TNotifyEvent', 'Proc ( Sender : TObject)
12646:  SIRegister_TFilterRegion(CL);
12647:  SIRegister_IFormCreator(CL);
12648:  SIRegister_TTeeFilter(CL);
12649:  //TFilterClass', 'class of TTeeFilter
12650:  SIRegister_TFilterItems(CL);
12651:  SIRegister_TConvolveFilter(CL);
12652:  SIRegister_TBlurFilter(CL);
12653:  SIRegister_TTeePicture(CL);
12654:  TPenEndStyle', '( esRound, esSquare, esFlat )
12655:  SIRegister_TChartPen(CL);
12656:  SIRegister_TChartHiddenPen(CL);
12657:  SIRegister_TDottedGrayPen(CL);
12658:  SIRegister_TDarkGrayPen(CL);
12659:  SIRegister_TWhitePen(CL);
12660:  SIRegister_TChartBrush(CL);
12661:  TTeeView3DScrolled', 'Proc ( IsHoriz :Bool)
12662:  TTeeView3DChangedZoom', 'Proc ( NewZoom : Int)
12663:  SIRegister_TView3DOptions(CL);
12664:  FindClass('TOBJECT'),'TTeeCanvas
12665:  TTeeTransparency', 'Int
12666:  SIRegister_TTeeBlend(CL);
12667:  FindClass('TOBJECT'),'TCanvas3D
12668:  SIRegister_TTeeShadow(CL);
12669:  teeTGradientDirection','(gdTopBottom,gdBottomTop,gdLeftRight,gdRightLeft,gdFromCenter, gdFromTopLeft,
       gdFromBottomLeft,gdRadial,gdDiagonalUp,gdDiagonalDown )
12670:  FindClass('TOBJECT'),'TSubGradient
12671:  SIRegister_TCustomTeeGradient(CL);
12672:  SIRegister_TSubGradient(CL);
12673:  SIRegister_TTeeGradient(CL);
12674:  SIRegister_TTeeFontGradient(CL);
12675:  SIRegister_TTeeFont(CL);
12676:  TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12677:  TCanvasTextAlign', 'Int
12678:   TTeeCanvasHandle', 'HDC
12679:   SIRegister_TTeeCanvas(CL);
12680:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12681:   SIRegister_TFloatXYZ(CL);
12682:   TPoint3D', 'record x : Int; y : Int; z : Int; end
12683:   TRGB', 'record blue: byte; green: byte; red: byte; end
12684:   {TRGB=packed record
12685:     Blue  : Byte;
12686:     Green : Byte;
12687:     Red   : Byte;
12688:  //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12689:
12690: TTeeCanvasCalcPoints','Function(x,z:Int;var P0,P1:TPoint3D;var Color0,Color1:TColor):Bool
12691: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12692: TCanvas3DPlane', '( cpX, cpY, cpZ )
12693:   //IInterface', 'IUnknown
12694:   SIRegister_TCanvas3D(CL);
12695:   SIRegister_TTeeCanvas3D(CL);
12696:   TTrianglePoints', 'Array[0..2] of TPoint;
12697:   TFourPoints', 'Array[0..3] of TPoint;
12698:  Func ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12699:  Func ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12700:  Func Point3D( const x, y, z : Int) : TPoint3D
12701:  Proc SwapDouble( var a, b : Double)
12702:  Proc SwapInt( var a, b : Int)
12703:  Proc RectSize( const R : TRect; var RectWidth, RectHeight : Int)
12704:  Proc teeRectCenter( const R : TRect; var X, Y : Int)
12705:  Func RectFromPolygon( const Points : array of TPoint; NumPoints : Int): TRect
12706:  Func RectFromTriangle( const Points : TTrianglePoints) : TRect
12707:  Func RectangleInRectangle( const Small, Big : TRect) :Bool
12708:  Proc ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12709:  Proc UnClipCanvas( ACanvas : TCanvas)
12710:  Proc ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12711:  Proc ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Int)
12712:  Proc ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Int)
12713:  'TeeCharForHeight','String 'W
12714:  'DarkerColorQuantity','Byte').SetUInt( 128);
12715:  'DarkColorQuantity','Byte').SetUInt( 64);
12716:   TButtonGetColorProc', 'Func  : TColor
12717:   SIRegister_TTeeButton(CL);
12718:   SIRegister_TButtonColor(CL);
12719:   SIRegister_TComboFlat(CL);
12720:  Proc TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12721:  Func TeePoint( const aX, aY : Int) : TPoint
12722:  Func TEEPointInRect( const Rect : TRect; const P : TPoint) :Bool;
12723:  Func PointInRect1( const Rect : TRect; x, y : Int) :Bool;
12724:  Func TeeRect( Left, Top, Right, Bottom : Int) : TRect
12725:  Func OrientRectangle( const R : TRect) : TRect
12726:  Proc TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Int)
12727:  Func PolygonBounds( const P : array of TPoint) : TRect
12728:  Func PolygonInPolygon( const A, B : array of TPoint) :Bool
12729:  Func RGBValue( const Color : TColor) : TRGB
12730:  Func EditColor( AOwner : TComponent; AColor : TColor) : TColor
12731:  Func EditColorDialog( AOwner : TComponent; var AColor : TColor) :Bool
```

```
12732:  Func PointAtDistance( AFrom, ATo : TPoint; ADist : Int) : TPoint
12733:  Func TeeCull( const P : TFourPoints) :Bool;
12734:  Func TeeCull1( const P0, P1, P2 : TPoint) :Bool;
12735:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12736:  Proc SmoothStretch( Src, Dst : TBitmap);
12737:  Proc SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12738:  Func TeeDistance( const x, y : Double) : Double
12739:  Func TeeLoadLibrary( const FileName :Str) : HInst
12740:  Proc TeeFreeLibrary( hLibModule : HMODULE)
12741:  Proc TeeBlendBitmaps(const Percent:Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12742:  //Proc TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12743:  Proc TeeShadowSmooth(Bitmap,Back:TBitmap;Left,Top,Width,Height,horz,
       vert:Int;Smoothness:Double;FullDraw:Bool;ACanvas:TCanvas3D;Clip:Bool)
12744:   SIRegister_ICanvasHyperlinks(CL);
12745:   SIRegister_ICanvasToolTips(CL);
12746:  Func Supports( const Instance : IInterface; const IID : TGUID) :Bool
12747: end;
12748:
12749: Proc SIRegister_ovcmisc(CL: TPSPascalCompiler);
12750: begin
12751:   TOvcHdc', 'Int
12752:   TOvcHWND', 'Cardinal
12753:   TOvcHdc', 'HDC
12754:   TOvcHWND', 'HWND
12755:  Func LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12756:  Func LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12757:  Func ovCompStruct( const S1, S2, Size :Card) : Int
12758:  Func DefaultEpoch : Int
12759:  Func DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12760:  Proc FixRealPrim( P : PChar; DC : Char)
12761:  Func GetDisplayString(Canvas: TCanvas; const S:Str; MinChars, MaxWidth : Int) :Str
12762:  Func GetLeftButton : Byte
12763:  Func GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12764:  Proc GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12765:  Func GetShiftFlags : Byte
12766:  Func ovCreateRotatedFont( F : TFont; Angle : Int) : hFont
12767:  Func GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Int;Ctl3D:Boolean): Int
12768:  Func ovExtractWord( N : Int; const S :Str; WordDelims : TCharSet) :Str
12769:  Func ovIsForegroundTask :Bool
12770:  Func ovTrimLeft( const S :Str) :Str
12771:  Func ovTrimRight( const S :Str) :Str
12772:  Func ovQuotedStr( const S :Str) :Str
12773:  Func ovWordCount( const S :Str; const WordDelims : TCharSet) : Int
12774:  Func ovWordPosition(const N:Int;const S:Str;const WordDelims : TCharSet) : Int
12775:  Func PtrDiff( const P1, P2 : PChar) : Word
12776:  Proc PtrInc( var P, Delta : Word)
12777:  Proc PtrDec( var P, Delta : Word)
12778:  Proc FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Int)
12779:  Proc TransStretchBlt(DstDC:TOvcHdc;DstX,DstY,DstW,DstH:Int;SrcDC:TOvcHdc;SrcX,SrcY,SrcW,
       SrcH:Int;MaskDC:TOvcHdc;MaskX,MaskY:Int)
12780:  Func ovMinI( X, Y : Int) : Int
12781:  Func ovMaxI( X, Y : Int) : Int
12782:  Func ovMinL( X, Y : LongInt) : LongInt
12783:  Func ovMaxL( X, Y : LongInt) : LongInt
12784:  Func GenerateComponentName( PF : TWinControl; const Root :Str) :Str
12785:  Func PartialCompare( const S1, S2 :Str) :Bool
12786:  Func PathEllipsis( const S :Str; MaxWidth : Int) :Str
12787:  Func ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12788:  Proc ovCopyParentImage( Control : TControl; Dest : TCanvas)
12789:  Proc ovDrawTransparentBitmap(Dest:TCanvas;X,Y,W,H:Int;Rect:TRect;Bitmap:TBitmap; TransparentColor:TColor)
12790:  Proc DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,
       Height:Int;Rect:TRect;TransparentColor : TColorRef)
12791:  Func ovWidthOf( const R : TRect) : Int
12792:  Func ovHeightOf( const R : TRect) : Int
12793:  Proc ovDebugOutput( const S :Str)
12794:  Func GetArrowWidth( Width, Height : Int) : Int
12795:  Proc StripCharSeq( CharSeq :Str; var Str :Str)
12796:  Proc StripCharFromEnd( aChr : Char; var Str :Str)
12797:  Proc StripCharFromFront( aChr : Char; var Str :Str)
12798:  Func SystemParametersInfo(uiAction,uiParam: UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12799:  Func SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12800:  Func SystemParametersInfoA(uiAction,uiParam: UINT; pvParam: UINT; fWinIni : UINT) : BOOL
12801:  Func CreateEllipticRgn( p1, p2, p3, p4 : Int) : HRGN
12802:  Func CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12803:  Func CreateFontIndirect( const p1 : TLogFont) : HFONT
12804:  Func CreateMetaFile( p1 : PChar) : HDC
12805:  Func DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12806:  Func DrawText(hDC:HDC;lpString:PChar;nCount:Int;var lpRect: TRect;uFormat:UINT):Int
12807:  Func DrawTextS(hDC:HDC;lpString:str;nCount:Int;var lpRect:TRect;uFormat:UINT):Int
12808:  Func SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12809:  Func SetGraphicsMode( hdc : HDC; iMode : Int) : Int
12810:  Func SetMapMode( DC : HDC; p2 : Int) : Int
12811:  Func SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12812:  //Func SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12813:  Func SetPixel( DC : HDC; X, Y : Int; Color : COLORREF) : COLORREF
12814:  Func SetPixelV( DC : HDC; X, Y : Int; Color : COLORREF) : BOOL
12815:  //Func SetPixelFormat(DC:HDC;PixelFormat:Int;FormatDef:PPixelFormatDescriptor) : BOOL
12816:  Func SetPolyFillMode( DC : HDC; PolyFillMode : Int) : Int
12817:  Func StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,SrcHeight:Int;Rop:DWORD):BOOL
```

```
12818:  Func SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Int) : BOOL
12819:  Func StretchDIBits(DC:HDC;DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,SrcHeight:Int;Bits:int;var
       BitsInfo:TBitmapInfo;Usage:UINT;Rop:DWORD):Int
12820:  Func SetROP2( DC : HDC; p2 : Int) : Int
12821:  Func SetStretchBltMode( DC : HDC; StretchMode : Int) : Int
12822:  Func SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12823:  Func SetTextCharacterExtra( DC : HDC; CharExtra : Int) : Int
12824:  Func SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12825:  Func SetTextAlign( DC : HDC; Flags : UINT) : UINT
12826:  Func SetTextJustification( DC : HDC; BreakExtra, BreakCount : Int) : Int
12827:  Func UpdateColors( DC : HDC) : BOOL
12828:  Func GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12829:  Func GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12830:  Func GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12831:  Func GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12832:  Func IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Int) : Int
12833:  Func InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12834:  Func MaskBlt(DestDC:HDC;XDest,YDest,Width,Height:Int;SrcDC:HDC;XScr,YScr:Int; Mask:HBITMAP;xMask,
       yMask:Int;Rop: DWORD): BOOL
12835:  Func PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12836:  Func OffsetClipRgn( DC : HDC; XOffset, YOffset : Int) : Int
12837:  Func OffsetRgn( RGN : HRGN; XOffset, YOffset : Int) : Int
12838:  Func PatBlt( DC : HDC; X, Y, Width, Height : Int; Rop : DWORD) : BOOL
12839:  Func Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Int) : BOOL
12840:  Func PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12841:  Func PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12842:  Func PtInRegion( RGN : HRGN; X, Y : Int) : BOOL
12843:  Func PtVisible( DC : HDC; X, Y : Int) : BOOL
12844:  Func RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12845:  Func RectVisible( DC : HDC; const Rect : TRect) : BOOL
12846:  Func Rectangle( DC : HDC; X1, Y1, X2, Y2 : Int) : BOOL
12847:  Func RestoreDC( DC : HDC; SavedDC : Int) : BOOL
12848: end;
12849:
12850: Proc SIRegister_ovcfiler(CL: TPSPascalCompiler);
12851: begin
12852: SIRegister_TOvcAbstractStore(CL);
12853: //PExPropInfo', '^TExPropInfo // will not work
12854: // TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12855:   SIRegister_TOvcPropertyList(CL);
12856:   SIRegister_TOvcDataFiler(CL);
12857:  Proc UpdateStoredList(AForm: TWinControl; AStoredList : TStrings; FromForm :Bool)
12858:  Proc UpdateStoredList1( AForm : TCustomForm; AStoredList: TStrings;FromForm:Bool)
12859:  Func CreateStoredItem( const CompName, PropName :Str) :Str
12860:  Func ParseStoredItem( const Item :Str; var CompName, PropName :Str) :Bool
12861:  //Func GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12862: end;
12863:
12864: Proc SIRegister_ovccoco(CL: TPSPascalCompiler);
12865: begin
12866:  'ovsetsize','LongInt'( 16);
12867:  'etSyntax','LongInt'( 0);
12868:  'etSymantic','LongInt'( 1);
12869:  'chCR',Char #13);
12870:  'chLF',Char #10);
12871:  'chLineSeparator', chCR);
12872:   SIRegister_TCocoError(CL);
12873:   SIRegister_TCommentItem(CL);
12874:   SIRegister_TCommentList(CL);
12875:   SIRegister_TSymbolPosition(CL);
12876: TGenListType', '( glNever, glAlways, glOnError )
12877: TovBitSet', 'set of Int
12878: //PStartTable', '^TStartTable // will not work
12879: TovCharSet', 'set of AnsiChar
12880: TAfterGenListEvent','Proc ( Sender : TObject; var PrintErrorCount :Bool)
12881: TCommentEvent','Proc ( Sender : TObject; CommentList : TCommentList)
12882: TCustomErrorEvent','Function(Sender:TObject;const ErrorCode:longint;const Data:str):str
12883: TovErrorEvent', 'Proc ( Sender : TObject; Error : TCocoError)
12884: TovErrorProc','Proc (ErrorCode:Int;Symbol:TSymbolPosition;const Data:str;ErrorType:Int)
12885: TFailureEvent','Proc (Sender : TObject; NumErrors : Int)
12886: TGetCH', 'Func (pos : longint) : char
12887: TStatusUpdateProc', 'Proc ( Sender : TObject; Status:Str; LineNum:Int)
12888:   SIRegister_TCocoRScanner(CL);
12889:   SIRegister_TCocoRGrammar(CL);
12890:  '_EF','Char #0);
12891:  '_TAB','Char').SetString( #09);
12892:  '_CR','Char').SetString( #13);
12893:  '_LF','Char').SetString( #10);
12894:  '_EL','').SetString( _CR);
12895:  '_EOF','Char').SetString( #26);
12896:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12897:  'minErrDist','LongInt'( 2);
12898:  Func ovPadL(S:Str; ch : char; L : Int) :Str
12899: end;
12900:
12901:   TFormatSettings = record
12902:     CurrencyFormat: Byte;
12903:     NegCurrFormat: Byte;
```

```
12904:     ThousandSeparator: Char;
12905:     DecimalSeparator: Char;
12906:     CurrencyDecimals: Byte;
12907:     DateSeparator: Char;
12908:     TimeSeparator: Char;
12909:     ListSeparator: Char;
12910:     CurrencyString:Str;
12911:     ShortDateFormat:Str;
12912:     LongDateFormat:Str;
12913:     TimeAMString:Str;
12914:     TimePMString:Str;
12915:     ShortTimeFormat:Str;
12916:     LongTimeFormat:Str;
12917:     ShortMonthNames: array[1..12] of string;
12918:     LongMonthNames: array[1..12] of string;
12919:     ShortDayNames: array[1..7] of string;
12920:     LongDayNames: array[1..7] of string;
12921:     TwoDigitYearCenturyWindow: Word;
12922:   end;
12923:
12924: Proc SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12925: begin
12926:  Func ovFormatSettings : TFormatSettings
12927: end;
12928:
12929: Proc SIRegister_ovcstr(CL: TPSPascalCompiler);
12930: begin
12931:   TOvcCharSet', 'set of Char
12932:   ovBTable', 'array[0..255] of Byte
12933:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12934:  Func BinaryBPChar( Dest : PChar; B : Byte) : PChar
12935:  Func BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12936:  Func BinaryWPChar( Dest : PChar; W : Word) : PChar
12937:  Proc BMMakeTable( MatchString : PChar; var BT : ovBTable)
12938:  Func BMSearch(var Buffer,BufLength:Card;var BT:ovBTable;MatchString:PChar;var Pos:Card):Bool;
12939:  Func BMSearchUC(var Buffer,BufLength:Card;var BT:ovBTable;MatchString:PChar;var Pos:Card):Bool
12940:  Func CharStrPChar( Dest : PChar; C : Char; Len :Card) : PChar
12941:  Func DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12942:  Func HexBPChar( Dest : PChar; B : Byte) : PChar
12943:  Func HexLPChar( Dest : PChar; L : LongInt) : PChar
12944:  Func HexPtrPChar( Dest : PChar; P : TObject) : PChar
12945:  Func HexWPChar( Dest : PChar; W : Word) : PChar
12946:  Func LoCaseChar( C : Char) : Char
12947:  Func OctalLPChar( Dest : PChar; L : LongInt) : PChar
12948:  Func StrChDeletePrim( P : PChar; Pos :Card) : PChar
12949:  Func StrChInsertPrim( Dest : PChar; C : Char; Pos:Card) : PChar
12950:  Func StrChPos( P : PChar; C : Char; var Pos :Card) :Bool
12951:  Proc StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12952:  Func StrStCopy( Dest, S : PChar; Pos, Count :Card) : PChar
12953:  Func StrStDeletePrim( P : PChar; Pos, Count :Card) : PChar
12954:  Func StrStInsert( Dest, S1, S2 : PChar; Pos :Card) : PChar
12955:  Func StrStInsertPrim( Dest, S : PChar; Pos :Card) : PChar
12956:  Func StrStPos( P, S : PChar; var Pos :Card) :Bool
12957:  Func StrToLongPChar( S : PChar; var I : LongInt) :Bool
12958:  Proc TrimAllSpacesPChar( P : PChar)
12959:  Func TrimEmbeddedZeros( const S :Str) :Str
12960:  Proc TrimEmbeddedZerosPChar( P : PChar)
12961:  Func TrimTrailPrimPChar( S : PChar) : PChar
12962:  Func TrimTrailPChar( Dest, S : PChar) : PChar
12963:  Func TrimTrailingZeros( const S :Str) :Str
12964:  Proc TrimTrailingZerosPChar( P : PChar)
12965:  Func UpCaseChar( C : Char) : Char
12966:  Func ovcCharInSet( C : Char; const CharSet : TOvcCharSet) :Bool
12967:  Func ovc32StringIsCurrentCodePage( const S : WideString) :Bool;
12968:  //Func ovc32StringIsCurrentCodePage1( const S:PWideChar; CP :Card) :Bool;
12969: end;
12970:
12971: Proc SIRegister_AfUtils(CL: TPSPascalCompiler);
12972: begin
12973:   //PRaiseFrame', '^TRaiseFrame // will not work
12974:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12975:    +ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12976:  Proc SafeCloseHandle( var Handle : THandle)
12977:  Proc ExchangeInt( X1, X2 : Int)
12978:  Proc FillInt( const Buffer, Size, Value : Int)
12979:  Func LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12980:  Func afCompareMem( P1, P2 : TObject; Length : Int) :Bool
12981:
12982: FILENAME_ADVAPI32      = 'ADVAPI32.DLL';
12983:     Func AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12984: Func AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12985:     Func AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12986:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12987:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12988:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12989:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12990:     Func AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12991:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
```

```
12992:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12993:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12994:       ObjectTypeListLength:DWORD;const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12995:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12996:     Func AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12997:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12998:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12999:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
13000:       ObjectTypeListLength: DWORD;const GenericMapping:TGenericMapping;  ObjectCreation: BOOL;
13001: var GrantedAccess:DWORD;var AccessStatusList:DWORD;var pfGenerateOnClose:BOOL):BOOL; stdcall;
13002:     Func BackupEventLog(hEventLog:THandle; lpBackupFileName: PKOLChar):BOOL; stdcall;
13003:     Func ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
13004:     Func CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
13005:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
13006:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
13007:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
13008: const lpStartupInfo:TStartupInfo;var lpProcessInformation: TProcessInformation):BOOL; stdcall;
13009:     Func GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
13010:     Func GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
13011: pSecurityDescriptor:PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded:DWORD):BOOL; stdcall;
13012:     Func GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
13013:     Func InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
13014:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
13015:     Func LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
13016:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
13017:     Func LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
13018:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
13019:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
13020:     Func LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
13021:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
13022:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
13023:     Func LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
13024:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
13025:     Func LookupPrivilegeName(lpSystemName: PKOLChar;
13026:       var lpLuid: TLargeInt; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
13027:     Func LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
13028:       var lpLuid: TLargeInt): BOOL; stdcall;
13029:     Func ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
13030:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
13031:     Func ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
13032:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
13033:     Func ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
13034:       ObjectTypeName:PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
13035:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
13036:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
13037:       var GenerateOnClose: BOOL): BOOL; stdcall;
13038:     Func ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
13039:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
13040:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
13041:     Func OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
13042:     Func OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
13043:     Func PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
13044:       ClientToken: THandle;var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
13045:     Func ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
13046:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
13047:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
13048:     Func RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
13049:       var phkResult: HKEY): Longint; stdcall;
13050:     Func RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
13051:       var phkResult: HKEY): Longint; stdcall;
13052:     Func RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
13053:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
13054:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
13055:       lpdwDisposition: PDWORD): Longint; stdcall;
13056:     Func RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
13057:     Func RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
13058:     Func RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
13059:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
13060:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
13061:     Func RegEnumKey(hKey:HKEY;dwIndex:DWORD;lpName:PKOLChar;cbName:DWORD):Longint;stdcall;
13062:     Func RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
13063:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
13064:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
13065:     Func RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
13066:     Func RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
13067:     Func RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
13068:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
13069:     Func RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
13070:       lpcbClass: PDWORD; lpReserved: Pointer;
13071:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
13072:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
13073:       lpftLastWriteTime: PFileTime): Longint; stdcall;
13074:     Func RegQueryMultipleValues(hKey: HKEY; var ValList;
13075:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
13076:     Func RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
13077:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
13078:     Func RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
13079:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
13080:     Func RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
```

```
13081:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
13082:     Func RegRestoreKey(hKey: HKEY; lpFile:PKOLChar; dwFlags: DWORD):Longint; stdcall;
13083:     Func RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
13084:       lpSecurityAttributes): PSecurityAttributes): Longint; stdcall;
13085:     Func RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
13086:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
13087:     Func RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
13088:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
13089:     Func RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
13090:     Func RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
13091:     Func ReportEvent(hEventLog: THandle; wType, wCategory: Word;
13092:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
13093:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
13094:     Func SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
13095:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
13096:  Func wAddAtom( lpString : PKOLChar) : ATOM
13097:  Func wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL): THandle
13098:  //Func wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
        lpOutBuffer:Pointer;nOutBufferSize:DWORD;var lpBytesRead:DWORD;nTimeOut: DWORD) : BOOL
13099:  //Func wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig): BOOL
13100:  Func
        wCompareString(Locale:LCID;dwCmpFlags:DWORD;lpString1:PKOLChar;cchCount1:Int;lpString2:PKOLChar;cchCount2:Int):Int
13101:  Func wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL):BOOL
13102:  //Func wCopyFileEx(lpExistingFileName,
        lpNewFileName:PKOLChar;lpProgressRoutine:TFNProgressRoutine;lpData:Pointer;pbCancel:PBool;
        dwCopyFlags:DWORD):BOOL
13103:  Func wCreateDirectory(lpPathName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
13104:  Func wCreateDirectoryEx(lpTemplateDirectory,lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
13105:  Func wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
        bInitialState:BOOL;lpName:PKOLChar):THandle;
13106:  Func wCreateFile(lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
        PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
13107:  Func wCreateFileMapping(hFile:THandle;lpFileMappingAttributes:PSecurityAttributes; flProtect,
        dwMaximumSizeHigh,dwMaximumSizeLow:DWORD;lpName:PKOLChar):THandle
13108:  Func wCreateHardLink(lpFileName,lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
13109:  Func
        CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
13110:  Func wCreateNamedPipe(lpName:PKOLChar;dwOpenMode,dwPipeMode,nMaxInstances,nOutBufferSize,nInBufferSize,
        nDefaultTimeOut:DWORD;lpSecurityAttributes:PSecurityAttributes): THandle
13111:  //Func CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
        lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
        Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
        lpProcessInfo:TProcessInformation):BOOL
13112:  Func wCreateSemaphore(lpSemaphoreAttributes:PSecurityAttributes;lInitialCount,
        lMaximumCount:Longint;lpName:PKOLChar):THandle
13113:  Func
        wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
13114:  Func wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
13115:  Func wDeleteFile( lpFileName : PKOLChar) : BOOL
13116:  Func wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
13117:  //Func
        wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
13118:  //Func wEnumDateFormats(lpDateFmtEnumProc:TFNDateFmtEnumProc;Locale:LCID;dwFlags:DWORD): BOOL
13119:  //Func wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
13120:  //Func wEnumResourceTypes(hModule:HMODULE;lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
13121:  //Func wEnumSystemCodePages(lpCodePageEnumProc:TFNCodepageEnumProc;dwFlags:DWORD): BOOL
13122:  //Func wEnumSystemLocales(lpLocaleEnumProc:TFNLocaleEnumProc;dwFlags:DWORD): BOOL
13123:  //Func wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
13124:  Func wExpandEnvironmentStrings(lpSrc:PKOLChar;lpDst:PKOLChar;nSize:DWORD): DWORD
13125:  Proc wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
13126:  //Func wFillConsoleOutputCharacter(hConsoleOutput:THandle;cCharacter:KOLChar;nLength:DWORD; dwWriteCoord:
        TCoord;var lpNumberOfCharsWritten:DWORD):BOOL
13127:  Func wFindAtom( lpString : PKOLChar) : ATOM
13128:  Func wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
13129:  Func wFindFirstFile(lpFileName:PKOLChar;var lpFindFileData:TWIN32FindData): THandle
13130:  //Func wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
13131:  Func wFindNextFile(hFindFile:THandle;var lpFindFileData:TWIN32FindData):BOOL
13132:  Func wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
13133:  Func wFindResourceEx(hModule:HMODULE;lpType,lpName:PKOLChar;wLanguage:Word): HRSRC
13134:  Func wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
13135:  //Func wFormatMessage(dwFlags:DWORD;lpSource:Pointer;dwMessageId:DWORD;dwLanguageId:DWORD;
        lpBuffer:PKOLChar;nSize:DWORD;Arguments:Pointer):DWORD
13136:  Func wFreeEnvironmentStrings( EnvBlock : PKOLChar): BOOL
13137:  Func wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Int) : UINT
13138:  Func wGetBinaryType(lpApplicationName:PKOLChar;var lpBinaryType:DWORD):BOOL
13139:  Func wGetCommandLine: PKOLChar
13140:  //Func wGetCompressedFileSize(lpFileName:PKOLChar;lpFileSizeHigh:PDWORD): DWORD
13141:  Func wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
13142:  Func wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
13143:  //Func wGetCurrencyFormat(Locale:LCID;dwFlags:DWORD;lpValue:PKOLChar;lpFormat :
        PCurrencyFmt;lpCurrencyStr:PKOLChar;cchCurrency:Int):Int
13144:  Func wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
13145:  //Func wGetDateFormat(Locale:LCID; dwFlags:DWORD;lpDate:PSystemTime;lpFormat: PKOLChar;
        lpDateStr:PKOLChar; cchDate: Int):Int
13146:  //Func wGetDefaultCommConfig(lpszName:PKOLChar;var lpCC:TCommConfig;var lpdwSize:DWORD):BOOL
13147:  Func wGetDiskFreeSpace(lpRootPathName:PKOLChar;var lpSectorsPerCluster,lpBytesPerSector,
        lpNumberOfFreeClusters,lpTotalNumberOfClusters:DWORD):BOOL
13148:  //Func wGetDiskFreeSpaceEx(lpDirectoryName:PKOLChar;var lpFreeBytesAvailableToCaller,lpTotalNumberOfBytes,
        lpTotalNumberOfFreeBytes:PLargeInt):BOOL
```

```
13149:  Func wGetDriveType( lpRootPathName : PKOLChar) : UINT
13150:  Func wGetEnvironmentStrings : PKOLChar
13151:  Func wGetEnvironmentVariable(lpName:PKOLChar;lpBuffer:PKOLChar;nSize:DWORD): DWORD;
13152:  Func wGetFileAttributes( lpFileName : PKOLChar) : DWORD
13153:  //Func
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
13154:  Func wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
13155:  //Func wGetLocaleInfo(Locale:LCID;LCType:LCTYPE;lpLCData:PKOLChar;cchData:Int):Int
13156:  Func wGetLogicalDriveStrings(nBufferLength:DWORD;lpBuffer:PKOLChar):DWORD
13157:  Func wGetModuleFileName(hModule:HINST;lpFilename:PKOLChar;nSize:DWORD):DWORD
13158:  Func wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
13159:  //Func wGetNamedPipeHandleState(hNamedPipe: THandle;lpState,lpCurInstances,lpMaxCollectionCount,
        lpCollectDataTimeout:PDWORD;lpUserName:PKOLChar;nMaxUserNameSize:DWORD): BOOL
13160: //Func
        wGetNumberFormat(Locale:CID;dwFlags:DWORD;lpValue:PKOLChar;lpFormat:PNumberFmt;lpNumberStr:PKOLChar;cchNumb
13161:  Func wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Int;lpFileName:PKOLChar):UINT;
13162:  Func
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
13163:  Func wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
13164:  Func wGetPrivateProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD
13165:  Func wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Int) : UINT
13166:  Func wGetProfileSection(lpAppName:PKOLChar;lpReturnedString:PKOLChar;nSize:DWORD):DWORD
13167:  Func wGetProfileString(lpAppName,lpKeyName,lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
13168:  Func wGetShortPathName(lpszLongPath:PKOLChar;lpszShortPath:PKOLChar;cchBuffer:DWORD):DWORD
13169:  //Proc wGetStartupInfo( var lpStartupInfo : TStartupInfo)
13170: //Func wGetStringTypeEx(Locale:LCID;dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;var lpCharType):BOOL
13171:  Func wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
13172:  Func wGetTempFileName(lpPathName,lpPrefixString:PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
13173:  Func wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
13174: //Func
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
13175: //Func wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13176: //Func GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize :
        DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength,lpFileSystemFlags:DWORD;
        lpFileSystemNameBuffer:PKOLChar;nFileSystemNameSize: DWORD) : BOOL
13177:  Func wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
13178:  Func wGlobalAddAtom( lpString : PKOLChar) : ATOM
13179:  Func wGlobalFindAtom( lpString : PKOLChar) : ATOM
13180:  Func wGlobalGetAtomName(nAtom:ATOM;lpBuffer:PKOLChar;nSize:Int):UINT
13181:  Func wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
13182:  Func
        wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
13183:  Func wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
13184:  Func wLoadLibraryEx(lpLibFileName:PKOLChar;hFile:THandle;dwFlags:DWORD): HMODULE
13185:  Func wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
13186:  Func wMoveFileEx(lpExistingFileName,lpNewFileName:PKOLChar;dwFlags:DWORD): BOOL
13187:  //Func wMoveFileWithProgress(lpExistingFileName,lpNewFileName:PKOLChar;lpProgressRoutine:
        TFNProgressRoutine;lpData:Pointer;dwFlags:DWORD): BOOL
13188:  Func wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
13189:  Func wOpenFileMapping(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpName: PKOLChar):THandle
13190:  Func wOpenMutex(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpName:PKOLChar): THandle
13191:  Func wOpenSemaphore( dwDesiredAccess:DWORD; bInheritHandle:BOOL;lpName:PKOLChar):THandle
13192:  Func wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
13193:  Proc wOutputDebugString( lpOutputString : PKOLChar)
13194: //Func wPeekConsoleInput(hConsoleInput:THandle;varlpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsRead:DWORD):BOOL;
13195:  Func wQueryDosDevice(lpDeviceName:PKOLChar;lpTargetPath:PKOLChar;ucchMax:DWORD):DWORD
13196:  //Func wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
13197:  //Func wReadConsole(hConsoleInput:THandle;lpBuffer:Pointer;nNumberOfCharsToRead:DWORD; var
        lpNumberOfCharsRead:DWORD;lpReserved Pointer):BOOL
13198:  //Func wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
        lpNumbOfEventsRead:DWORD):BOOL;
13199:  //Func wReadConsoleOutput(hConsoleOutput:THandle;lpBuffer:Pointer;dwBufferSize,dwBufferCoord:TCoord;var
        lpReadRegion:TSmallRect) : BOOL
13200:  //Func wReadConsoleOutputCharacter(hConsoleOutput:THandle;lpCharacter:PKOLChar;nLength :
        DWORD;dwReadCoord:TCoord;var lpNumberOfCharsRead:DWORD): BOOL
13201:  Func wRemoveDirectory( lpPathName : PKOLChar) : BOOL
13202:  //Func wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
        lpClipRectangle:PSmallRect;dwDestinationOrigin:TCoord;var lpFill:TCharInfo):BOOL
13203:  Func wSearchPath(lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
13204:  Func wSetComputerName( lpComputerName: PKOLChar): BOOL
13205:  Func wSetConsoleTitle( lpConsoleTitle: PKOLChar): BOOL
13206:  Func wSetCurrentDirectory( lpPathName: PKOLChar): BOOL
13207:  //Func wSetDefaultCommConfig(lpszName: PKOLChar; lpCC:PCommConfig; dwSize: DWORD) : BOOL
13208:  Func wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
13209:  Func wSetFileAttributes(lpFileName:PKOLChar;dwFileAttributes:DWORD): BOOL
13210:  //Func wSetLocaleInfo(Locale:LCID;LCType  LCTYPE;lpLCData:PKOLChar): BOOL
13211:  Func wSetVolumeLabel(lpRootPathName:PKOLChar;lpVolumeName:PKOLChar): BOOL
13212: //Func wUpdateResource(hUpdate:THandle;lpType,lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
13213:  Func wVerLanguageName(wLang:DWORD;szLang:PKOLChar;nSize:DWORD):DWORD
13214:  Func wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
13215:  //Func wWriteConsole(hConsoleOutput:THandle;const lpBuffer:Pointer;nNumberOfCharsToWrite:DWORD;var
        lpNumberOfCharsWritten:DWORD;lpReserved:Pointer):BOOL
13216:  //Func wWriteConsoleInput(hConsoleInput:THandle;const lpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsWritten:DWORD):BOOL
13217:  //Func wWriteConsoleOutput(hConsoleOutput:THandle;lpBuffer:Pointer;dwBufferSize,dwBufferCoord:TCoord;var
        lpWriteRegion:TSmallRect):BOOL
```

```
13218:  //Func
        wWriteConsoleOutputCharacter(hConsoleOutput:THandle;lpCharacter:PKOLChar;nLength:DWORD;dwWriteCoord:TCoord;
        lpNumberOfCharsWritten:DWORD):BOOL
13219:  Func wWritePrivateProfileSection(lpAppName,lpString,lpFileName:PKOLChar):BOOL
13220:  Func wWritePrivateProfileString(lpAppName,lpKeyName,lpString,lpFileName:PKOLChar):BOOL
13221:  Func wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
13222:  Func wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
13223:  Func wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
13224:  Func wlstrcmp( lpString1, lpString2 : PKOLChar) : Int
13225:  Func wlstrcmpi( lpString1, lpString2 : PKOLChar) : Int
13226:  Func wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
13227:  Func wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Int) : PKOLChar
13228:  Func wlstrlen( lpString : PKOLChar) : Int
13229:  //Func
        wMultinetGetConnectionPerformance(lpNetResource:PNetResource;lpNetConnectInfoStruc:PNetConnectInfoStruct):D
13230:  //Func wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
13231:  //Func wWNetAddConnection3(hwndOwner:HWND;var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD
13232:  Func wWNetAddConnection(lpRemoteName,lpPassword,lpLocalName:PKOLChar):DWORD
13233:  Func wWNetCancelConnection2(lpName:PKOLChar;dwFlags:DWORD;fForce:BOOL):DWORD
13234:  Func wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
13235:  //Func wWNetConnectionDialog1(var lpConnDlgStruct:TConnectDlgStruct): DWORD
13236:  //Func wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct): DWORD
13237:  //Func wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
13238:  Func wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar;var lpnLength:DWORD):DWORD;
13239:  Func wWNetGetLastError(var lpError:DWORD;lpErrorBuf:PKOLChar;nErrorBufSize: DWORD;
        lpNameBuf:PKOLChar;nNameBufSize:DWORD):DWORD
13240:  //Func wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
13241:  Func wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
13242:  //Func wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
13243: //Func wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
13244:  Func wWNetGetUser(lpName:PKOLChar;lpUserName PKOLChar;var lpnLength:DWORD): DWORD
13245: //Func wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
13246: //Func wWNetSetConnection(lpName:PKOLChar;dwProperties:DWORD; pvValues : Pointer): DWORD
13247: //Func wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
13248:  Func wGetFileVersionInfo(lptstrFilenam:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
13249:  Func wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
13250:  Func wVerFindFile(uFlags:DWORD;szFileName,szWinDir,szAppDir,szCurDir:PKOLChar;var
        lpuCurDirLen:UINT;szDestDir:PKOLChar;var lpuDestDirLen:UINT):DWORD
13251:  Func wVerInstallFile(uFlags:DWORD;szSrcFileName,szDestFileName,szSrcDir,szDestDir,szCurDir,
        szTmpFile:PKOLChar;var lpuTmpFileLen:UINT):DWORD
13252: //Func wVerQueryValue(pBlock:Pter;lpSubBlock:PKOLChar;varlplpBuffer:Ptr;varpuLen:UINT):BOOL;
13253: //Func wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
13254: //Func wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
13255:  Func wAddFontResource( FileName : PKOLChar) : Int
13256:  //Func wAddFontResourceEx(p1:PKOLChar;p2  DWORD;p3:PDesignVector): Int
13257:  Func wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
13258:  Func wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
13259: //Func wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
13260: //Func wCreateDC(lpszDriver,lpszDevice,lpszOutput:PKOLChar; lpdvmInit: PDeviceMode) : HDC
13261: //Func wCreateEnhMetaFile(DC:HDC;FileName:PKOLChar;Rect:PRect;Desc: PKOLChar) : HDC
13262:  Func wCreateFont( nHeight,nWidth,nEscapement,nOrientaion,fnWeight:Int;fdwItalic, fdwUnderline,
        fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
13263:  Func wCreateFontIndirect( const p1 : TLogFont) : HFONT
13264:  //Func wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
13265: // Func wCreateIC(lpszDriver,lpszDevice, lpszOutput:PKOLChar; lpdvmInit:PDeviceMode): HDC
13266:  Func wCreateMetaFile( p1 : PKOLChar) : HDC
13267:  Func wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
13268: //Func wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
13269: // Func wEnumFontFamilies(DC:HDC; p2: PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
13270:  //Func wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
13271:  //Func wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Int;
13272:  //Func wEnumICMProfiles(DC:HDC; ICMProc:TFNICMEnumProc; p3  LPARAM) : Int
13273:  //Func wExtTextOut(DC:HDC;X,Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInt:BOOL
13274:  //Func wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs): BOOL
13275:  //Func wGetCharABCWidthsFloat(DC:HDC; FirstChar,LastChar: UINT;const ABCFloatSturcts): BOOL
13276:  //Func wGetCharWidth32(DC:HDC;FirstChar,LastChar:UINT; const Widths): BOOL
13277:  //Func wGetCharWidth(DC:HDC;FirstChar,LastChar:UINT; const Widths): BOOL
13278: // Func wGetCharWidthFloat(DC:HDC;FirstChar,LastChar  UINT;const Widths):BOOL
13279: // Func wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
13280:  Func wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
13281:  Func wGetEnhMetaFileDescription(p1 HENHMETAFILE;p2:UINT;p3:PKOLChar):UINT
13282: // Func wGetGlyphIndices(DC:HDC;p2:PKOLChar;p3:Int;p4:PWORD;p5:DWORD): DWORD
13283: //Func wGetGlyphOutline(DC:HDC;uChar,uFormat:UINT;const
        lpgm:TGlyphMetrics;cbBuffer:DWORD;lpvBuffer:Pointer;const lpmat2:TMat2): DWORD
13284:  Func wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
13285: // Func wGetLogColorSpace(p1:HCOLORSPACE;var ColorSpace:TLogColorSpace;Size:DWORD):BOOL
13286:  Func wGetMetaFile( p1 : PKOLChar) : HMETAFILE
13287: // Func wGetObject( p1 : HGDIOBJ; p2 : Int; p3 : Pointer) : Int
13288:  //Func wGetOutlineTextMetrics(DC:HDC;p2:UINT;OTMetricStructs:Pointer): UINT
13289:  //Func wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Int;p5,p6:PInt;var p7:TSize):BOOL
```

```
13290:  Func wGetTextExtentPoint32(DC:HDC;Str:PKOLChar;Count:Int;var Size:TSize):BOOL
13291:  Func wGetTextExtentPoint(DC:HDC;Str:PKOLChar;Count:Int;var Size:TSize):BOOL
13292:  Func wGetTextFace( DC : HDC; Count : Int; Buffer : PKOLChar) : Int
13293:  //Func wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
13294:  Func wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Int) : BOOL
13295:  Func wRemoveFontResource( FileName : PKOLChar) : BOOL
13296:  //Func wRemoveFontResourceEx(p1:PKOLChar;p2 DWORD;p3: PDesignVector): BOOL
13297:  //Func wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
13298:  Func wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
13299:  //Func wStartDoc( DC : HDC; const p2 : TDocInfo) : Int
13300:  Func wTextOut( DC : HDC; X, Y : Int; Str : PKOLChar; Count : Int) : BOOL
13301:  Func wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
13302:  Func wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
13303:  //Func wwglUseFontOutlines(p1:HDC;p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
13304:  Func wAppendMenu(hMenu:HMENU;uFlags,uIDNewItem:UINT;lpNewItem:PKOLChar):BOOL
13305:  Func wCallMsgFilter( var lpMsg : TMsg; nCode : Int) : BOOL
13306:  //Func wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
13307:  //Func wChangeDisplaySettings(var lpDevMode:TDeviceMode; dwFlags : DWORD) : Longint
13308: // Func wChangeDisplaySettingsEx(lpszDeviceName:PKOLChar;var lpDevMode:
       TDeviceMode;wnd:HWND;dwFlags:DWORD;lParam:Pointer):Longint
13309:  Func wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
13310:  Func wCharLower( lpsz : PKOLChar) : PKOLChar
13311:  Func wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
13312:  Func wCharNext( lpsz : PKOLChar) : PKOLChar
13313:  //Func wCharNextEx(CodePage:Word;lpCurrentChar:LPCSTR;dwFlags:DWORD): LPSTR
13314:  Func wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
13315: // Func wCharPrevEx(CodePage:Word;lpStart,lpCurrentChar:LPCSTR; dwFlags:DWORD) : LPSTR
13316:  Func wCharToOem(lpszSrc:PKOLChar; lpszDst : PKOLChar) : BOOL
13317:  Func wCharToOemBuff(lpszSrc:PKOLChar;lpszDst:PKOLChar;cchDstLength:DWORD): BOOL
13318:  Func wCharUpper( lpsz : PKOLChar) : PKOLChar
13319:  Func wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
13320:  Func wCopyAcceleratorTable(hAccelSrc:HACCEL;var lpAccelDst,cAccelEntries:Int) :Int
13321:  Func wCreateAcceleratorTable( var Accel, Count : Int) : HACCEL
13322:  //Func wCreateDesktop(lpszDesktop,
       lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
13323:  //Func wCreateDialogIndirectParam(hInstance:HINST;const lpTemplate:TDlgTemplate;hWndParent
       :HWND;lpDialogFunc:TFNDlgProc;dwInitParam:LPARAM):HWND
13324:  //Func
       wCreateDialogParam(hInstance:HINST;lpTemplateName:PKOLChar;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitPa
13325:  Func wCreateMDIWindow(lpClassName,lpWindowName:PKOLChar;dwStyle:DWORD;X,Y,nWidth,
       nHeight:Int;hWndParent:HWND;hInstance:HINST;lParam:LPARAM):HWND
13326: //Func wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,nWidth,
        nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
13327:  //Func wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
       dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
13328:  Func wDefDlgProc(hDlg:HWND;Msg:UINT;wParam WPARAM;lParam:LPARAM):LRESULT
13329:  Func wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
13330:  Func wDefMDIChildProc(hWnd:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
13331:  Func wDefWindowProc(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM): RESULT
13332:  //Func wDialogBoxIndirectParam(hInstance:HINST;const
       lpDialogTemplate:TDlgTemplate;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitParam:LPARAM):Int
13333:  //Func
       wDialogBoxParam(hInstance:HINST;lpTemplateName:PKOLChar;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitParam
       LPARAM): Int
13334:  Func wDispatchMessage(const lpMsg : TMsg) : Longint
13335:  Func wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,nIDStaticPath:Int;uFileType:UINT):Int;
13336:  Func wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,nIDStaticPath:Int;uFiletype:UINT):Int;
13337:  Func wDlgDirSelectComboBoxEx(hDlg:HWND;lpString:PKOLChar;nCount,nIDComboBox:Int): BOOL
13338:  Func wDlgDirSelectEx(hDlg:HWND;lpString:PKOLChar;nCount,nIDListBox:Int):BOOL
13339: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
       cy:Int;Flags:UINT):BOOL;
13340:  Func wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Int;var lpRect:TRect;uFormat:UINT):Int;
13341:  Func wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
13342:  Func wFindWindowEx(Parent,Child:HWND;ClassName,WindowName:PKOLChar): HWND
13343:  //Func wGetAltTabInfo(hwnd:HWND;iItem:Int;var
       pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
13344: // Func wGetClassInfo(hInstance:HINST;lpClassName:PKOLChar;var lpWndClass:TWndClass):BOOL
13345:  //Func wGetClassInfoEx(Instance:HINST;Classname:PKOLChar;var WndClass:TWndClassEx):BOOL
13346:  Func wGetClassLong( hWnd : HWND; nIndex : Int) : DWORD
13347:  Func wGetClassName(hWnd:HWND; lpClassName:PKOLChar;nMaxCount:Int) : Int
13348:  Func wGetClipboardFormatName(format:UINT;lpszFormatName:PKOLChar;cchMaxCount:Int):Int;
13349:  Func wGetDlgItemText(hDlg:HWND;nIDDlgItem:Int;lpString:PKOLChar;nMaxCount:Int):UINT
13350:  Func wGetKeyNameText(lParam:Longint;lpString:PKOLChar;nSize:Int):Int
13351:  Func wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
13352:  //Func wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
13353:  Func wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Int;uFlag:UINT):Int;
13354:  Func wGetMessage(var lpMsg:TMsg;hWnd:HWND;wMsgFilterMin wMsgFilterMax:UINT):BOOL
13355:  Func wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
13356: //Func wGetTabbedTextExtent(hDC:HDC;lpString:PKOLChar;nCount,nTabPositions:Int;var
       lpnTabStopPositions):DWORD
13357: //Func wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var lpnLengthNeed:DWORD)BOOL;
13358:  Func wGetWindowLong( hWnd : HWND; nIndex : Int) : Longint
13359:  Func wGetWindowModuleFileName(hwnd:HWND;pszFileName:PKOLChar;cchFileNameMax:UINT): UINT
13360:  Func wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Int) : Int
13361:  Func wGetWindowTextLength( hWnd : HWND) : Int
13362: //Func wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
       nHeigt:Int):BOOL;
13363:  Func wInsertMenu(hMenu:HMENU;uPosition,uFlags,uIDNewItem:UINT; lpNewItem: PKOLChar): BOOL
13364:  //Func wInsertMenuItem(p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
```

```
13365:  Func wIsCharAlpha( ch : KOLChar) : BOOL
13366:  Func wIsCharAlphaNumeric( ch : KOLChar) : BOOL
13367:  Func wIsCharLower( ch : KOLChar) : BOOL
13368:  Func wIsCharUpper( ch : KOLChar) : BOOL
13369:  Func wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
13370:  Func wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
13371:  Func wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
13372:  Func wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
13373:  Func wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
13374:  Func wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
13375:  Func wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Int;Flags:UINT): THandle
13376:  Func wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
13377:  Func wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
13378:  //Func wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
13379:  Func wLoadString(hInstance:HINST;uID:UINT;lpBuffer:PKOLChar;nBufferMax:Int):Int
13380:  Func wMapVirtualKey( uCode, uMapType : UINT) : UINT
13381:  Func wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
13382:  Func wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Int
13383:  Func wMessageBoxEx(hWnd:HWND;lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word)Int
13384:  //Func wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
13385:  Func wModifyMenu(hMnu: HMENU; uPosition,uFlags,uIDNewItem:UINT; lpNewItem:PKOLChar): BOOL
13386:  //Func wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
13387:  //7Func wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
13388:  //Func wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
13389:  Func wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
13390:  Func wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
13391:  Func wOpenWindowStation(lpszWinSta:PKOLChar;fInherit:BOOL;dwDesiredAccess:DWORD):HWINSTA
13392:  Func wPeekMessage(var lpMsg:TMsg;hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
13393:  Func wPostMessage(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):BOOL
13394:  Func wPostThreadMessage(idThread:DWORD;Msg:UINT;wParam:WPARAM;lParam:LPARAM): BOOL
13395:  Func wRealGetWindowClass(hwnd:HWND;pszType PKOLChar;cchType:UINT):UINT
13396: // Func wRegisterClass( const lpWndClass : TWndClass) : ATOM
13397: // Func wRegisterClassEx( const WndClass : TWndClassEx): ATOM
13398:  Func wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
13399: //Func wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13400:  Func wRegisterWindowMessage( lpString : PKOLChar) : UINT
13401:  Func wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
13402:  Func wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13403:  Func wSendMessage(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM): LRESULT
13404:  //Func wSendMessageCallback(hWnd: HWND; Msg:UINT;wParam:
       WPARAM;lParam:LPARAM;lpResultCallBack:TFNSendAsyncProc;dwData DWORD):BOOL
13405:  Func wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM;fuFlags,uTimeout:UINT;var
       lpdwResult:DWORD):LRESULT
13406:  Func wSendNotifyMessage(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):BOOL
13407:  Func wSetClassLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint) : DWORD
13408:  Func wSetDlgItemText(hDlg: HWND;nIDDlgItem:Int;lpString : PKOLChar) : BOOL
13409:  //Func wSetMenuItemInfo(p1:HMENU;p2:UINT;p3:BOOL;const p4:TMenuItemInfo):BOOL
13410:  Func wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13411: // Func wSetUserObjectInformation(hObj:THandle;nIndex:Int;pvInfo:Pointer;nLength:DWORD):BOOL
13412:  Func wSetWindowLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint) : Longint
13413:  Func wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13414:  //Func wSetWindowsHook(nFilterType:Int; pfnFilterProc:TFNHookProc): HHOOK
13415:  //Func wSetWindowsHookEx(idHook:Int;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13416: // Func wSystemParametersInfo(uiAction,uiParam:UINT; pvParam:Pointer; fWinIni: UINT):BOOL
13417:  Func wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var lpnTabStopPositions,
       nTabOrigin:Int):Longint;
13418:  Func wTranslateAccelerator(hWnd:HWND;hAccTable:HACCEL;var lpMsg:TMsg): Int
13419:  Func wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13420:  Func wVkKeyScan( ch : KOLChar) : SHORT
13421:  Func wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13422:  Func wWinHelp(hWndMain:HWND;lpszHelp:PKOLChar;uCommand:UINT;dwData:DWORD):BOOL
13423:  Func wwsprintf( Output : PKOLChar; Format : PKOLChar) : Int
13424:  Func wwvsprintf( Output : PKOLChar; Format: PKOLChar; arglist:va_list): Int
13425:
13426: //TestDrive!
13427: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13428:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13429:  Func GetDomainUserSidS(const domainName:Str;const userName:Str;var foundDomain:Str):str;
13430:  Func GetLocalUserSidStr( const UserName :Str) :Str
13431:  Func getPid4user(const domain:str;const user:str;var pid:dword):boolean
13432:  Func Impersonate2User( const domain :Str; const user :Str):Bool
13433:  Func GetProcessUserBypid(pid:DWORD;var UserName,Domain:Ansistr):Bool
13434:  Func KillProcessbyname( const exename :Str; var found : Int) : Int
13435:  Func getWinProcessList : TStringList
13436:  Func WaitTilClose(hWnd: Int): Int;
13437:  Func DoUserMsgs:Bool;
13438:  Func MsgFunc(hWnd,Msg,wParam,lParam:Int):Int; stdcall;
13439:  Proc ShowMsg(hParent:Int; const Mess, Title:Str);  //modal but NOT blockable
13440:  Proc DeleteMsgForm(Handle: Int);
13441:  Proc DisableForms;
13442:  Func FoundTopLevel(hWnd, LParam: Int): BOOL; StdCall;
13443: end;
13444:
13445: Proc SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13446: begin
13447:  'AfMaxSyncSlots','LongInt'( 64);
13448:  'AfSynchronizeTimeout','LongInt'( 2000);
13449:   TAfSyncSlotID', 'DWORD
13450:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
```

```
13451:   TAfSafeSyncEvent', 'Proc ( ID : TAfSyncSlotID)
13452:   TAfSafeDirectSyncEvent', 'Procedure
13453:  Func AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13454:  Func AfReleaseSyncSlot( const ID : TAfSyncSlotID) :Bool
13455:  Func AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable :Bool) :Bool
13456:  Func AfValidateSyncSlot( const ID : TAfSyncSlotID) :Bool
13457:  Func AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) :Bool
13458:  Func AfDirectSyncEvent(Event:TAfSafeDirectSyncEvent;Timeout:DWORD):Boolean
13459:  Func AfIsSyncMethod :Bool
13460:  Func AfSyncWnd : HWnd
13461:  Func AfSyncStatistics : TAfSyncStatistics
13462:  Proc AfClearSyncStatistics
13463: end;
13464:
13465: Proc SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13466: begin
13467:  'fBinary','LongWord')( $00000001);
13468:  'fParity','LongWord'( $00000002);
13469:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13470:  'fOutxDsrFlow','LongWord')( $00000008);
13471:  'fDtrControl','LongWord')( $00000030);
13472:  'fDtrControlDisable','LongWord')( $00000000);
13473:  'fDtrControlEnable','LongWord')( $00000010);
13474:  'fDtrControlHandshake','LongWord')( $00000020);
13475:  'fDsrSensitivity','LongWord')( $00000040);
13476:  'fTXContinueOnXoff','LongWord')( $00000080);
13477:  'fOutX','LongWord')( $00000100);
13478:  'fInX','LongWord')( $00000200);
13479:  'fErrorChar','LongWord')( $00000400);
13480:  'fNull','LongWord')( $00000800);
13481:  'fRtsControl','LongWord')( $00003000);
13482:  'fRtsControlDisable','LongWord')( $00000000);
13483:  'fRtsControlEnable','LongWord')( $00001000);
13484:  'fRtsControlHandshake','LongWord')( $00002000);
13485:  'fRtsControlToggle','LongWord')( $00003000);
13486:  'fAbortOnError','LongWord')( $00004000);
13487:  'fDummy2','LongWord')( $FFFF8000);
13488:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13489:   FindClass('TOBJECT'),'EAfComPortCoreError
13490:   FindClass('TOBJECT'),'TAfComPortCore
13491:   TAfComPortCoreEvent', 'Proc ( Sender : TAfComPortCore; Even'
13492:    +'tKind : TAfCoreEvent; Data : DWORD)
13493:   SIRegister_TAfComPortCoreThread(CL);
13494:   SIRegister_TAfComPortEventThread(CL);
13495:   SIRegister_TAfComPortWriteThread(CL);
13496:   SIRegister_TAfComPortCore(CL);
13497:  Func FormatDeviceName( PortNumber : Int) :Str
13498: end;
13499:
13500: Proc SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13501: begin
13502:   TAFIOFileStreamEvent', 'Func ( const fileName :Str; mode: Word) : TStream
13503:   TAFIOFileStreamExistsEvent', 'Func ( const fileName :Str) :Bool
13504:   SIRegister_TApplicationFileIO(CL);
13505:   TDataFileCapability', '( dfcRead, dfcWrite )
13506:   TDataFileCapabilities', 'set of TDataFileCapability
13507:   SIRegister_TDataFile(CL);
13508:   //TDataFileClass', 'class of TDataFile
13509:  Func ApplicationFileIODefined :Bool
13510:  Func CreateFileStream(const fileName:str;mode:WordfmShareDenyNone):TStream
13511:  Func FileStreamExists(const fileName:Str):Bool
13512:  //Proc Register
13513: end;
13514:
13515: Proc SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13516: begin
13517:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13518:    +', uftCstring, uftSmallint, uftInt, uftQuad, uftFloat, uftDoublePrecisi'
13519:    +'on,uftTimestamp,uftBlob,uftBlobId,uftDate,uftTime,uftInt64,uftArray,uftNull)
13520:   TALFBXScale', 'Int
13521:   FindClass('TOBJECT'),'EALFBXConvertError
13522:   SIRegister_EALFBXError(CL);
13523:   SIRegister_EALFBXException(CL);
13524:   FindClass('TOBJECT'),'EALFBXGFixError
13525:   FindClass('TOBJECT'),'EALFBXDSQLError
13526:   FindClass('TOBJECT'),'EALFBXDynError
13527:   FindClass('TOBJECT'),'EALFBXGBakError
13528:   FindClass('TOBJECT'),'EALFBXGSecError
13529:   FindClass('TOBJECT'),'EALFBXLicenseError
13530:   FindClass('TOBJECT'),'EALFBXGStatError
13531:   //EALFBXExceptionClass', 'class of EALFBXError
13532:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13533:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13534:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13535:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13536:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13537:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13538:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13539:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
```

```
13540:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13541:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13542:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13543:    +'Commit,tpRecVersion,tpNoRecVersion,tpRestartRequests,tpNoAutoUndo, tpLockTimeout)
13544:   TALFBXTransParams', 'set of TALFBXTransParam
13545:  Func ALFBXStrToCharacterSet( const CharacterSet : Ansistr) : TALFBXCharacterSet
13546:  Func ALFBXCreateDBParams( Params : Ansistr; Delimiter : Char) : Ansistr
13547:  Func ALFBXCreateBlobParams( Params : Ansistr; Delimiter : Char) : Ansistr
13548:  'cALFBXMaxParamLength','LongInt'( 125);
13549:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13550:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13551:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13552:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13553:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13554:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13555:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13556:   SIRegister_TALFBXSQLDA(CL);
13557:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13558:   SIRegister_TALFBXPoolStream(CL);
13559:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13560:   TALFBXBlobData', 'record Size : Int; Buffer :Str; end
13561:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13562:   //TALFBXArrayDesc', 'TISCArrayDesc
13563:   //TALFBXBlobDesc', 'TISCBlobDesc
13564:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13565:   //TALFBXArrayInfo', 'record index : Int; size:Int;info: TALFBXArrayDesc; end
13566:   SIRegister_TALFBXSQLResult(CL);
13567:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13568:   SIRegister_TALFBXSQLParams(CL);
13569:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13570:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13571:    +'atementType : TALFBXStatementType; end
13572:   FindClass('TOBJECT'),'TALFBXLibrary
13573:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13574:   TALFBXOnConnectionLost', 'Proc ( Lib : TALFBXLibrary)
13575:   //TALFBXOnGetDBExceptionClass', 'Proc ( Number : Int; out'
13576:    //+' Excep : EALFBXExceptionClass)
13577:   SIRegister_TALFBXLibrary(CL);
13578:  'cAlFBXDateOffset','LongInt'( 15018);
13579:  'cALFBXTimeCoeff','LongInt'( 864000000);
13580:  //Proc ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13581:  //Proc ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13582:  //Func ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13583:  Proc ALFBXDecodeSQLDate( v : Int; out Year : SmallInt; out Month, Day : Word)
13584:  Proc ALFBXDecodeSQLTime(v:Card;out Hour,Minute,Second:Word;out Fractions: LongWord)
13585:  //Proc ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13586:  //Proc ALFBXEncodeTimeStamp1( const Date : Int; v : PISCTimeStamp);
13587:  //Proc ALFBXEncodeTimeStamp2( const Time :Card; v : PISCTimeStamp);
13588:  Func ALFBXEncodeSQLDate(Year : Int; Month, Day : Int) : Int
13589:  Func ALFBXEncodeSQLTime(Hour,Minute,Second:Word;var Fractions:LongWord):Card
13590:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13591:   TALFBXDPBInfo', 'record Name : Ansistr; ParamType : TALFBXParamType; end
13592:  Func ALFBXSQLQuote( const name : Ansistr) : Ansistr
13593:  Func ALFBXSQLUnQuote( const name : Ansistr) : Ansistr
13594: end;
13595:
13596: Proc SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13597: begin
13598:   TALFBXClientSQLParam', 'record Value : Ansistr; IsNull :Bool; end
13599:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13600:   TALFBXClientSelectDataSQL', 'record SQL : Ansistr; Params : T'
13601:    +ALFBXClientSQLParams; RowTag : Ansistr; ViewTag : Ansistr; Skip : in'
13602:    +teger; First : Int; CacheThreshold : Int; end
13603:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13604:   TALFBXClientUpdateDataSQL','record SQL : Ansistr; Params: TALFBXClientSQLParams; end
13605:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13606:   TALFBXClientMonitoringIOStats','record
       page_reads:int64;page_writes:int64;page_fetches:int64;page_marks:int64;end
13607:   SIRegister_TALFBXClient(CL);
13608:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13609:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13610:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13611:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13612:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13613:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13614:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13615:   SIRegister_TALFBXConnectionPoolClient(CL);
13616:   SIRegister_TALFBXEventThread(CL);
13617:   Func AlMySqlClientSlashedStr( const Str : Ansistr) : Ansistr
13618: end;
13619:
13620: Proc SIRegister_ovcBidi(CL: TPSPascalCompiler);
13621: begin
13622: _OSVERSIONINFOA = record
13623:   dwOSVersionInfoSize: DWORD;
13624:   dwMajorVersion: DWORD;
13625:   dwMinorVersion: DWORD;
13626:   dwBuildNumber: DWORD;
13627:   dwPlatformId: DWORD;
```

```
13628:   szCSDVersion: array[0..127] of AnsiChar;{ Maintenance Ansistr for PSS usage}
13629: end;
13630:  TOSVersionInfoA', '_OSVERSIONINFOA
13631:   TOSVersionInfo', 'TOSVersionInfoA
13632:  'WS_EX_RIGHT','LongWord')( $00001000);
13633:  'WS_EX_LEFT','LongWord')( $00000000);
13634:  'WS_EX_RTLREADING','LongWord')( $00002000);
13635:  'WS_EX_LTRREADING','LongWord')( $00000000);
13636:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13637:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13638:  Func SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13639:  'LAYOUT_RTL','LongWord')( $00000001);
13640:  'LAYOUT_BTT','LongWord')( $00000002);
13641:  'LAYOUT_VBH','LongWord')( $00000004);
13642:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13643:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13644:  Func SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13645:  Func GetLayout( dc : hdc) : DWORD
13646:  Func IsBidi :Bool
13647:  Func GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13648:  Func GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13649:  Func SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13650:  Func GetPriorityClass( hProcess : THandle) : DWORD
13651:  Func OpenClipboard( hWndNewOwner : HWND) : BOOL
13652:  Func CloseClipboard : BOOL
13653:  Func GetClipboardSequenceNumber : DWORD
13654:  Func GetClipboardOwner : HWND
13655:  Func SetClipboardViewer( hWndNewViewer : HWND) : HWND
13656:  Func GetClipboardViewer : HWND
13657:  Func ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13658:  Func SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13659:  Func GetClipboardData( uFormat : UINT) : THandle
13660:  Func RegisterClipboardFormat( lpszFormat : PChar) : UINT
13661:  Func CountClipboardFormats : Int
13662:  Func EnumClipboardFormats( format : UINT) : UINT
13663:  Func GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Int):Int
13664:  Func EmptyClipboard : BOOL
13665:  Func IsClipboardFormatAvailable( format : UINT) : BOOL
13666:  Func GetPriorityClipboardFormat(var paFormatPriorityList,cFormats:Int):Int
13667:  Func GetOpenClipboardWindow : HWND
13668:  Func EndDialog( hDlg : HWND; nResult : Int) : BOOL
13669:  Func GetDlgItem( hDlg : HWND; nIDDlgItem : Int) : HWND
13670:  Func SetDlgItemInt(hDlg:HWND;nIDDlgItem:Int;uValue:UINT;bSigned:BOOL): BOOL
13671:  Func GetDlgItemInt(hDlg:HWND;nIDDlgItem:Int;var lpTranslated:BOOL;bSigned: BOOL):UINT
13672:  Func SetDlgItemText( hDlg : HWND; nIDDlgItem : Int; lpString : PChar) : BOOL
13673:  Func CheckDlgButton( hDlg : HWND; nIDButton : Int; uCheck : UINT) : BOOL
13674:  Func CheckRadioButton(hDlg:HWND;nIDFirstButton,nIDLastButton,nIDCheckButton:Int):BOOL
13675:  Func IsDlgButtonChecked( hDlg : HWND; nIDButton : Int) : UINT
13676:  Func SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13677: end;
13678:
13679: Proc SIRegister_DXPUtils(CL: TPSPascalCompiler);
13680: begin
13681:  Func glExecuteAndWait(cmdLine:Str;visibility:Word;timeout:Card;killAppOnTimeOut:Bool):Int;
13682:  Func GetTemporaryFilesPath :Str
13683:  Func GetTemporaryFileName :Str
13684:  Func FindFileInPaths( const fileName, paths :Str) :Str
13685:  Func PathsToString( const paths : TStrings) :Str
13686:  Proc StringToPaths( const pathsString :Str; paths : TStrings)
13687:  //Func MacroExpandPath( const aPath :Str) :Str
13688: end;
13689:
13690: Proc SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13691: begin
13692:   SIRegister_TALMultiPartBaseContent(CL);
13693:   SIRegister_TALMultiPartBaseContents(CL);
13694:   SIRegister_TAlMultiPartBaseStream(CL);
13695:   SIRegister_TALMultipartBaseEncoder(CL);
13696:   SIRegister_TALMultipartBaseDecoder(CL);
13697:  Func ALMultipartExtractBoundaryFromContentType(aContentType:Ansistr): Ansistr
13698:  Func ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:Ansistr;aName:Ansistr):Ansistr;
13699:  Func ALMultipartSetSubValueInHeaderLine(aHeaderLine:Ansistr;aName,AValue:Ansistr):Ansistr;
13700: end;
13701:
13702: Proc SIRegister_SmallUtils(CL: TPSPascalCompiler);
13703: begin
13704:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13705:   TWinVerRec','record WinPlatform : Int; WinMajorVersion : In'
13706:    +teger; WinMinorVersion :Int; WinBuildNumber : Int; WinCSDVersion:Str; end
13707:  Func aAllocPadedMem( Size :Card) : TObject
13708:  Proc aFreePadedMem( var P : TObject);
13709:  Proc aFreePadedMem1( var P : PChar);
13710:  Func aCheckPadedMem( P : Pointer) : Byte
13711:  Func aGetPadMemSize( P : Pointer) :Card
13712:  Func aAllocMem( Size :Card) : Pointer
13713:  Func aStrLen( const Str : PChar) :Card
13714:  Func aStrLCopy( Dest : PChar; const Source : PChar; MaxLen :Card) : PChar
13715:  Func aStrECopy( Dest : PChar; const Source : PChar) : PChar
13716:  Func aStrCopy( Dest : PChar; const Source : PChar) : PChar
```

```
13717:  Func aStrEnd( const Str : PChar) : PChar
13718:  Func aStrScan( const Str : PChar; aChr : Char) : PChar
13719:  Func aStrMove( Dest : PChar; const Source : PChar; Count :Card) : PChar
13720:  Func aPCharLength( const Str : PChar) :Card
13721:  Func aPCharUpper( Str : PChar) : PChar
13722:  Func aPCharLower( Str : PChar) : PChar
13723:  Func aStrCat( Dest : PChar; const Source : PChar) : PChar
13724:  Func aLastDelimiter( const Delimiters, S :Str) : Int
13725:  Func aCopyTail( const S :Str; Len : Int) :Str
13726:  Func aInt2Thos( I : Int64) :Str
13727:  Func aUpperCase( const S :Str) :Str
13728:  Func aLowerCase( const S :Str) :Str
13729:  Func aCompareText( const S1, S2 :Str) : Int
13730:  Func aSameText( const S1, S2 :Str) :Bool
13731:  Func aInt2Str( Value : Int64) :Str
13732:  Func aStr2Int( const Value :Str) : Int64
13733:  Func aStr2IntDef( const S :Str; Default : Int64) : Int64
13734:  Func aGetFileExt( const FileName :Str) :Str
13735:  Func aGetFilePath( const FileName :Str) :Str
13736:  Func aGetFileName( const FileName :Str) :Str
13737:  Func aChangeExt( const FileName, Extension :Str) :Str
13738:  Func aAdjustLineBreaks( const S :Str) :Str
13739:  Func aGetWindowStr( WinHandle : HWND) :Str
13740:  Func aDiskSpace( Drive :Str) : TdriveSize
13741:  Func aFileExists( FileName :Str) :Bool
13742:  Func aFileSize( FileName :Str) : Int64
13743:  Func aDirectoryExists( const Name :Str) :Bool
13744:  Func aSysErrorMessage( ErrorCode : Int) :Str
13745:  Func aShortPathName( const LongName :Str) :Str
13746:  Func aGetWindowVer : TWinVerRec
13747:  Proc InitDriveSpacePtr;
13748: end;
13749:
13750: Proc SIRegister_MakeApp(CL: TPSPascalCompiler);
13751: begin
13752:  aZero','LongInt'( 0);
13753:  'makeappDEF','LongInt'( - 1);
13754:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13755:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13756:  'CS_KEYCVTWINDOW','LongInt'( 4);
13757:  'CS_DBLCLKS','LongInt'( 8);
13758:  'CS_OWNDC','LongWord')( $20);
13759:  'CS_CLASSDC','LongWord')( $40);
13760:  'CS_PARENTDC','LongWord')( $80);
13761:  'CS_NOKEYCVT','LongWord')( $100);
13762:  'CS_NOCLOSE','LongWord')( $200);
13763:  'CS_SAVEBITS','LongWord')( $800);
13764:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13765:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13766:  'CS_GLOBALCLASS','LongWord')( $4000);
13767:  'CS_IME','LongWord')( $10000);
13768:  'CS_DROPSHADOW','LongWord')( $20000);
13769:   //PPanelFunc', '^TPanelFunc // will not work
13770:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13771:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13772:   TFontLooks', 'set of TFontLook
13773:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Int):Int)
13774:  Func SetWinClass(const ClassName:str; pMessFunc: Tmessagefunc; wcStyle : Int): Word
13775:  Func SetWinClassO( const ClassNam:str; pMessFunc: TObject; wcStyle : Int): Word
13776:  Func SetWinClass(const ClassName:str; pMessFunc : TObject; wcStyle : Int) : Word
13777:  Func MakeForm(Left,Top,Width,Height:Int;const Caption:str;WinStyle:Int):Int
13778:  Proc RunMsgLoop( Show :Bool)
13779:  Func MakeFont(Height,Width:Int;const FontName:str; Look:TFontLooks;Roman:Boolean):Int
13780:  Func MakeButton(Left,Top,Width,Height:Int;pCaption:PChar;hParent,ID_Number:Cardi;hFont:Int):Int;
13781:  Func MakeListBox(Left,Top,Width,Height,Parent:Int;const ListItems:str;WinStyle:Int):Int
13782:  Func MakeComboBox(Left,Top,Width,Height,Parent:Int;const ListItems:str;WinStyle:Int):Int
13783:  Func MakePanel(Left,Top,Width,Height,hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13784:  Func MakeSubMenu(const ItemList :Str; ID1, ID2 :Card; hMenu : Int) : Int
13785:  Func id4menu( a, b : Byte; c : Byte; d : Byte) :Card
13786:  Proc DoInitMakeApp   //set first to init formclasscontrol!
13787: end;
13788:
13789: Proc SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13790: begin
13791:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13792:    +KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13793:   TScreenSaverOptions', 'set of TScreenSaverOption
13794:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13795:   TScreenSaverPreviewEvent', 'Proc ( Sender : TObject; previewHwnd: HWND)
13796:   SIRegister_TScreenSaver(CL);
13797:  //Proc Register
13798:  Proc SetScreenSaverPassword
13799: end;
13800:
13801: Proc SIRegister_XCollection(CL: TPSPascalCompiler);
13802: begin
13803:   FindClass('TOBJECT'),'TXCollection
13804:   SIRegister_EFilerException(CL);
```

```
13805:   SIRegister_TXCollectionItem(CL);
13806:   //TXCollectionItemClass', 'class of TXCollectionItem
13807:   SIRegister_TXCollection(CL);
13808:  Proc RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13809:  Proc DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13810:  Proc RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13811:  Proc UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13812:  Func FindXCollectionItemClass( const className :Str) : TXCollectionItemClass
13813:  Func GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13814: end;
13815:
13816: Proc SIRegister_XOpenGL(CL: TPSPascalCompiler);
13817: begin
13818:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13819:  Proc xglMapTexCoordToNull
13820:  Proc xglMapTexCoordToMain
13821:  Proc xglMapTexCoordToSecond
13822:  Proc xglMapTexCoordToDual
13823:  Proc xglMapTexCoordToArbitrary( const units : array of Cardinal);
13824:  Proc xglMapTexCoordToArbitrary1( const bitWiseUnits :Card);
13825:  Proc xglMapTexCoordToArbitraryAdd( const bitWiseUnits :Card)
13826:  Proc xglBeginUpdate
13827:  Proc xglEndUpdate
13828:  Proc xglPushState
13829:  Proc xglPopState
13830:  Proc xglForbidSecondTextureUnit
13831:  Proc xglAllowSecondTextureUnit
13832:  Func xglGetBitWiseMapping :Card
13833: end;
13834:
13835: Proc SIRegister_VectorLists(CL: TPSPascalCompiler);
13836: begin
13837:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13838:   TBaseListOptions', 'set of TBaseListOption
13839:   SIRegister_TBaseList(CL);
13840:   SIRegister_TBaseVectorList(CL);
13841:   SIRegister_TAffineVectorList(CL);
13842:   SIRegister_TVectorList(CL);
13843:   SIRegister_TTexPointList(CL);
13844:   SIRegister_TXIntList(CL);
13845:   //PSingleArrayList', '^TSingleArrayList // will not work
13846:   SIRegister_TSingleList(CL);
13847:   SIRegister_TByteList(CL);
13848:   SIRegister_TQuaternionList(CL);
13849:  Proc QuickSortLists(startIndex,endIndex:Int;refList:TSingleList; objList : TList);
13850:  Proc QuickSortLists1(startIndex,endIndex:Int;refList: TSingleList; objList: TBaseList);
13851:  Proc FastQuickSortLists(startIndex,endIndex:Int;refList:TSingleList;objList:TPersistentObjectList);
13852: end;
13853:
13854: ********** File C:\maXbox\maxbox3\maxbox3\maXbox3\source\REST\uPSI_UIntList.pas
13855:            File C:\maXbox\maxbox3\maxbox3\maXbox3\source\JvJCLUtils.pas
13856: { compile-time registration functions }
13857: 24: Proc SIRegister_TIntList(CL: TPSPascalCompiler);
13858: Proc SIRegister_UIntList(CL: TPSPascalCompiler);
13859: { run-time registration functions }
13860: 28: Proc RIRegister_TIntList(CL: TPSRuntimeClassImporter);
13861: Proc RIRegister_UIntList(CL: TPSRuntimeClassImporter);
13862: (*----------------------------------------------------------------------------*)
13863: 49: Proc SIRegister_TIntList(CL: TPSPascalCompiler);
13864: 51:   //with RegClassS(CL,'TPersistent', 'TIntList') do
13865:   with ClassN(CL.FindClass('TPersistent'),'TIntList') do begin
13866:   with ClassN(CL.FindClass('TList'),'TIntList') do begin
13867:    //with RegClassS(CL,'TPersistent', 'TIntList') do
13868:
13869:  TIntList = class(TPersistent)
13870:   private
13871:     FUpDateCount: Int;
13872:     FList: PIntItemList;
13873:     FCount: Int;
13874:     FCapacity: Int;
13875:     FSorted:Bool;
13876:     FDuplicates: TDuplicates;
13877:     FOnChange: TNotifyEvent;
13878:     FOnChanging: TNotifyEvent;
13879:     Proc ExchangeItems(Index1, Index2: Int);
13880:     Proc Grow;
13881:     Proc QuickSort(L, R: Int; SCompare: TIntListSortCompare);
13882:     Proc InsertItem(Index: Int; const S: int64);
13883:     Proc SetSorted(Value:Bool);
13884:   protected
13885:     Proc Error(const Msg:Str; Data: Int);
13886:     Proc Changed; virtual;
13887:     Proc Changing; virtual;
13888:     Func Get(Index: Int): int64;
13889:     Func GetCapacity: Int;
13890:     //Func GetCount: Int;
13891:     Func GetObject(Index: Int): TObject;
13892:     Proc Put(Index: Int; const S: int64);
13893:     Proc PutObject(Index: Int; AObject: TObject);
```

```
13894:     Proc SetCapacity(NewCapacity: Int);
13895:     Proc SetUpdateState(Updating:Bool);
13896:   public
13897:     Func GetCount: Int;
13898:     destructor Destroy; override;
13899:     Func Add(const S: int64): Int;
13900:     Func AddObject(const S: int64; AObject: TObject): Int; virtual;
13901:     Proc Clear;
13902:     Proc Delete(Index: Int);
13903:     Proc Exchange(Index1, Index2: Int);
13904:     Func Find(const S: int64; var Index: Int):Bool; virtual;
13905:     Func IndexOf(const S: int: Int;
13906:     Proc Insert(Index: Int; const S: int64);
13907:     Proc Sort; virtual;
13908:     Proc CustomSort(Compare: TIntListSortCompare); virtual;
13909:     Proc LoadFromFile(const FileName:Str); virtual;
13910:     Proc LoadFromStream(Stream:TStream); virtual;
13911:     Proc SaveToFile(const FileName:Str); virtual;
13912:     Proc SaveToStream(Stream: TStream);
13913:     property Duplicates: TDuplicates read FDuplicates write FDuplicates;
13914:     property Sorted:Bool read FSorted write SetSorted;
13915:     property OnChange: TNotifyEvent read FOnChange write FOnChange;
13916:     property OnChanging: TNotifyEvent read FOnChanging write FOnChanging;
13917:     property Ints [Index: Int]: int64 read Get write Put; default;
13918:     property Count: Int read GetCount;
13919:     property Objects[Index: Int]: TObject read GetObject write PutObject;
13920:   end;
13921:
13922: procedure SIRegister_TInteger(CL: TPSPascalCompiler);  //biginteger   bigint
13923: begin
13924:   //with RegClassS(CL,'TObject', 'TInteger') do
13925:   with CL.AddClassN(CL.FindClass('TObject'),'TInteger') do begin
13926:     Constructor Create1( const initialValue : int64)
13927:     Constructor Create
13928:     Procedure Free
13929:     RegisterProperty('Digits', 'TDigits', iptr);
13930:     Procedure Assign( const I2 : TInteger);');
13931:     Procedure Assign1( const I2 : int64);
13932:     Procedure Assign2( const I2 : string);
13933:     Procedure AbsoluteValue');
13934:     Procedure Add( const I2 : TInteger);
13935:     Procedure Add1( const I2 : int64);
13936:     Procedure AssignZero');
13937:     Procedure AssignOne');
13938:     Procedure Subtract( const I2 : TInteger);
13939:     Procedure Subtract1( const I2 : int64);
13940:     Procedure Mult( const I2 : TInteger);
13941:     Procedure Mult1( const I2 : int64);
13942:     Procedure FastMult( const I2 : TInteger);
13943:     Procedure Divide( const I2 : TInteger);
13944:     Procedure Divide1( const I2 : int64);
13945:     Procedure Modulo( const I2 : TInteger);
13946:     Procedure Modulo1( const N : int64);
13947:     Procedure ModPow( const I2, m : TInteger);
13948:     Procedure InvMod( I2 : TInteger);
13949:     Procedure DivideRem( const I2 : TInteger; var remain : TInteger);
13950:     Procedure DivideRemTrunc( const I2 : TInteger; var remain : TInteger);
13951:     Procedure DivideRemFloor( const I2 : TInteger; var remain : TInteger);
13952:     Procedure DivideRemEuclidean( const I2 : TInteger; var remain : TInteger);
13953:     Function Compare( I2 : TInteger) : integer;
13954:     Function Compare1( I2 : int64) : integer;
13955:     Procedure Factorial');
13956:     Function ConvertToDecimalString( commas : boolean) : string');
13957:     Function ConvertToInt64( var N : int64) : boolean');
13958:     Function DigitCount : ');
13959:     Procedure SetSign( s : integintegerer);
13960:     Function GetSign : integer');
13961:     Function IsOdd : boolean');
13962:     Function IsPositive : boolean');
13963:     Function IsNegative : boolean');
13964:     Function IsProbablyPrime : boolean');
13965:     Function IsZero : boolean');
13966:     Procedure ChangeSign');
13967:     Procedure Pow( const exponent : int64);
13968:     Procedure Sqroot');
13969:     Procedure Square');
13970:     Procedure FastSquare');
13971:     Procedure Gcd( const I2 : TInteger);
13972:     Procedure Gcd1( const I2 : int64);
13973:     Procedure NRoot( const Root : int64);
13974:     Function GetBase : integer');
13975:     Function BitCount : integer');
13976:     Function ConvertToHexString : String');
13977:     Function AssignRandomPrime( BitLength:integer; seed:String; mustMatchBitLength:boolean):boolean');
13978:     Function AssignHex( HexStr : String) : boolean');
13979:     Procedure RandomOfSize( size : integer);
13980:     Procedure Random( maxint : TInteger);
13981:     Procedure Getnextprime');
13982:   end;
```

```
13983: end;
13984:
13985: procedure SIRegister_TBigFloat(CL: TPSPascalCompiler); //bigdecimal bigdec
13986: begin
13987:   //with RegClassS(CL,'TObject', 'TBigFloat') do
13988:   with CL.AddClassN(CL.FindClass('TObject'),'TBigFloat') do begin
13989:     decpart', 'TFloatInt', iptrw);
13990:     sigdigits', 'word', iptrw);
13991:     exponent', 'integer', iptrw);
13992:     Constructor Create;
13993:     Constructor Create1( const MaxSig : TMaxSig);
13994:     Procedure Assign( A : TBigFloat);
13995:     Procedure Assign3( A : TBigFloat; SigDig : word);
13996:     Procedure Assign4( A : TInteger);
13997:     Procedure Assign5( N : int64);
13998:     Procedure Assign6( N : int64; SigDig : integer);
13999:     Procedure Assign7( d : extended);
14000:     Procedure Assign8( S : string);
14001:     Procedure Assign9( S : string; SigDig : word);
14002:     Procedure AssignZero');
14003:     Procedure AssignOne');
14004:     Procedure Add( B : TBigFloat);
14005:     Procedure Add1( B : int64);
14006:     Procedure AbsAdd( B : TBigFloat);
14007:     Procedure Subtract( B : TBigFloat);
14008:     Procedure Subtract1( B : int64);
14009:     Procedure Mult( B : TBigFloat);
14010:     Procedure Mult1( B : TBigfloat; const MaxSig : TMaxSig);
14011:     Procedure Mult2( B : TInteger);
14012:     Procedure Mult3( B : int64);
14013:     Procedure MultRaw( B : TBigFloat);
14014:     Procedure Reciprocal( const MaxSig : TMaxSig);
14015:     Procedure Divide( B : TBigFloat; const MaxSig : TMaxSig);
14016:     Procedure Divide1( B : TInteger; const MaxSig : TMaxSig);
14017:     Procedure Divide2( B : int64; const MaxSig : TMaxSig);
14018:     Procedure Square( const MaxSig : TMaxSig);
14019:     Function Compare( B : TBigFloat) : integer');
14020:     Function IsZero : boolean');
14021:     Procedure MaxBigFloat( B : TBigFloat);
14022:     Procedure MinBigFloat( B : TBigFloat);
14023:     Procedure Sqrt;
14024:     Procedure Sqrt1( const MaxSig : TMaxSig);
14025:     Procedure NRoot( N : integer; const MaxSig : TMaxSig);
14026:     Procedure IntPower( intpower : integer; const MaxSig : TMaxSig);
14027:     Procedure Power( power : TBigfloat; const MaxSig : TMaxSig);
14028:     Procedure Log( const MaxSig : TMaxSig);
14029:     Procedure Log10( const MaxSig : TMaxSig);
14030:     Procedure Exp( const MaxSig : TMaxSig);
14031:     Procedure PiConst( const MaxSig : TMaxSig);
14032:     Procedure Log2Const( const MaxSig : TMaxSig);
14033:     Procedure RoundToPrec( const MaxSig : TMaxSig);
14034:     Procedure RoundToPrec1;
14035:     Procedure Trunc( const x : integer);
14036:     Procedure Floor( const x : integer);
14037:     Procedure Ceiling( const x : integer);
14038:     Procedure Round( const x : integer);
14039:     Procedure AbsoluteValue');
14040:     Procedure Negate');
14041:     Procedure SetSigDigits( const newsigdigits : integer);
14042:     Function ConvertToString( const View : TView) : string');
14043:     Function ToString( const View : TView) : string');
14044:     Function ConvertToExtended( var num : extended) : boolean');
14045:     Function ConvertToInt64( var N : int64) : boolean');
14046:     Function IntPart : int64');
14047:   end;
14048: end;
14049:
14050: Proc SIRegister_MeshUtils(CL: TPSPascalCompiler);
14051: begin
14052:  Proc ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
14053:  Proc ConvertStripToList1( const strip : TIntList; list : TIntList);
14054:  Proc ConvertStripToList2(const strip:TAffineVectorList;const indices:TIntList;list:TAffineVectorList);
14055:  Proc ConvertIndexedListToList(const data:TAffineVectlist;const indices:TIntList;list:TAffineVectorList);
14056:  Func BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList;const
       normals:TAffineVectorList;const texCoords:TAffineVectorList):TIntList
14057:  Proc RemapReferences( reference : TAffineVectorList; const indices : TIntList);
14058:  Proc RemapReferences1( reference : TIntList; const indices : TIntList);
14059:  Proc RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntList)
14060:  Func RemapIndicesToIndicesMap( remapIndices : TIntList) : TIntList
14061:  Proc RemapTrianglesIndices( indices, indicesMap : TIntList)
14062:  Proc RemapIndices( indices, indicesMap : TIntList)
14063:  Proc UnifyTrianglesWinding( indices : TIntList)
14064:  Proc InvertTrianglesWinding( indices : TIntList)
14065:  Func BuildNormals( reference : TAffineVectorList; indices : TIntList) : TAffineVectorList
14066:  Func
       BuildNonOrientedEdgesList(triangleIndices:TIntList;triangleEdges:TIntList;edgesTriangles:TIntList):TIntList
14067:  Proc WeldVertices(vertices:TAffineVectorList;indicesMap:TIntList;weldRadius: Single)
14068:  Func StripifyMesh(indices:TIntList;maxVertexIndex:Int;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
```

```
14069:  Proc IncreaseCoherency(indices : TIntList; cacheSize : Int)
14070:  Proc
       SubdivideTriangles(smoothFactor:Single;vertices:TAffineVectorList;triangleIndices:TIntList;normals:TAffineVectorList;o
14071: end;
14072:
14073: Proc SIRegister_JclSysUtils(CL: TPSPascalCompiler);
14074: begin
14075:  Proc GetAndFillMem( var P : TObject; const Size : Int; const Value : Byte)
14076:  Proc FreeMemAndNil( var P : TObject)
14077:  Func PCharOrNil( const S :Str) : PChar
14078:   SIRegister_TJclReferenceMemoryStream(CL);
14079:   FindClass('TOBJECT'),'EJclVMTError
14080:  {Func GetVirtualMethodCount( AClass : TClass) : Int
14081:  Func GetVirtualMethod( AClass : TClass; const Index : Int) : Pointer
14082:  Proc SetVirtualMethod( AClass : TClass; const Index : Int; const Method:Pointer)
14083:   PDynamicIndexList', '^TDynamicIndexList // will not work
14084:   PDynamicAddressList', '^TDynamicAddressList // will not work
14085:  Func GetDynamicMethodCount( AClass : TClass) : Int
14086:  Func GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
14087:  Func GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
14088:  Func HasDynamicMethod( AClass : TClass; Index : Int) :Bool
14089:  Func GetDynamicMethod( AClass : TClass; Index : Int) : Pointer
14090:  Func GetInitTable( AClass : TClass) : PTypeInfo
14091:   PFieldEntry', '^TFieldEntry // will not work}
14092:   TFieldEntry', 'record OffSet : Int; IDX : Word; Name : ShortString; end
14093:  Func JIsClass( Address : Pointer) :Bool
14094:  Func JIsObject( Address : Pointer) :Bool
14095:  Func GetImplementorOfInterface( const I : IInterface) : TObject
14096:   TDigitCount', 'Int
14097:   SIRegister_TJclNumericFormat(CL);
14098:  Func JIntToStrZeroPad( Value, Count : Int) : Ansistr
14099:   TTextHandler', 'Proc ( const Text :Str)
14100: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
14101:  Func JExecute(const CommandLine:str;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
14102:  Func JExecute1(const CommandLine:str;var Output:str;RawOutput:Bool;AbortPtr:PBool):Card;
14103:  Func JExecute2(const CommandLine:str;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
14104:  Func JExecute3('cmd /C dir *.*',@TTextHandlerQ, true, false)));
14105:  Func JExecute3('cmd /C dir *.*',Nil, true, false)));
14106:  Func ReadKey : Char   //to and from the DOS console !
14107:   TModuleHandle', 'HINST
14108:   //TModuleHandle', 'Pointer
14109:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
14110:  Func LoadModule( var Module : TModuleHandle; FileName :Str) :Bool
14111:  Func LoadModuleEx(var Module: TModuleHandle; FileName:str;Flags:Card):Bool
14112:  Proc UnloadModule( var Module : TModuleHandle)
14113:  Func GetModuleSymbol( Module : TModuleHandle; SymbolName :Str) : Pointer
14114:  Func GetModuleSymbolEx(Module:TModuleHandle;SymbolName:str;var Accu:Boolean):Pointer
14115:  Func ReadModuleData(Module:TModuleHandle;SymbolName:str;var Buffer,Size:Card):Bool;
14116:  Func WriteModuleData(Module:TModuleHandle;SymbolName:str;var Buffer,Size:Card):Bool;
14117:   FindClass('TOBJECT'),'EJclConversionError
14118:  Func JStrToBoolean( const S :Str) :Bool
14119:  Func JBooleanToStr( B :Bool) :Str
14120:  Func JIntToBool( I : Int) :Bool
14121:  Func JBoolToInt( B :Bool) : Int
14122:  'ListSeparator','String '; 'ListSeparator1','String ':
14123:  Proc ListAddItems( var List :Str; const Separator, Items:Str)
14124:  Proc ListIncludeItems( var List :Str; const Separator, Items :Str)
14125:  Proc ListRemoveItems( var List :Str; const Separator, Items :Str)
14126:  Proc ListDelItem( var List :Str; const Separator :Str; const Index : Int)
14127:  Func ListItemCount( const List, Separator :Str) : Int
14128:  Func ListGetItem( const List, Separator :Str; const Index : Int) :Str
14129:  Proc ListSetItem(var List:str;const Separator:str;const Index:Int;const Value:str)
14130:  Func ListItemIndex( const List, Separator, Item :Str) : Int
14131:  Func SystemTObjectInstance : LongWord
14132:  Func IsCompiledWithPackages :Bool
14133:  Func JJclGUIDToString( const GUID : TGUID) :Str
14134:  Func JJclStringToGUID( const S :Str) : TGUID
14135:   SIRegister_TJclIntfCriticalSection(CL);
14136:   SIRegister_TJclSimpleLog(CL);
14137:  Proc InitSimpleLog( const ALogFileName :Str)
14138: end;
14139:
14140: Proc SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
14141: begin
14142:   FindClass('TOBJECT'),'EJclBorRADException
14143:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
14144:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
14145:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
14146:   TJclBorRADToolPath', 'string
14147:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
14148:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
14149:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
14150:  BorRADToolRepositoryPagesSection','String 'Repository Pages
14151:  BorRADToolRepositoryDialogsPage','String 'Dialogs
14152:  BorRADToolRepositoryFormsPage','String 'Forms
14153:  BorRADToolRepositoryProjectsPage','String 'Projects
14154:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
14155:  BorRADToolRepositoryObjectType','String 'Type
14156:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
```

```
14157:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
14158:  BorRADToolRepositoryObjectName','String 'Name
14159:  BorRADToolRepositoryObjectPage','String 'Page
14160:  BorRADToolRepositoryObjectIcon','String 'Icon
14161:  BorRADToolRepositoryObjectDescr','String 'Description
14162:  BorRADToolRepositoryObjectAuthor','String 'Author
14163:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
14164:  BorRADToolRepositoryObjectDesigner','String 'Designer
14165:  BorRADToolRepositoryDesignerDfm','String 'dfm
14166:  BorRADToolRepositoryDesignerXfm','String 'xfm
14167:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
14168:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
14169:  SourceExtensionDelphiPackage','String '.dpk
14170:  SourceExtensionBCBPackage','String '.bpk
14171:  SourceExtensionDelphiProject','String '.dpr
14172:  SourceExtensionBCBProject','String '.bpr
14173:  SourceExtensionBDSProject','String '.bdsproj
14174:  SourceExtensionDProject','String '.dproj
14175:  BinaryExtensionPackage','String '.bpl
14176:  BinaryExtensionLibrary','String '.dll
14177:  BinaryExtensionExecutable','String '.exe
14178:  CompilerExtensionDCP','String '.dcp
14179:  CompilerExtensionBPI','String '.bpi
14180:  CompilerExtensionLIB','String '.lib
14181:  CompilerExtensionTDS','String '.tds
14182:  CompilerExtensionMAP','String '.map
14183:  CompilerExtensionDRC','String '.drc
14184:  CompilerExtensionDEF','String '.def
14185:  SourceExtensionCPP','String '.cpp
14186:  SourceExtensionH','String '.h
14187:  SourceExtensionPAS','String '.pas
14188:  SourceExtensionDFM','String '.dfm
14189:  SourceExtensionXFM','String '.xfm
14190:  SourceDescriptionPAS','String 'Pascal source file
14191:  SourceDescriptionCPP','String 'C++ source file
14192:  DesignerVCL','String 'VCL
14193:  DesignerCLX','String 'CLX
14194:  ProjectTypePackage','String 'package
14195:  ProjectTypeLibrary','String 'library
14196:  ProjectTypeProgram','String 'program
14197:  Personality32Bit','String '32 bit
14198:  Personality64Bit','String '64 bit
14199:  PersonalityDelphi','String 'Delphi
14200:  PersonalityDelphiDotNet','String 'Delphi.net
14201:  PersonalityBCB','String 'C++Builder
14202:  PersonalityCSB','String 'C#Builder
14203:  PersonalityVB','String 'Visual Basic
14204:  PersonalityDesign','String 'Design
14205:  PersonalityUnknown','String 'Unknown personality
14206:  PersonalityBDS','String 'Borland Developer Studio
14207:  DOFDirectoriesSection','String 'Directories
14208:  DOFUnitOutputDirKey','String 'UnitOutputDir
14209:  DOFSearchPathName','String 'SearchPath
14210:  DOFConditionals','String 'Conditionals
14211:  DOFLinkerSection','String 'Linker
14212:  DOFPackagesKey','String 'Packages
14213:  DOFCompilerSection','String 'Compiler
14214:  DOFPackageNoLinkKey','String 'PackageNoLink
14215:  DOFAdditionalSection','String 'Additional
14216:  DOFOptionsKey','String 'Options
14217:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
14218:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
14219:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
14220:   TJclBorPersonalities', 'set of TJclBorPersonality
14221:   TJclBorDesigner', '( bdVCL, bdCLX )
14222:   TJclBorDesigners', 'set of TJClBorDesigner
14223:   TJclBorPlatform', '( bp32bit, bp64bit )
14224:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
14225:   SIRegister_TJclBorRADToolInstallationObject(CL);
14226:   SIRegister_TJclBorlandOpenHelp(CL);
14227:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
14228:   TJclHelp2Objects', 'set of TJclHelp2Object
14229:   SIRegister_TJclHelp2Manager(CL);
14230:   SIRegister_TJclBorRADToolIdeTool(CL);
14231:   SIRegister_TJclBorRADToolIdePackages(CL);
14232:   SIRegister_IJclCommandLineTool(CL);
14233:   FindClass('TOBJECT'),'EJclCommandLineToolError
14234:   SIRegister_TJclCommandLineTool(CL);
14235:   SIRegister_TJclBorlandCommandLineTool(CL);
14236:   SIRegister_TJclBCC32(CL);
14237:   SIRegister_TJclDCC32(CL);
14238:   TJclDCC', 'TJclDCC32
14239:   SIRegister_TJclBpr2Mak(CL);
14240:   SIRegister_TJclBorlandMake(CL);
14241:   SIRegister_TJclBorRADToolPalette(CL);
14242:   SIRegister_TJclBorRADToolRepository(CL);
14243:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
14244:   TCommandLineTools', 'set of TCommandLineTool
14245:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
```

```
14246:   SIRegister_TJclBorRADToolInstallation(CL);
14247:   SIRegister_TJclBCBInstallation(CL);
14248:   SIRegister_TJclDelphiInstallation(CL);
14249:   SIRegister_TJclDCCIL(CL);
14250:   SIRegister_TJclBDSInstallation(CL);
14251:   TTraverseMethod', 'Func ( Installation: TJclBorRADToolInstallation) :Bool
14252:   SIRegister_TJclBorRADToolInstallations(CL);
14253:  Func BPLFileName( const BPLPath, PackageFileName :Str) :Str
14254:  Func BinaryFileName( const OutputPath, ProjectFileName :Str) :Str
14255:  Func IsDelphiPackage( const FileName :Str) :Bool
14256:  Func IsDelphiProject( const FileName :Str) :Bool
14257:  Func IsBCBPackage( const FileName :Str) :Bool
14258:  Func IsBCBProject( const FileName :Str) :Bool
14259:  Proc GetDPRFileInfo(const DPRFileName:str;out BinaryExtensio:str;const LibSuffx:PString);
14260:  Proc GetBPRFileInfo(const BPRFileName:str;out BinaryFileName:str;const Descript:PString);
14261:  Proc GetDPKFileInfo(const DPKFileName:str;out RunOnly:Bool;const LibSuffix:PString;const Descript:PString;
14262:  Proc GetBPKFileInfo(const BPKFileName:str;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
14263:  Func SamePath(const Path1, Path2:Str):Bool;
14264: end;
14265:
14266: Proc SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
14267: begin
14268:  'ERROR_NO_MORE_FILES','LongInt'( 18);
14269:  //Func stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Int
14270:  //Func fstat64( FileDes: Int;var StatBuffer : TStatBuf64) : Int
14271:  //Func lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Int
14272:  'LPathSeparator','String '/
14273:  'LDirDelimiter','String '/
14274:  'LDirSeparator','String ':
14275:  'JXPathDevicePrefix','String '\\.\
14276:  'JXPathSeparator','String '\
14277:  'JXDirDelimiter','String '\
14278:  'JXDirSeparator','String ';
14279:  'JXPathUncPrefix','String '\\
14280:  'faNormalFile','LongWord')( $00000080);
14281:  //'faUnixSpecific',' faSymLink);
14282:   JXTCompactPath', '( cpCenter, cpEnd )
14283:      _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
14284:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
14285:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
14286:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
14287:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
14288:  Func jxPathAddSeparator( const Path :Str) :Str
14289:  Func jxPathAddExtension( const Path, Extension :Str) :Str
14290:  Func jxPathAppend( const Path, Append :Str) :Str
14291:  Func jxPathBuildRoot( const Drive : Byte) :Str
14292:  Func jxPathCanonicalize( const Path :Str) :Str
14293:  Func jxPathCommonPrefix( const Path1, Path2 :Str) : Int
14294:  Func jxPathCompactPath(const DC:HDC;const Path:str;const Width:Int;CmpFmt:TCompactPath):str
14295:  Proc jxPathExtractElements(const Source:str;var Drive,Path,FileName,Ext:str)
14296:  Func jxPathExtractFileDirFixed( const S :Str) :Str
14297:  Func jxPathExtractFileNameNoExt( const Path :Str) :Str
14298:  Func jxPathExtractPathDepth( const Path :Str; Depth : Int) :Str
14299:  Func jxPathGetDepth( const Path :Str) : Int
14300:  Func jxPathGetLongName( const Path :Str) :Str
14301:  Func jxPathGetShortName( const Path :Str) :Str
14302:  Func jxPathGetLongName( const Path :Str) :Str
14303:  Func jxPathGetShortName( const Path :Str) :Str
14304:  Func jxPathGetRelativePath( Origin, Destination :Str) :Str
14305:  Func jxPathGetTempPath :Str
14306:  Func jxPathIsAbsolute( const Path :Str) :Bool
14307:  Func jxPathIsChild( const Path, Base :Str) :Bool
14308:  Func jxPathIsDiskDevice( const Path :Str) :Bool
14309:  Func jxPathIsUNC( const Path :Str) :Bool
14310:  Func jxPathRemoveSeparator( const Path :Str) :Str
14311:  Func jxPathRemoveExtension( const Path :Str) :Str
14312:  Func jxPathGetPhysicalPath( const LocalizedPath :Str) :Str
14313:  Func jxPathGetLocalizedPath( const PhysicalPath :Str) :Str
14314:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
14315:   JxTFileListOptions', 'set of TFileListOption
14316:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
14317:   TFileHandler', 'Proc ( const FileName :Str)
14318:   TFileHandlerEx', 'Proc ( const Directory :Str; const FileInfo : TSearchRec)
14319:  Func BuildFileList( const Path :Str; const Attr : Int; const List : TStrings) :Bool
14320:  //Func AdvBuildFileList( const Path :Str; const Attr : Int; const Files : TStrings; const
       AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:str;const
       FileMatchFunc:TFileMatchFunc):Bool;
14321:  Func jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int): Bool
14322:  Func jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Bool;
14323:  Func jxFileAttributesStr( const FileInfo : TSearchRec) :Str
14324:  Func jxIsFileNameMatch(FileName:str;const Mask:str;const CaseSensitive:Bool):Bool;
14325:  Proc jxEnumFiles(const
       Path:str;HandleFile:TFileHandlerEx;RejectedAttributes:Int;RequiredAttributes:Int;Abort:TObject)
14326:  Proc jxEnumDirectories(const Root:str;const HandleDirectory:TFileHandler;const
       IncludeHiddenDirects:Boolean;const SubDirectoriesMask:str;Abort:TObject;ResolveSymLinks:Bool)
14327:  Proc jxCreateEmptyFile( const FileName :Str)
14328:  Func jxCloseVolume( var Volume : THandle) :Bool
14329:  Func jxDeleteDirectory(const DirectoryName:str; MoveToRecycleBin:Boolean:Boolean
```

```
14330:  Func jxCopyDirectory( ExistingDirectoryName, NewDirectoryName :Str) :Bool
14331:  Func jxMoveDirectory( ExistingDirectoryName, NewDirectoryName :Str) :Bool
14332:  Func jxDelTree( const Path :Str) :Bool
14333:  //Func DelTreeEx(const Path:str;AbortOnFailure:Bool; Progress:TDelTreeProgress):Bool
14334:  Func jxDiskInDrive( Drive : Char) :Bool
14335:  Func jxDirectoryExists( const Name :Str; ResolveSymLinks :Bool) :Bool
14336:  Func jxFileCreateTemp( var Prefix :Str) : THandle
14337:  Func jxFileBackup( const FileName :Str; Move :Bool) :Bool
14338:  Func jxFileCopy(const ExistingFileName,NewFileName:str; ReplaceExisting:Boolean):Bool
14339:  Func jxFileDelete( const FileName :Str; MoveToRecycleBin :Bool) :Bool
14340:  Func jxFileExists( const FileName :Str) :Bool
14341:  Func jxFileMove(const ExistingFileName,NewFileName:Str;ReplaceExisting:Bool):Boolean
14342:  Func jxFileRestore( const FileName :Str) :Bool
14343:  Func jxGetBackupFileName( const FileName :Str) :Str
14344:  Func jxIsBackupFileName( const FileName :Str) :Bool
14345:  Func jxFileGetDisplayName( const FileName :Str) :Str
14346:  Func jxFileGetGroupName( const FileName :Str; ResolveSymLinks :Bool) :Str
14347:  Func jxFileGetOwnerName( const FileName :Str; ResolveSymLinks :Bool) :Str
14348:  Func jxFileGetSize( const FileName :Str) : Int64
14349:  Func jxFileGetTempName( const Prefix :Str) :Str
14350:  Func jxFileGetTypeName( const FileName :Str) :Str
14351:  Func jxFindUnusedFileName(FileName:str;const FileExt:str;NumberPrefix:str):str
14352:  Func jxForceDirectories( Name :Str) :Bool
14353:  Func jxGetDirectorySize( const Path :Str) : Int64
14354:  Func jxGetDriveTypeStr( const Drive : Char) :Str
14355:  Func jxGetFileAgeCoherence( const FileName :Str) :Bool
14356:  Proc jxGetFileAttributeList( const Items : TStrings; const Attr : Int)
14357:  Proc jxGetFileAttributeListEx( const Items : TStrings; const Attr : Int)
14358:  Func jxGetFileInformation( const FileName :Str; out FileInfo : TSearchRec) :Bool;
14359:  Func jxGetFileInformation1( const FileName :Str) : TSearchRec;
14360: //Func GetFileStatus(const FileName:str;out StatBuf:TStatBuf64;const ResolveSymLinks:Boolean):Int
14361:  Func jxGetFileLastWrite( const FName :Str) : TFileTime;
14362:  Func jxGetFileLastWrite1( const FName :Str; out LocalTime : TDateTime) :Bool;
14363:  Func jxGetFileLastAccess( const FName :Str) : TFileTime;
14364:  Func jxGetFileLastAccess1(const FName:str; out LocalTime : TDateTime) :Bool;
14365:  Func jxGetFileCreation( const FName :Str) : TFileTime;
14366:  Func jxGetFileCreation1( const FName :Str; out LocalTime : TDateTime) :Bool;
14367:  Func jxGetFileLastWrite(const FName:str;out TimeStamp:Int;ResolveSymLinks:Bool):Bool;
14368:  Func jxGetFileLastWrite1(const FName:str;out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool
14369:  Func jxGetFileLastWrite2(const FName :Str; ResolveSymLinks :Bool) : Int;
14370:  Func jxGetFileLastAccess(const FName:str;out TimeStamp:Int;ResolveSymLinks: Bool): Bool;
14371:  Func jxGetFileLastAccess1(const FName:str;out LocalTime:TDateTime;ResolveSymLinks:Bool)Bool
14372:  Func jxGetFileLastAccess2(const FName:str; ResolveSymLinks:Boolean): Int;
14373:  Func jxGetFileLastAttrChange(const FName:str;out TimeStamp:Int;ResolveSymLinks:Bool):Bool;
14374:  Func jxGetFileLastAttrChange1(const FName:str;out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
14375:  Func jxGetFileLastAttrChange2( const FName :Str; ResolveSymLinks:Boolean): Int;
14376:  Func jxGetModulePath( const Module : HMODULE) :Str
14377:  Func jxGetSizeOfFile( const FileName :Str) : Int64;
14378:  Func jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
14379:  Func jxGetSizeOfFile2( Handle : THandle) : Int64;
14380:  Func jxGetStandardFileInfo( const FileName :Str) : TWin32FileAttributeData
14381:  Func jxIsDirectory( const FileName :Str; ResolveSymLinks :Bool) :Bool
14382:  Func jxIsRootDirectory( const CanonicFileName :Str) :Bool
14383:  Func jxLockVolume( const Volume :Str; var Handle : THandle) :Bool
14384:  Func jxOpenVolume( const Drive : Char) : THandle
14385:  Func jxSetDirLastWrite( const DirName :Str; const DateTime : TDateTime) :Bool
14386:  Func jxSetDirLastAccess( const DirName :Str; const DateTime : TDateTime) :Bool
14387:  Func jxSetDirCreation( const DirName :Str; const DateTime : TDateTime) :Bool
14388:  Func jxSetFileLastWrite( const FileName :Str; const DateTime : TDateTime) :Bool
14389:  Func jxSetFileLastAccess( const FileName :Str; const DateTime : TDateTime) :Bool
14390:  Func jxSetFileCreation( const FileName :Str; const DateTime : TDateTime) :Bool
14391:  Proc jxShredFile( const FileName :Str; Times : Int)
14392:  Func jxUnlockVolume( var Handle : THandle) :Bool
14393:  Func jxCreateSymbolicLink( const Name, Target :Str) :Bool
14394:  Func jxSymbolicLinkTarget( const Name :Str) :Str
14395:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
14396:   SIRegister_TJclCustomFileAttrMask(CL);
14397:   SIRegister_TJclFileAttributeMask(CL);
14398:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
14399:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
14400:   TFileSearchOptions', 'set of TFileSearchOption
14401:   TFileSearchTaskID', 'Int
14402:   TFileSearchTerminationEvent','Procedure(const ID:TFileSearchTaskID;const Aborted:Boolean)
14403:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
14404:   SIRegister_IJclFileEnumerator(CL);
14405:   SIRegister_TJclFileEnumerator(CL);
14406:  Func JxFileSearch : IJclFileEnumerator
14407:   JxTFileFlag', '(ffDebug,ffInfoInferred,ffPatched,ffPreRelease,ffPrivateBuild,ffSpecialBuild )
14408:   JxTFileFlags', 'set of TFileFlag
14409:   FindClass('TOBJECT'),'EJclFileVersionInfoError
14410:   SIRegister_TJclFileVersionInfo(CL);
14411:  Func jxOSIdentToString( const OSIdent : DWORD) :Str
14412:  Func jxOSFileTypeToString(const OSFileType: DWORD; const OSFileSubType : DWORD) :Str
14413:  Func jxVersionResourceAvailable( const FileName :Str) :Bool
14414:   TFileVersionFormat', '( vfMajorMinor, vfFull )
14415:  Func jxFormatVersionString( const HiV, LoV : Word) :Str;
14416:  Func jxFormatVersionString1( const Major, Minor, Build, Revision : Word) :Str;
14417:  //Func FormatVersionString2(const FixedInfo:TVSFixedFileInfo;VersionFrmat:TFileVersionFormat):str;
14418:  //Proc VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
```

```
14419:  //Proc VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
14420:  //Func VersionFixedFileInfo(const FileName:str;var FixedInfo:TVSFixedFileInfo):Boolean
14421:  Func jxVersionFixedFileInfoString(const FileName:str;VersionFormat:TFileVersionFormat;const
        NotAvailableText :str):str
14422:   SIRegister_TJclTempFileStream(CL);
14423:   FindClass('TOBJECT'),'TJclCustomFileMapping
14424:   SIRegister_TJclFileMappingView(CL);
14425:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
14426:   SIRegister_TJclCustomFileMapping(CL);
14427:   SIRegister_TJclFileMapping(CL);
14428:   SIRegister_TJclSwapFileMapping(CL);
14429:   SIRegister_TJclFileMappingStream(CL);
14430:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
14431:   //PPCharArray', '^TPCharArray // will not work
14432:   SIRegister_TJclMappedTextReader(CL);
14433:   SIRegister_TJclFileMaskComparator(CL);
14434:   FindClass('TOBJECT'),'EJclPathError
14435:   FindClass('TOBJECT'),'EJclFileUtilsError
14436:   FindClass('TOBJECT'),'EJclTempFileStreamError
14437:   FindClass('TOBJECT'),'EJclTempFileStreamError
14438:   FindClass('TOBJECT'),'EJclFileMappingError
14439:   FindClass('TOBJECT'),'EJclFileMappingViewError
14440:  Func jxPathGetLongName2( const Path :Str) :Str
14441:  Func jxWin32DeleteFile( const FileName :Str; MoveToRecycleBin :Bool) :Bool
14442:  Func jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName :Str) :Bool
14443:  Func jxWin32BackupFile( const FileName :Str; Move :Bool) :Bool
14444:  Func jxWin32RestoreFile( const FileName :Str) :Bool
14445:  Func jxSamePath( const Path1, Path2 :Str) :Bool
14446:  Proc jxPathListAddItems( var List :Str; const Items :Str)
14447:  Proc jxPathListIncludeItems( var List :Str; const Items :Str)
14448:  Proc jxPathListDelItems( var List :Str; const Items :Str)
14449:  Proc jxPathListDelItem( var List :Str; const Index : Int)
14450:  Func jxPathListItemCount( const List :Str) : Int
14451:  Func jxPathListGetItem( const List :Str; const Index : Int) :Str
14452:  Proc jxPathListSetItem( var List :Str; const Index : Int; const Value :Str)
14453:  Func jxPathListItemIndex( const List, Item :Str) : Int
14454:  Func jxParamName(Idx:Int;const Separator:str;const AllowedPrefixChars:str;TrimName:Bool):str;
14455:  Func jxParamValue(Index:Int; const Separator :Str; TrimValue :Bool) :Str;
14456:  Func jxParamValue1(const SearchName:str; const Separator :Str; CaseSensitive :Bool;
14457:                                  const AllowedPrefixCharacters :Str; TrimValue :Bool) :Str;
14458:  Func jxParamPos(const SearchName:Str;const Separator:str;CaseSensitive:Boolean;const
        AllowedPrefixCharacters string): Int
14459: end;
14460:
14461: Proc SIRegister_FileUtil(CL: TPSPascalCompiler);
14462: begin
14463:  'UTF8FileHeader','String #$ef#$bb#$bf);
14464:  Func lCompareFilenames( const Filename1, Filename2 :Str) : Int
14465:  Func lCompareFilenamesIgnoreCase( const Filename1, Filename2 :Str) : Int
14466:  Func lCompareFilenames(const Filename1,Filename2 :Str; ResolveLinks :Bool): Int
14467:  Func lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLiks:bool):int;
14468:  Func lFilenameIsAbsolute( const TheFilename :Str) :Bool
14469:  Func lFilenameIsWinAbsolute( const TheFilename :Str) :Bool
14470:  Func lFilenameIsUnixAbsolute( const TheFilename :Str) :Bool
14471:  Proc lCheckIfFileIsExecutable( const AFilename :Str)
14472:  Proc lCheckIfFileIsSymlink( const AFilename :Str)
14473:  Func lFileIsReadable( const AFilename :Str) :Bool
14474:  Func lFileIsWritable( const AFilename :Str) :Bool
14475:  Func lFileIsText( const AFilename :Str) :Bool
14476:  Func lFileIsText( const AFilename :Str; out FileReadable :Bool) :Bool
14477:  Func lFileIsExecutable( const AFilename :Str) :Bool
14478:  Func lFileIsSymlink( const AFilename :Str) :Bool
14479:  Func lFileIsHardLink( const AFilename :Str) :Bool
14480:  Func lFileSize( const Filename :Str) : int64;
14481:  Func lGetFileDescription( const AFilename :Str) :Str
14482:  Func lReadAllLinks( const Filename :Str; ExceptionOnError :Bool) :Str
14483:  Func lTryReadAllLinks( const Filename :Str) :Str
14484:  Func lDirPathExists( const FileName :Str) :Bool
14485:  Func lForceDirectory( DirectoryName :Str) :Bool
14486:  Func lDeleteDirectory(const DirectoryName:Str; OnlyChildren:Bool) :Bool
14487:  Func lProgramDirectory :Str
14488:  Func lDirectoryIsWritable( const DirectoryName :Str) :Bool
14489:  Func lExtractFileNameOnly( const AFilename :Str) :Str
14490:  Func lExtractFileNameWithoutExt( const AFilename :Str) :Str
14491:  Func lCompareFileExt( const Filename, Ext :Str; CaseSensitive :Bool) : Int;
14492:  Func lCompareFileExt( const Filename, Ext :Str) : Int;
14493:  Func lFilenameIsPascalUnit( const Filename :Str) :Bool
14494:  Func lAppendPathDelim( const Path :Str) :Str
14495:  Func lChompPathDelim( const Path :Str) :Str
14496:  Func lTrimFilename( const AFilename :Str) :Str
14497:  Func lCleanAndExpandFilename( const Filename :Str) :Str
14498:  Func lCleanAndExpandDirectory( const Filename :Str) :Str
14499:  Func lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory :Str) :Str
14500:  Func lCreateRelativePath(const Filename,BaseDirectory
        string;UsePointDirectory:boolean;AlwaysRequireSharedBaseFolder:Bool):str
14501:  Func lCreateAbsolutePath( const Filename, BaseDirectory :Str) :Str
14502:  Func lFileIsInPath( const Filename, Path :Str) :Bool
14503:  Func lFileIsInDirectory( const Filename, Directory :Str) :Bool
14504:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
```

```
14505:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag AllDirectoryEntriesMask','String '*
14506:  Func lGetAllFilesMask :Str
14507:  Func lGetExeExt :Str
14508:  Func lSearchFileInPath(const Filename,BasePath,SearchPath,Delimiter:Str;Flags:TSearchFileInPathFlags):str
14509:  Func lSearchAllFilesInPath(const Filename,BasePath,SearchPath,Delimiter:str;Flags:
       TSearchFileInPathFlags):TStrings
14510:  Func lFindDiskFilename( const Filename :Str) :Str
14511:  Func lFindDiskFileCaseInsensitive( const Filename :Str) :Str
14512:  Func lFindDefaultExecutablePath( const Executable :Str; const BaseDir:Str):str
14513:  Func lGetDarwinSystemFilename( Filename :Str) :Str
14514:   SIRegister_TFileIterator(CL);
14515:   TFileFoundEvent', 'Proc ( FileIterator : TFileIterator)
14516:   TDirectoryFoundEvent', 'Proc ( FileIterator : TFileIterator)
14517:   TDirectoryEnterEvent', 'Proc ( FileIterator : TFileIterator)
14518:   SIRegister_TFileSearcher(CL);
14519:  Func lFindAllFiles(const SearchPath:str;SearchMsk:str;SearchSubDirs:Bool):TStringList
14520:  Func lFindAllDirectories(const SearchPath:Str; SearchSubDirs:Bool): TStringList
14521:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
14522:  // TCopyFileFlags', 'set of TCopyFileFlag
14523:  Func lCopyFile(const SrcFilename,DestFilename:Str; Flags : TCopyFileFlags):Bool
14524:  Func lCopyFile(const SrcFilename,DestFilename:Str; PreserveTime :Bool):Bool
14525:  Func lCopyDirTree( const SourceDir, TargetDir :Str; Flags : TCopyFileFlags):Bool
14526:  Func lReadFileToString( const Filename :Str) :Str
14527:  Func lGetTempFilename( const Directory, Prefix :Str) :Str
14528:  {Func NeedRTLAnsi :Bool
14529:  Proc SetNeedRTLAnsi( NewValue :Bool)
14530:  Func UTF8ToSys( const s :Str) :Str
14531:  Func SysToUTF8( const s :Str) :Str
14532:  Func ConsoleToUTF8( const s :Str) :Str
14533:  Func UTF8ToConsole( const s :Str) :Str}
14534:  Func FileExistsUTF8( const Filename :Str) :Bool
14535:  Func FileAgeUTF8( const FileName :Str) : Longint
14536:  Func DirectoryExistsUTF8( const Directory :Str) :Bool
14537:  Func ExpandFileNameUTF8( const FileName :Str) :Str
14538:  Func ExpandUNCFileNameUTF8( const FileName :Str) :Str
14539:  Func ExtractShortPathNameUTF8( const FileName :Str) :Str
14540:  Func FindFirstUTF8(const Path:Str; Attr : Longint; out Rslt : TSearchRec) : Longint
14541:  Func FindNextUTF8( var Rslt : TSearchRec) : Longint
14542:  Proc FindCloseUTF8( var F : TSearchrec)
14543:  Func FileSetDateUTF8( const FileName :Str; Age : Longint) : Longint
14544:  Func FileGetAttrUTF8( const FileName :Str) : Longint
14545:  Func FileSetAttrUTF8( const Filename :Str; Attr : longint) : Longint
14546:  Func DeleteFileUTF8( const FileName :Str) :Bool
14547:  Func RenameFileUTF8( const OldName, NewName :Str) :Bool
14548:  Func FileSearchUTF8(const Name, DirList :Str; ImplicitCurrentDir :Bool) :Str
14549:  Func FileIsReadOnlyUTF8( const FileName :Str) :Bool
14550:  Func GetCurrentDirUTF8 :Str
14551:  Func SetCurrentDirUTF8( const NewDir :Str) :Bool
14552:  Func CreateDirUTF8( const NewDir :Str) :Bool
14553:  Func RemoveDirUTF8( const Dir :Str) :Bool
14554:  Func ForceDirectoriesUTF8( const Dir :Str) :Bool
14555:  Func FileOpenUTF8( const FileName :Str; Mode : Int) : THandle
14556:  Func FileCreateUTF8( const FileName :Str) : THandle;
14557:  Func FileCreateUTF81( const FileName :Str; Rights :Card) : THandle;
14558:  Func ParamStrUTF8( Param : Int) :Str
14559:  Func GetEnvironmentStringUTF8( Index : Int) :Str
14560:  Func GetEnvironmentVariableUTF8( const EnvVar :Str) :Str
14561:  Func GetAppConfigDirUTF8( Global :Bool; Create :Bool) :Str
14562:  Func GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir :Bool) :Str
14563:  Func SysErrorMessageUTF8( ErrorCode : Int) :Str
14564: end;
14565:
14566: Proc SIRegister_Keyboard(CL: TPSPascalCompiler);
14567: begin
14568:   //VK_F23 = 134;
14569:   //{$EXTERNALSYM VK_F24}
14570:   //VK_F24 = 135;
14571:  TVirtualKeyCode', 'Int
14572:  'VK_MOUSEWHEELUP','Int'(134);
14573:  'VK_MOUSEWHEELDOWN','Int'(135);
14574:  Func glIsKeyDown( c : Char) :Bool;
14575:  Func glIsKeyDown1( vk : TVirtualKeyCode) :Bool;
14576:  Func glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
14577:  Func glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) :Str
14578:  Func glKeyNameToVirtualKeyCode( const keyName :Str) : TVirtualKeyCode
14579:  Func glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
14580:  Proc glKeyboardNotifyWheelMoved( wheelDelta : Int)
14581: end;
14582:
14583: Proc SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14584: begin
14585:   TGLPoint', 'TPoint
14586:   //PGLPoint', '^TGLPoint // will not work
14587:   TGLRect', 'TRect
14588:   //PGLRect', '^TGLRect // will not work
14589:   TDelphiColor', 'TColor
14590:   TGLPicture', 'TPicture
14591:   TGLGraphic', 'TGraphic
14592:   TGLBitmap', 'TBitmap
```

```
14593:   //TGraphicClass', 'class of TGraphic
14594:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14595:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14596:   TGLMouseEvent', 'Proc ( Sender : TObject; Button : TGLMouse'
14597:    +'Button; Shift : TShiftState; X, Y : Int)
14598:   TGLMouseMoveEvent', 'TMouseMoveEvent
14599:   TGLKeyEvent', 'TKeyEvent
14600:   TGLKeyPressEvent', 'TKeyPressEvent
14601:   EGLOSError', 'EWin32Error
14602:   EGLOSError', 'EWin32Error
14603:   EGLOSError', 'EOSError
14604:  'glsAllFilter','string'All // sAllFilter
14605:  Func GLPoint( const x, y : Int) : TGLPoint
14606:  Func GLRGB( const r, g, b : Byte) : TColor
14607:  Func GLColorToRGB( color : TColor) : TColor
14608:  Func GLGetRValue( rgb : DWORD) : Byte
14609:  Func GLGetGValue( rgb : DWORD) : Byte
14610:  Func GLGetBValue( rgb : DWORD) : Byte
14611:  Proc GLInitWinColors
14612:  Func GLRect( const aLeft, aTop, aRight, aBottom : Int) : TGLRect
14613:  Proc GLInflateGLRect( var aRect : TGLRect; dx, dy : Int)
14614:  Proc GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14615:  Proc GLInformationDlg( const msg :Str)
14616:  Func GLQuestionDlg( const msg :Str) :Bool
14617:  Func GLInputDlg( const aCaption, aPrompt, aDefault :Str) :Str
14618:  Func GLSavePictureDialog( var aFileName :Str; const aTitle :Str) :Bool
14619:  Func GLOpenPictureDialog( var aFileName :Str; const aTitle :Str) :Bool
14620:  Func GLApplicationTerminated :Bool
14621:  Proc GLRaiseLastOSError
14622:  Proc GLFreeAndNil( var anObject: TObject)
14623:  Func GLGetDeviceLogicalPixelsX( device :Card) : Int
14624:  Func GLGetCurrentColorDepth : Int
14625:  Func GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Int
14626:  Func GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Int) : Pointer
14627:  Proc GLSleep( length :Card)
14628:  Proc GLQueryPerformanceCounter( var val : Int64)
14629:  Func GLQueryPerformanceFrequency( var val : Int64) :Bool
14630:  Func GLStartPrecisionTimer : Int64
14631:  Func GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14632:  Func GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14633:  Func GLRDTSC : Int64
14634:  Proc GLLoadBitmapFromInstance( ABitmap : TBitmap; AName :Str)
14635:  Func GLOKMessageBox( const Text, Caption :Str) : Int
14636:  Proc GLShowHTMLUrl( Url :Str)
14637:  Proc GLShowCursor( AShow :Bool)
14638:  Proc GLSetCursorPos( AScreenX, AScreenY : Int)
14639:  Proc GLGetCursorPos( var point : TGLPoint)
14640:  Func GLGetScreenWidth : Int
14641:  Func GLGetScreenHeight : Int
14642:  Func GLGetTickCount : int64
14643:  Func RemoveSpaces(const str :Str) :Str;
14644:   TNormalMapSpace','( nmsObject, nmsTangent )
14645:  Proc CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14646:  Proc SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14647:  Proc CalcTangentSpaceLightVectors(Light:TAffineVector;Vertices,Normals,Tangents,
       BiNormals:TAffineVectorList;Colors:TVectorList)
14648:  Func CreateObjectSpaceNormalMap(Width,Height:Int;HiNormals,HiTexCoords:TAffineVectorList):TGLBitmap
14649:  Func CreateTangentSpaceNormalMap(Width, Height:Int;HiNormals, HiTexCoords,LoNormals,LoTexCoords,Tangents,
14650:         BiNormals : TAffineVectorList) : TGLBitmap
14651: end;
14652:
14653: Proc SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14654: begin
14655:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14656:  // PGLStarRecord', '^TGLStarRecord // will not work
14657:  Func StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14658:  Func StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14659:  Func StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14660: end;
14661:
14662: Proc SIRegister_GeometryBB(CL: TPSPascalCompiler);
14663: begin
14664:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14665:   //PAABB', '^TAABB // will not work
14666:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14667:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14668:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14669:  Func AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14670:  Proc AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14671:  Proc SetBB( var c : THmgBoundingBox; const v : TVector)
14672:  Proc SetAABB( var bb : TAABB; const v : TVector)
14673:  Proc BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14674:  Proc AABBTransform( var bb : TAABB; const m : TMatrix)
14675:  Proc AABBScale( var bb : TAABB; const v : TAffineVector)
14676:  Func BBMinX( const c : THmgBoundingBox) : Single
14677:  Func BBMaxX( const c : THmgBoundingBox) : Single
14678:  Func BBMinY( const c : THmgBoundingBox) : Single
14679:  Func BBMaxY( const c : THmgBoundingBox) : Single
14680:  Func BBMinZ( const c : THmgBoundingBox) : Single
```

```
14681:  Func BBMaxZ( const c : THmgBoundingBox) : Single
14682:  Proc AABBInclude( var bb : TAABB; const p : TAffineVector)
14683:  Proc AABBFromSweep(var SweepAABB: TAABB; const Start,Dest:TVector; const Radius:Single)
14684:  Func AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14685:  Func BBToAABB( const aBB : THmgBoundingBox) : TAABB
14686:  Func AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14687:  Func AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14688:  Proc OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14689:  Proc OffsetAABB1( var aabb : TAABB; const delta : TVector);
14690:  Func IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) :Bool;
14691:  Func IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) :Bool
14692:  Func IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) :Bool
14693:  Func IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) :Bool
14694:  Func AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) :Bool
14695:  Func PointInAABB( const p : TAffineVector; const aabb : TAABB) :Bool;
14696:  Func PointInAABB1( const p : TVector; const aabb : TAABB) :Bool;
14697:  Func PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) :Bool
14698:  Func TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) :Bool
14699:  Proc ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14700:  Proc AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14701:  Proc BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14702:  Func BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14703:  Func BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14704:  Func AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14705:  Func BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14706:  Func BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14707:  Func AABBContainsBSphere(const mainAABB:TAABB;const testBSphere:TBSphere): TSpaceContains
14708:  Func PlaneContainsBSphere(const Location,Normal:TAffineVector;const testBSphere:TBSphere):TSpaceContains
14709:  Func FrustumContainsBSphere(const Frustum:TFrustum;const testBSphere:TBSphere): TSpaceContains
14710:  Func FrustumContainsAABB( const Frustum:TFrustum;const testAABB:TAABB): TSpaceContains
14711:  Func ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14712:  Func BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) :Bool
14713:  Proc IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14714:  Func AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14715: end;
14716:
14717: Proc SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14718: begin
14719:  Proc Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14720:  Proc Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14721:  Proc Cylindrical_Cartesian2(const r,theta,z1: single; var x,y,z  single; var ierr : Int);
14722:  Proc Cylindrical_Cartesian3(const r,theta,z1: double; var x,y,z: double; var ierr : Int);
14723:  Proc Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14724:  Proc Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14725:  Proc Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14726:  Proc Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14727:  Proc Spherical_Cartesian2(const r,theta, phi: single;var x,y,z:single;varierr:Int);
14728:  Proc Spherical_Cartesian3(const r,theta, phi:double;var x,y,z:double;var ierr:Int);
14729:  Proc Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14730:  Proc Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14731:  Proc Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14732:  Proc ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14733:  Proc ProlateSpheroidal_Cartesian1(const xi,eta,phi,a: double;var x,y, z : double);
14734:  Proc ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:singl;var x,y,z:single;var ierr:Int);
14735:  Proc ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:doubl;var x,y,z:double;var ierr:Int);
14736:  Proc OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14737:  Proc OblateSpheroidal_Cartesian1(const xi, eta, phi, a : double; var x, y, z : double);
14738:  Proc OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr:Int);
14739:  Proc OblateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr:Int);
14740:  Proc BipolarCylindrical_Cartesian(const u, v, z1, a: single; var x, y, z : single);
14741:  Proc BipolarCylindrical_Cartesian1(const u, v, z1, a: double; var x, y, z: double);
14742:  Proc BipolarCylindrical_Cartesian2(const u,v,z1,a:single;var x,y,z:single;var ierr:Int);
14743:  Proc BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double;var ierr:Int);
14744: end;
14745:
14746: Proc SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14747: begin
14748:  'EPSILON','Single').setExtended( 1e-40);
14749:  'EPSILON2','Single').setExtended( 1e-30);  }
14750:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14751:    +irection : TVector; viewPortRadius:Single;farClippingDistance:Single;frustum:TFrustum;end
14752:  THmgPlane', 'TVector
14753:   TDoubleHmgPlane', 'THomogeneousDblVector
14754:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14755:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14756:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14757:   TSingleArray', 'array of Single
14758:   TTransformations','array [0..15] of Single)
14759:   TPackedRotationMatrix','array [0..2] of Smallint)
14760:   TVertex', 'TAffineVector
14761:   //TVectorGL', 'THomogeneousFltVector
14762:   //TMatrixGL', 'THomogeneousFltMatrix
14763:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14764:  Func glTexPointMake( const s, t : Single) : TTexPoint
14765:  Func glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14766:  Func glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14767:  Proc glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14768:  Proc glSetVector( var v : TAffineVector; const x, y, z : Single);
```

```
14769:  Proc glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14770:  Proc glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14771:  Proc glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14772:  Proc glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14773:  Func glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14774:  Func glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14775:  Func glPointMake( const x, y, z : Single) : TVectorGL;
14776:  Func glPointMake1( const v : TAffineVector) : TVectorGL;
14777:  Func glPointMake2( const v : TVectorGL) : TVectorGL;
14778:  Proc glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14779:  Proc glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14780:  Proc glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14781:  Proc glMakePoint( var v : TVectorGL; const x, y, z : Single);
14782:  Proc glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14783:  Proc glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14784:  Proc glMakeVector( var v : TAffineVector; const x, y, z : Single);
14785:  Proc glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14786:  Proc glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14787:  Proc glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14788:  Proc glRstVector( var v : TAffineVector);
14789:  Proc glRstVector1( var v : TVectorGL);
14790:  Func glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14791:  Proc glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14792:  //Proc VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14793:  Func glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14794:  Proc glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14795:  Func glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14796:  Func glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14797:  Proc glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14798:  Proc glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14799:  Proc glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14800:  Proc glAddVector10( var v : TAffineVector; const f : Single);
14801:  Proc glAddVector11( var v : TVectorGL; const f : Single);
14802:  //Proc TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Int;dest:PTexPointArray);
14803:  //Proc TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Int;const scale:
        TTexPoint; dest : PTexPointArray);
14804:  //Proc VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Int;dest:
        PAffineVectorArray);
14805:  Func glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14806:  Proc glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14807:  Proc glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14808:  Proc glVectorSubtract3(const v1: TVectorGL; v2: TAffineVector; var result : TVectorGL);
14809:  Func glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14810:  Proc glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14811:  Proc glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14812:  Func glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14813:  Func glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14814:  Proc glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14815:  Proc glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14816:  Proc glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14817:  //Proc CombineVector1(var vr: TAffineVector; const v : TAffineVector; pf : PFloat);
14818:  Func glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14819:  Func glVectorCombine2(const V1,V2: TAffineVector; const F1, F2 : Single) : TAffineVector;
14820:  Func glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14821:  Proc glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14822:  Proc glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14823:  Proc glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14824:  Func glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14825:  Func glVectorCombine8(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single): TVectorGL;
14826:  Proc glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14827:  Proc glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14828:  Proc glVectorCombine11(const V1,V2 : TVectorGL; const F2 : Single; var vr: TVectorGL);
14829:  Func glVectorCombine3(const V1,V2,V3: TVectorGL; const F1, F2, F3 : Single): TVectorGL;
14830:  Proc glVectorCombine31(const V1, V2, V3:TVectorGL; const F1,F2, F3:Single; var vr : TVectorGL);
14831:  Func glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14832:  Func glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14833:  Func glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14834:  Func glPointProject( const p, origin, direction : TAffineVector) : Single;
14835:  Func glPointProject1( const p, origin, direction : TVectorGL) : Single;
14836:  Func glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14837:  Func glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14838:  Proc glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14839:  Proc glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14840:  Proc glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14841:  Proc glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14842:  Func glLerp( const start, stop, t : Single) : Single
14843:  Func glAngleLerp( start, stop, t : Single) : Single
14844:  Func glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14845:  Func glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14846:  Func glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14847:  Proc glVectorLerp1(const v1,v2: TAffineVector; t : Single; var vr : TAffineVector);
14848:  Func glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14849:  Proc glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14850:  Func glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14851:  Func glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14852: //Proc VectorArrayLerp(const src1,src2:PVectorArray; t:Single; n:Int; dest:PVectorArray);
14853: //Proc VectorArrayLerp1(const src1,src2:PAffineVectorArray; t:Single;n:Int;dest: PAffineVectorArray);
14854:  Func glVectorLength( const x, y : Single) : Single;
14855:  Func glVectorLength1( const x, y, z : Single) : Single;
```

```
14856:  Func glVectorLength2( const v : TAffineVector) : Single;
14857:  Func glVectorLength3( const v : TVectorGL) : Single;
14858:  Func glVectorLength4( const v : array of Single) : Single;
14859:  Func glVectorNorm( const x, y : Single) : Single;
14860:  Func glVectorNorm1( const v : TAffineVector) : Single;
14861:  Func glVectorNorm2( const v : TVectorGL) : Single;
14862:  Func glVectorNorm3( var V : array of Single) : Single;
14863:  Proc glNormalizeVector( var v : TAffineVector);
14864:  Proc glNormalizeVector1( var v : TVectorGL);
14865:  Func glVectorNormalize( const v : TAffineVector) : TAffineVector;
14866:  Func glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14867: // Proc NormalizeVectorArray( list : PAffineVectorArray; n : Int);
14868:  Func glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14869:  Func glVectorNegate( const v : TAffineVector) : TAffineVector;
14870:  Func glVectorNegate1( const v : TVectorGL) : TVectorGL;
14871:  Proc glNegateVector( var V : TAffineVector);
14872:  Proc glNegateVector2( var V : TVectorGL);
14873:  Proc glNegateVector3( var V : array of Single);
14874:  Proc glScaleVector( var v : TAffineVector; factor : Single);
14875:  Proc glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14876:  Proc glScaleVector2( var v : TVectorGL; factor : Single);
14877:  Proc glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14878:  Func glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14879:  Proc glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14880:  Func glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14881:  Proc glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14882:  Proc glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14883:  Proc glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14884:  Func glVectorEquals( const V1, V2 : TVectorGL) :Bool;
14885:  Func glVectorEquals1( const V1, V2 : TAffineVector) :Bool;
14886:  Func glAffineVectorEquals( const V1, V2 : TVectorGL) :Bool;
14887:  Func glVectorIsNull( const v : TVectorGL) :Bool;
14888:  Func glVectorIsNull1( const v : TAffineVector) :Bool;
14889:  Func glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14890:  Func glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14891:  Func glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14892:  Func glVectorDistance( const v1, v2 : TAffineVector) : Single;
14893:  Func glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14894:  Func glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14895:  Func glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14896:  Func glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14897:  Func glVectorReflect( const V, N : TAffineVector) : TAffineVector
14898:  Proc glRotateVector(var vector:TVectorGL;const axis: TAffineVector; angle : Single);
14899:  Proc glRotateVector1( var vector:TVectorGL;const axis : TVectorGL; angle : Single);
14900:  Proc glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14901:  Func glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14902:  Func glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14903:  Proc glVectorRotateAroundY1(const v:TAffineVector;alpha:Single; var vr : TAffineVector);
14904:  Func glVectorRotateAroundZ(const v : TAffineVector; alpha : Single) : TAffineVector;
14905:  Proc glAbsVector( var v : TVectorGL);
14906:  Proc glAbsVector1( var v : TAffineVector);
14907:  Func glVectorAbs( const v : TVectorGL) : TVectorGL;
14908:  Func glVectorAbs1( const v : TAffineVector) : TAffineVector;
14909:  Proc glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14910:  Proc glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14911:  Proc glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14912:  Proc glSetMatrixRow( var dest : TMatrixGL; rowNb : Int; const aRow : TVectorGL);
14913:  Func glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14914:  Func glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14915:  Func glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14916:  Func glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14917:  Func glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14918:  Func glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14919:  Func glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14920:  Func glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14921:  Func glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14922:  Func glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14923:  Func glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14924:  Func glCreateRotationMatrix(const anAxis:TAffineVector; angle: Single): TMatrixGL;
14925:  Func glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14926:  Func glCreateAffineRotationMatrix(const anAxis:TAffineVector;angle:Single):TAffineMatrix
14927:  Func glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14928:  Func glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14929:  Proc glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14930:  Func glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14931:  Func glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14932:  Func glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14933:  Func glVectorTransform3(const V : TAffineVector; const M : TAffineMatrix):TAffineVector;
14934:  Func glMatrixDeterminant( const M : TAffineMatrix) : Single;
14935:  Func glMatrixDeterminant1( const M : TMatrixGL) : Single;
14936:  Proc glAdjointMatrix( var M : TMatrixGL);
14937:  Proc glAdjointMatrix1( var M : TAffineMatrix);
14938:  Proc glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14939:  Proc glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14940:  Proc glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14941:  Proc glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14942:  Proc glNormalizeMatrix( var M : TMatrixGL)
14943:  Proc glTransposeMatrix( var M : TAffineMatrix);
14944:  Proc glTransposeMatrix1( var M : TMatrixGL);
```

```
14945:  Proc glInvertMatrix( var M : TMatrixGL);
14946:  Proc glInvertMatrix1( var M : TAffineMatrix);
14947:  Func glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14948:  Func glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) :Bool
14949:  Func glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14950:  Func glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14951:  Func glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14952:  Func glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14953:  Proc glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14954:  Proc glNormalizePlane( var plane : THmgPlane)
14955:  Func glPlaneEvaluatePoint(const plane:THmgPlane;const point: TAffineVector): Single;
14956:  Func glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14957:  Func glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14958:  Proc glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14959:  Proc glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14960:  Func glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) :Bool;
14961:  Func glPointIsInHalfSpace1(const point,planePoint,planeNormal : TAffineVector) :Bool;
14962:  Func glPointPlaneDistance(const point, planePoint, planeNormal: TVectorGL) : Single;
14963:  Func glPointPlaneDistance1(const point, planePoint, planeNormal: TAffineVector): Single;
14964:  Func glPointSegmentClosestPoint(const point,segmentStart, segmentStop:TAffineVector):TAffineVector
14965:  Func glPointSegmentDistance(const point,segmentStart,segmentStop:TAffineVector): single
14966:  Func glPointLineClosestPoint(const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14967:  Func glPointLineDistance(const point,linePoint,lineDirection:TAffineVector): Single
14968:  Proc SglegmentSegmentClosestPoint(const S0Start,S0Stop,S1Start,S1Stop TAffineVector;
14969:                                      var Segment0Closest,Segment1Closest: TAffineVector)
14970:  Func glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14971:                                    TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14972:  Func glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14973:  Func glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14974:  Func glQuaternionMagnitude( const Q : TQuaternion) : Single
14975:  Proc glNormalizeQuaternion( var Q : TQuaternion)
14976:  Func glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14977:  Proc glQuaternionToPoints(const Q: TQuaternion; var ArcFrom,ArcTo : TAffineVector)
14978:  Func glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14979:  Func glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14980:  Func glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14981:  Func glQuaternionFromAngleAxis(const angle:Single;const axis:TAffineVector) : TQuaternion
14982:  Func glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14983:  Func glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14984:  Func glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14985:  Func glQuaternionSlerp(const QStart,QEnd:TQuaternion;Spin:Int; t: Single): TQuaternion;
14986:  Func glQuaternionSlerp1(const source, dest:TQuaternion; const t: Single) : TQuaternion;
14987:  Func glLnXP1( X : Extended) : Extended
14988:  Func glLog10( X : Extended) : Extended
14989:  Func glLog2( X : Extended) : Extended;
14990:  Func glLog21( X : Single) : Single;
14991:  Func glLogN( Base, X : Extended) : Extended
14992:  Func glIntPower( Base : Extended; Exponent : Int) : Extended
14993:  Func glPower( const Base, Exponent : Single) : Single;
14994:  Func glPower1( Base : Single; Exponent : Int) : Single;
14995:  Func glDegToRad( const Degrees : Extended) : Extended;
14996:  Func glDegToRad1( const Degrees : Single) : Single;
14997:  Func glRadToDeg( const Radians : Extended) : Extended;
14998:  Func glRadToDeg1( const Radians : Single) : Single;
14999:  Func glNormalizeAngle( angle : Single) : Single
15000:  Func glNormalizeDegAngle( angle : Single) : Single
15001:  Proc glSinCos( const Theta : Extended; var Sin, Cos : Extended);
15002:  Proc glSinCos11( const Theta : Double; var Sin, Cos : Double);
15003:  Proc glSinCos0( const Theta : Single; var Sin, Cos : Single);
15004:  Proc glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
15005:  Proc glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
15006:  Proc glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
15007:  Proc glPrepareSinCosCache(var s,c: array of Single; startAngle,stopAngle : Single)
15008:  Func glArcCos( const X : Extended) : Extended;
15009:  Func glArcCos1( const x : Single) : Single;
15010:  Func glArcSin( const X : Extended) : Extended;
15011:  Func glArcSin1( const X : Single) : Single;
15012:  Func glArcTan21( const Y, X : Extended) : Extended;
15013:  Func glArcTan21( const Y, X : Single) : Single;
15014:  Func glFastArcTan2( y, x : Single) : Single
15015:  Func glTan( const X : Extended) : Extended;
15016:  Func glTan1( const X : Single) : Single;
15017:  Func glCoTan( const X : Extended) : Extended;
15018:  Func glCoTan1( const X : Single) : Single;
15019:  Func glSinh( const x : Single) : Single;
15020:  Func glSinh1( const x : Double) : Double;
15021:  Func glCosh( const x : Single) : Single;
15022:  Func glCosh1( const x : Double) : Double;
15023:  Func glRSqrt( v : Single) : Single
15024:  Func glRLength( x, y : Single) : Single
15025:  Func glISqrt( i : Int) : Int
15026:  Func glILength( x, y : Int) : Int;
15027:  Func glILength1( x, y, z : Int) : Int;
15028:  Proc glRegisterBasedExp
15029:  Proc glRandomPointOnSphere( var p : TAffineVector)
15030:  Func glRoundInt( v : Single) : Single;
15031:  Func glRoundInt1( v : Extended) : Extended;
15032:  Func glTrunc( v : Single) : Int;
15033:  Func glTrunc64( v : Extended) : Int64;
```

```
15034:  Func glInt( v : Single) : Single;
15035:  Func glInt1( v : Extended) : Extended;
15036:  Func glFrac( v : Single) : Single;
15037:  Func glFrac1( v : Extended) : Extended;
15038:  Func glRound( v : Single) : Int;
15039:  Func glRound64( v : Single) : Int64;
15040:  Func glRound641( v : Extended) : Int64;
15041:  Func glTrunc( X : Extended) : Int64
15042:  Func glRound( X : Extended) : Int64
15043:  Func glFrac( X : Extended) : Extended
15044:  Func glCeil( v : Single) : Int;
15045:  Func glCeil64( v : Extended) : Int64;
15046:  Func glFloor( v : Single) : Int;
15047:  Func glFloor64( v : Extended) : Int64;
15048:  Func glScaleAndRound( i : Int; var s : Single) : Int
15049:  Func glSign( x : Single) : Int
15050:  Func glIsInRange( const x, a, b : Single) :Bool;
15051:  Func glIsInRange1( const x, a, b : Double) :Bool;
15052:  Func glIsInCube( const p, d : TAffineVector) :Bool;
15053:  Func glIsInCube1( const p, d : TVectorGL) :Bool;
15054:  //Func MinFloat( values : PSingleArray; nbItems : Int) : Single;
15055:  //Func MinFloat1( values : PDoubleArray; nbItems : Int) : Double;
15056:  //Func MinFloat2( values : PExtendedArray; nbItems : Int) : Extended;
15057:  Func glMinFloat3( const v1, v2 : Single) : Single;
15058:  Func glMinFloat4( const v : array of Single) : Single;
15059:  Func glMinFloat5( const v1, v2 : Double) : Double;
15060:  Func glMinFloat6( const v1, v2 : Extended) : Extended;
15061:  Func glMinFloat7( const v1, v2, v3 : Single) : Single;
15062:  Func glMinFloat8( const v1, v2, v3 : Double) : Double;
15063:  Func glMinFloat9( const v1, v2, v3 : Extended) : Extended;
15064:  //Func MaxFloat10( values : PSingleArray; nbItems : Int) : Single;
15065:  //Func MaxFloat( values : PDoubleArray; nbItems : Int) : Double;
15066:  //Func MaxFloat1( values : PExtendedArray; nbItems : Int) : Extended;
15067:  Func glMaxFloat2( const v : array of Single) : Single;
15068:  Func glMaxFloat3( const v1, v2 : Single) : Single;
15069:  Func glMaxFloat4( const v1, v2 : Double) : Double;
15070:  Func glMaxFloat5( const v1, v2 : Extended) : Extended;
15071:  Func glMaxFloat6( const v1, v2, v3 : Single) : Single;
15072:  Func glMaxFloat7( const v1, v2, v3 : Double) : Double;
15073:  Func glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
15074:  Func glMinInt9( const v1, v2 : Int) : Int;
15075:  Func glMinInt( const v1, v2 :Card) :Card;
15076:  Func glMaxInt( const v1, v2 : Int) : Int;
15077:  Func glMaxInt1( const v1, v2 :Card) :Card;
15078:  Func glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
15079:  //Func PolygonArea( const p : PAffineVectorArray; nSides : Int) : Single;
15080:  Func glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
15081:  //Func PolygonSignedArea( const p : PAffineVectorArray; nSides : Int) : Single;
15082:  //Proc ScaleFloatArray( values : PSingleArray; nb : Int; var factor : Single);
15083:  Proc glScaleFloatArray( var values : TSingleArray; factor : Single);
15084:  //Proc OffsetFloatArray( values : PSingleArray; nb : Int; var delta : Single);
15085:  Proc glOffsetFloatArray1( var values : array of Single; delta : Single);
15086:  //Proc OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Int);
15087:  Func glMaxXYZComponent( const v : TVectorGL) : Single;
15088:  Func glMaxXYZComponent1( const v : TAffineVector) : single;
15089:  Func glMinXYZComponent( const v : TVectorGL) : Single;
15090:  Func glMinXYZComponent1( const v : TAffineVector) : single;
15091:  Func glMaxAbsXYZComponent( v : TVectorGL) : Single
15092:  Func glMinAbsXYZComponent( v : TVectorGL) : Single
15093:  Proc glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
15094:  Proc glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
15095:  Proc glMinVector( var v : TVectorGL; const v1 : TVectorGL);
15096:  Proc glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
15097:  Proc glSortArrayAscending( var a : array of Extended)
15098:  Func glClampValue( const aValue, aMin, aMax : Single) : Single;
15099:  Func glClampValue1( const aValue, aMin : Single) : Single;
15100:  Func glGeometryOptimizationMode :Str
15101:  Proc glBeginFPUOnlySection
15102:  Proc glEndFPUOnlySection
15103:  Func glConvertRotation( const Angles : TAffineVector) : TVectorGL
15104:  Func glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
15105:  Func glMakeDblVector( var v : array of Double) : THomogeneousDblVector
15106:  Func glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
15107:  Func glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
15108:  Func glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
15109:  Func glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
15110:  Func glPointInPolygon( var xp, yp : array of Single; x, y : Single) :Bool
15111:  Proc glDivMod( Dividend : Int; Divisor : Word; var Result, Remainder : Word)
15112:  Func glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
15113:  Func glTurn1(const Matrix:TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
15114:  Func glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
15115:  Func glPitch1(const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
15116:  Func glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
15117:  Func glRoll1(const Matrix:TMatrixGL;const MasterDirection:TAffineVector;Angle:Single): TMatrixGL;
15118:  Func glRayCastMinDistToPoint(const rayStart,rayVector:TVectorGL;const point:TVectorGL):Single
15119:  Func glRayCastIntersectsSphere(const rayStart,rayVector:TVectorGL;const sphereCenter:TVectorGL;const
        sphereRadius:Single):Bool;
15120:  Func glRayCastSphereIntersect(const rayStart,rayVector:TVectorGL;const sphereCenter:TVectorGL;const
        sphereRadius:Single;var i1,i2:TVectorGL):Int;
```

```
15121:  Func glSphereVisibleRadius( distance, radius : Single) : Single
15122:  Func glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
15123:  Func glIsVolumeClipped(const objPos:TVectorGL;const objRadius:Single;const
        rcci:TRenderContextClippingInfo):Bool;
15124:  Func glIsVolumeClipped1(const objPos:TAffineVector;const objRadius:Single;const
        rcci:TRenderContextClippingInfo):Bool;
15125:  Func glIsVolumeClipped2(const min,max:TAffineVector;const rcci:TRenderContextClippingInfo):Bool;
15126:  Func glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const Frustum:TFrustum):Bool;
15127:  Func glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
15128:  Func glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
15129:  Func glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
15130:  Func glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
15131:  Func glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
15132:  'cPI','Single').setExtended( 3.141592654);
15133:  'cPIdiv180','Single').setExtended( 0.017453292);
15134:  'c180divPI','Single').setExtended( 57.29577951);
15135:  'c2PI','Single').setExtended( 6.283185307);
15136:  'cPIdiv2','Single').setExtended( 1.570796326);
15137:  'cPIdiv4','Single').setExtended( 0.785398163);
15138:  'c3PIdiv4','Single').setExtended( 2.35619449);
15139:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
15140:  'cInv360','Single').setExtended( 1 / 360);
15141:  'c180','Single').setExtended( 180);
15142:  'c360','Single').setExtended( 360);
15143:  'cOneHalf','Single').setExtended( 0.5);
15144:  'cLn10','Single').setExtended( 2.302585093);
15145:  {'MinSingle','Extended').setExtended( 1.5e-45);
15146:  'MaxSingle','Extended').setExtended( 3.4e+38);
15147:  'MinDouble','Extended').setExtended( 5.0e-324);
15148:  'MaxDouble','Extended').setExtended( 1.7e+308);
15149:  'MinExtended','Extended').setExtended( 3.4e-4932);
15150:  'MaxExtended','Extended').setExtended( 1.1e+4932);
15151:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
15152:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
15153: end;
15154:
15155: Proc SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
15156: begin
15157:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
15158:   (FindClass('TOBJECT'),'TFaceGroups
15159:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
15160:   TMeshAutoCenterings', 'set of TMeshAutoCentering
15161:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
15162:   SIRegister_TBaseMeshObject(CL);
15163:   (FindClass('TOBJECT'),'TSkeletonFrameList
15164:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
15165:   SIRegister_TSkeletonFrame(CL);
15166:   SIRegister_TSkeletonFrameList(CL);
15167:   (FindClass('TOBJECT'),'TSkeleton
15168:   (FindClass('TOBJECT'),'TSkeletonBone
15169:   SIRegister_TSkeletonBoneList(CL);
15170:   SIRegister_TSkeletonRootBoneList(CL);
15171:   SIRegister_TSkeletonBone(CL);
15172:   (FindClass('TOBJECT'),'TSkeletonColliderList
15173:   SIRegister_TSkeletonCollider(CL);
15174:   SIRegister_TSkeletonColliderList(CL);
15175:   (FindClass('TOBJECT'),'TGLBaseMesh
15176:   TBlendedLerpInfo', 'record frameIndex1 : Int; frameIndex2 : '
15177:    +'Int; lerpFactor : Single; weight : Single; externalPositions : TAffine'
15178:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
15179:    +'QuaternionList; end
15180:   SIRegister_TSkeleton(CL);
15181:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
15182:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
15183:   SIRegister_TMeshObject(CL);
15184:   SIRegister_TMeshObjectList(CL);
15185:   //TMeshObjectListClass', 'class of TMeshObjectList
15186:   (FindClass('TOBJECT'),'TMeshMorphTargetList
15187:   SIRegister_TMeshMorphTarget(CL);
15188:   SIRegister_TMeshMorphTargetList(CL);
15189:   SIRegister_TMorphableMeshObject(CL);
15190:   TVertexBoneWeight', 'record BoneID : Int; Weight : Single; end
15191:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
15192:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
15193:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
15194:   SIRegister_TSkeletonMeshObject(CL);
15195:   SIRegister_TFaceGroup(CL);
15196:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
15197:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
15198:   SIRegister_TFGVertexIndexList(CL);
15199:   SIRegister_TFGVertexNormalTexIndexList(CL);
15200:   SIRegister_TFGIndexTexCoordList(CL);
15201:   SIRegister_TFaceGroups(CL);
15202:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
15203:   SIRegister_TVectorFile(CL);
15204:   //TVectorFileClass', 'class of TVectorFile
15205:   SIRegister_TGLGLSMVectorFile(CL);
15206:   SIRegister_TGLBaseMesh(CL);
15207:   SIRegister_TGLFreeForm(CL);
```

```
15208:   TGLActorOption', '( aoSkeletonNormalizeNormals )
15209:   TGLActorOptions', 'set of TGLActorOption
15210:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
15211:   (FindClass('TOBJECT'),'TGLActor
15212:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
15213:   SIRegister_TActorAnimation(CL);
15214:   TActorAnimationName', 'String
15215:   SIRegister_TActorAnimations(CL);
15216:   SIRegister_TGLBaseAnimationControler(CL);
15217:   SIRegister_TGLAnimationControler(CL);
15218:   TActorFrameInterpolation', '( afpNone, afpLinear )
15219:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
15220:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
15221:   SIRegister_TGLActor(CL);
15222:   SIRegister_TVectorFileFormat(CL);
15223:   SIRegister_TVectorFileFormatsList(CL);
15224:   (FindClass('TOBJECT'),'EInvalidVectorFile
15225:  Func GetVectorFileFormats : TVectorFileFormatsList
15226:  Func VectorFileFormatsFilter :Str
15227:  Func VectorFileFormatsSaveFilter :Str
15228:  Func VectorFileFormatExtensionByIndex( index : Int) :Str
15229:  Proc RegisterVectorFileFormat(const aExtension,aDescription:str;aClass: TVectorFileClass)
15230:  Proc UnregisterVectorFileClass( aClass : TVectorFileClass)
15231: end;
15232:
15233: Proc SIRegister_AxCtrls(CL: TPSPascalCompiler);
15234: begin
15235:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
15236:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
15237:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
15238:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
15239:   SIRegister_TOleStream(CL);
15240:   (FindClass('TOBJECT'),'TConnectionPoints
15241:   TConnectionKind', '( ckSingle, ckMulti )
15242:   SIRegister_TConnectionPoint(CL);
15243:   SIRegister_TConnectionPoints(CL);
15244:   TDefinePropertyPage', 'Proc ( const GUID : TGUID)
15245:   (FindClass('TOBJECT'),'TActiveXControlFactory
15246:   SIRegister_TActiveXControl(CL);
15247:   //TActiveXControlClass', 'class of TActiveXControl
15248:   SIRegister_TActiveXControlFactory(CL);
15249:   SIRegister_TActiveFormControl(CL);
15250:   SIRegister_TActiveForm(CL);
15251:   //TActiveFormClass', 'class of TActiveForm
15252:   SIRegister_TActiveFormFactory(CL);
15253:   (FindClass('TOBJECT'),'TPropertyPageImpl
15254:   SIRegister_TPropertyPage(CL);
15255:   //TPropertyPageClass', 'class of TPropertyPage
15256:   SIRegister_TPropertyPageImpl(CL);
15257:   SIRegister_TActiveXPropertyPage(CL);
15258:   SIRegister_TActiveXPropertyPageFactory(CL);
15259:   SIRegister_TCustomAdapter(CL);
15260:   SIRegister_TAdapterNotifier(CL);
15261:   SIRegister_IFontAccess(CL);
15262:   SIRegister_TFontAdapter(CL);
15263:   SIRegister_IPictureAccess(CL);
15264:   SIRegister_TPictureAdapter(CL);
15265:   SIRegister_TOleGraphic(CL);
15266:   SIRegister_TStringsAdapter(CL);
15267:   SIRegister_TReflectorWindow(CL);
15268:  Proc EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
15269:  Proc GetOleFont( Font : TFont; var OleFont : IFontDisp)
15270:  Proc SetOleFont( Font : TFont; OleFont : IFontDisp)
15271:  Proc GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
15272:  Proc SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
15273:  Proc GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
15274:  Proc SetOleStrings( Strings : TStrings; OleStrings : IStrings)
15275:  Func ParkingWindow : HWND
15276: end;
15277:
15278: Proc SIRegister_synaip(CL: TPSPascalCompiler);
15279: begin
15280:  // TIp6Bytes = array [0..15] of Byte;
15281: {:binary form of IPv6 adress (for string conversion routines)}
15282:  // TIp6Words = array [0..7] of Word;
15283:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;;
15284:   AddTypeS('TIp6Words', 'array [0..7] of Word;;
15285:   AddTypeS(TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end;
15286:  Func synaIsIP( const Value :Str) :Bool;
15287:  Func synaIsIP6( const Value :Str) :Bool;
15288:  Func synaIPToID( Host :Str) : Ansistr;
15289:  Func synaStrToIp6( value :Str) : TIp6Bytes;
15290:  Func synaIp6ToStr( value : TIp6Bytes) :Str;
15291:  Func synaStrToIp( value :Str) : Int;
15292:  Func synaIpToStr( value : Int) :Str;
15293:  Func synaReverseIP( Value : Ansistr) : Ansistr;
15294:  Func synaReverseIP6( Value : Ansistr) : Ansistr;
15295:  Func synaExpandIP6( Value : Ansistr) : Ansistr;
15296:  Func xStrToIP( const Value :Str) : TIPAdr;
```

```
15297:  Func xIPToStr( const Adresse : TIPAdr) :Str;
15298:  Func IPToCardinal( const Adresse : TIPAdr) :Card;
15299:  Func CardinalToIP( const Value :Card) : TIPAdr;
15300:  Func ReverseIP( Value : Ansistr) : Ansistr');
15301:  Func ReverseIP6( Value : Ansistr) : Ansistr');
15302: end;
15303:
15304: Proc SIRegister_synacode(CL: TPSPascalCompiler);
15305: begin
15306:  AddTypeS('TSpecials', 'set of Char;
15307:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_;
15308:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+;
15309:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=;
15310:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=;
15311:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_;
15312:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;
15313:  Func DecodeTriplet( const Value : Ansistr; Delimiter : Char) : Ansistr;
15314:  Func DecodeQuotedPrintable( const Value : Ansistr) : Ansistr;
15315:  Func DecodeURL( const Value : Ansistr) : Ansistr;
15316:  Func EncodeTriplet(const Value:Ansistr;Delimiter:Char;Specials:TSpecials):Ansistr);
15317:  Func EncodeQuotedPrintable( const Value : Ansistr) : Ansistr;
15318:  Func EncodeSafeQuotedPrintable( const Value : Ansistr) : Ansistr;
15319:  Func EncodeURLElement( const Value : Ansistr) : Ansistr;
15320:  Func EncodeURL( const Value : Ansistr) : Ansistr;
15321:  Func Decode4to3( const Value, Table : Ansistr) : Ansistr;
15322:  Func Decode4to3Ex( const Value, Table : Ansistr) : Ansistr;
15323:  Func Encode3to4( const Value, Table : Ansistr) : Ansistr;
15324:  Func synDecodeBase64( const Value : Ansistr) : Ansistr;
15325:  Func synEncodeBase64( const Value : Ansistr) : Ansistr;
15326:  Func DecodeBase64mod( const Value : Ansistr) : Ansistr;
15327:  Func EncodeBase64mod( const Value : Ansistr) : Ansistr;
15328:  Func DecodeUU( const Value : Ansistr) : Ansistr;
15329:  Func EncodeUU( const Value : Ansistr) : Ansistr;
15330:  Func DecodeXX( const Value : Ansistr) : Ansistr;
15331:  Func DecodeYEnc( const Value : Ansistr) : Ansistr;
15332:  Func UpdateCrc32( Value : Byte; Crc32 : Int) : Int;
15333:  Func synCrc32( const Value : Ansistr) : Int;
15334:  Func UpdateCrc16( Value : Byte; Crc16 : Word) : Word;
15335:  Func Crc16( const Value : Ansistr) : Word;
15336:  Func synMD5( const Value : Ansistr) : Ansistr;
15337:  Func HMAC_MD5( Text, Key : Ansistr) : Ansistr;
15338:  Func MD5LongHash( const Value : Ansistr; Len : Int) : Ansistr;
15339:  Func synSHA1( const Value : Ansistr) : Ansistr;
15340:  Func HMAC_SHA1( Text, Key : Ansistr) : Ansistr;
15341:  Func SHA1LongHash( const Value : Ansistr; Len : Int) : Ansistr;
15342:  Func synMD4( const Value : Ansistr) : Ansistr;
15343: end;
15344:
15345: Proc SIRegister_synachar(CL: TPSPascalCompiler);
15346: begin
15347:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
15348:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
15349:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
15350:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
15351:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
15352:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
15353:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
15354:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
15355:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
15356:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
15357:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
15358:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
15359:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
15360:    +', CP864, CP865, CP869, CP1125 );
15361:   AddTypeS('TMimeSetChar', 'set of TMimeChar;
15362:  Func CharsetConversion(const Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiStr;
15363:  Func CharsetConversionEx(const Value:Ansistr;CharFrom:TMimeChar;CharTo:TMimeChar;const
        TransformTable:array of Word):Ansistr;
15364:  Func CharsetConversionTrans(Value:Ansistr;CharFrom:TMimeChar;CharTo:TMimeChar;const TransformTable:array
        of Word;Translit:Boolean):Ansistr;
15365:  Func GetCurCP : TMimeChar;
15366:  Func GetCurOEMCP : TMimeChar;
15367:  Func GetCPFromID( Value : Ansistr) : TMimeChar;
15368:  Func GetIDFromCP( Value : TMimeChar) : Ansistr;
15369:  Func NeedCharsetConversion( const Value : Ansistr) :Bool;
15370:  Func IdealCharsetCoding(const Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
15371:  Func GetBOM( Value : TMimeChar) : Ansistr;
15372:  Func StringToWide( const Value : Ansistr) : WideString;
15373:  Func WideToString( const Value : WideString) : Ansistr;
15374: end;
15375:
15376: Proc SIRegister_synamisc(CL: TPSPascalCompiler);
15377: begin
15378:   AddTypeS('TProxySetting', 'record Host :Str; Port :Str; Bypass :Str; end;
15379:  Proc WakeOnLan( MAC, IP :Str);
15380:  Func GetDNS :Str;
15381:  Func GetIEProxy( protocol :Str) : TProxySetting;
15382:  Func GetLocalIPs :Str;
15383: end;
```

```
15384:
15385: Proc SIRegister_synaser(CL: TPSPascalCompiler);
15386: begin
15387:  AddConstantN('synCR','Char #$0d);
15388:  Const('synLF','Char #$0a);
15389:  Const('cSerialChunk','LongInt'( 8192);
15390:  Const('LockfileDirectory','String '/var/lock;
15391:  Const('PortIsClosed','LongInt'( - 1);
15392:  Const('ErrAlreadyOwned','LongInt'( 9991);
15393:  Const('ErrAlreadyInUse','LongInt'( 9992);
15394:  Const('ErrWrongParameter','LongInt'( 9993);
15395:  Const('ErrPortNotOpen','LongInt'( 9994);
15396:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
15397:  Const('ErrMaxBuffer','LongInt'( 9996);
15398:  Const('ErrTimeout','LongInt'( 9997);
15399:  Const('ErrNotRead','LongInt'( 9998);
15400:  Const('ErrFrame','LongInt'( 9999);
15401:  Const('ErrOverrun','LongInt'( 10000);
15402:  Const('ErrRxOver','LongInt'( 10001);
15403:  Const('ErrRxParity','LongInt'( 10002);
15404:  Const('ErrTxFull','LongInt'( 10003);
15405:  Const('dcb_Binary','LongWord')( $00000001);
15406:  Const('dcb_ParityCheck','LongWord')( $00000002);
15407:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
15408:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
15409:  Const('dcb_DtrControlMask','LongWord')( $00000030);
15410:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
15411:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
15412:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
15413:  Const('dcb_DsrSensivity','LongWord')( $00000040);
15414:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
15415:  Const('dcb_OutX','LongWord')( $00000100);
15416:  Const('dcb_InX','LongWord')( $00000200);
15417:  Const('dcb_ErrorChar','LongWord')( $00000400);
15418:  Const('dcb_NullStrip','LongWord')( $00000800);
15419:  Const('dcb_RtsControlMask','LongWord')( $00003000);
15420:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
15421:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
15422:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
15423:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
15424:  Const('dcb_AbortOnError','LongWord')( $00004000);
15425:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
15426:  Const('synSB1','LongInt'( 0);
15427:  Const('SB1andHalf','LongInt'( 1);
15428:  Const('synSB2','LongInt'( 2);
15429:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
15430:  Const('CS7fix','LongWord')( $0000020);
15431:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
15432:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
15433:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
15434:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end;
15435:   //AddTypeS('PDCB', '^TDCB // will not work;
15436:  //Const('MaxRates','LongInt'( 18);
15437:  //Const('MaxRates','LongInt'( 30);
15438:  //Const('MaxRates','LongInt'( 19);
15439:  Const('O_SYNC','LongWord')( $0080);
15440:  Const('synOK','LongInt'( 0);
15441:  Const('synErr','LongInt'( Int ( - 1 ));
15442:   AddTypeS('THookSerialReason','(HR_SerialClose,HR_Connect,HR_CanRead,HR_CanWrite,HR_ReadCount,
      HR_WriteCount,HR_Wait);
15443:  Type('THookSerialStatus',Procedure(Sender:TObject;Reason:THookSerialReason;const Value:str);
15444:   SIRegister_ESynaSerError(CL);
15445:   SIRegister_TBlockSerial(CL);
15446:  Func GetSerialPortNames :Str;
15447: end;
15448:
15449: Proc SIRegister_synaicnv(CL: TPSPascalCompiler);
15450: begin
15451:  Const('DLLIconvName','String 'libiconv.so;
15452:  Const('DLLIconvName','String 'iconv.dll;
15453:   AddTypeS('size_t', 'Cardinal;
15454:   AddTypeS('iconv_t', 'Int;
15455:   //AddTypeS('iconv_t', 'Pointer;
15456:   AddTypeS('argptr', 'iconv_t;
15457:  Func SynaIconvOpen( const tocode, fromcode : Ansistr) : iconv_t;
15458:  Func SynaIconvOpenTranslit( const tocode, fromcode : Ansistr) : iconv_t;
15459:  Func SynaIconvOpenIgnore( const tocode, fromcode : Ansistr) : iconv_t;
15460:  Func SynaIconv( cd : iconv_t; inbuf : Ansistr; var outbuf : Ansistr) : Int;
15461:  Func SynaIconvClose( var cd : iconv_t) : Int;
15462:  Func SynaIconvCtl( cd : iconv_t; request : Int; argument : argptr) : Int;
15463:  Func IsIconvloaded :Bool;
15464:  Func InitIconvInterface :Bool;
15465:  Func DestroyIconvInterface :Bool;
15466:  Const('ICONV_TRIVIALP','LongInt'( 0);
15467:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
15468:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
15469:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
15470:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
15471: end;
```

```
15472:
15473: Proc SIRegister_pingsend(CL: TPSPascalCompiler);
15474: begin
15475:  Const 'ICMP_ECHO','LongInt'( 8);
15476:  Const('ICMP_ECHOREPLY','LongInt'( 0);
15477:  Const('ICMP_UNREACH','LongInt'( 3);
15478:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
15479:  Const('ICMP6_ECHO','LongInt'( 128);
15480:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
15481:  Const('ICMP6_UNREACH','LongInt'( 1);
15482:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
15483:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
15484:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort );
15485:   SIRegister_TPINGSend(CL);
15486:  Func PingHost( const Host :Str) : Int;
15487:  Func TraceRouteHost( const Host :Str) :Str;
15488: end;
15489:
15490: Proc SIRegister_asn1util(CL: TPSPascalCompiler);
15491: begin
15492:  AddConstantN('synASN1_BOOL','LongWord')( $01);
15493:  Const('synASN1_INT','LongWord')( $02);
15494:  Const('synASN1_OCTSTR','LongWord')( $04);
15495:  Const('synASN1_NULL','LongWord')( $05);
15496:  Const('synASN1_OBJID','LongWord')( $06);
15497:  Const('synASN1_ENUM','LongWord')( $0a);
15498:  Const('synASN1_SEQ','LongWord')( $30);
15499:  Const('synASN1_SETOF','LongWord')( $31);
15500:  Const('synASN1_IPADDR','LongWord')( $40);
15501:  Const('synASN1_COUNTER','LongWord')( $41);
15502:  Const('synASN1_GAUGE','LongWord')( $42);
15503:  Const('synASN1_TIMETICKS','LongWord')( $43);
15504:  Const('synASN1_OPAQUE','LongWord')( $44);
15505:  Func synASNEncOIDItem( Value : Int) : Ansistr;
15506:  Func synASNDecOIDItem( var Start : Int; const Buffer : Ansistr) : Int;
15507:  Func synASNEncLen( Len : Int) : Ansistr;
15508:  Func synASNDecLen( var Start : Int; const Buffer : Ansistr) : Int;
15509:  Func synASNEncInt( Value : Int) : Ansistr;
15510:  Func synASNEncUInt( Value : Int) : Ansistr;
15511:  Func synASNObject( const Data : Ansistr; ASNType : Int) : Ansistr;
15512:  Func synASNItem(var Start:Int;const Buffer:Ansistr;var ValueType:Int):Ansistr;
15513:  Func synMibToId( Mib :Str) : Ansistr;
15514:  Func synIdToMib( const Id : Ansistr) :Str;
15515:  Func synIntMibToStr( const Value : Ansistr) : Ansistr;
15516:  Func ASNdump( const Value : Ansistr) : Ansistr;
15517:  Func GetMailServers(const DNSHost, Domain : Ansistr; const Servers:TStrings):Bool;
15518:  Func LDAPResultDump( const Value : TLDAPResultList) : Ansistr;
15519: end;
15520:
15521: Proc SIRegister_ldapsend(CL: TPSPascalCompiler);
15522: begin
15523:  Const('cLDAPProtocol','String '389;
15524:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
15525:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
15526:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
15527:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
15528:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
15529:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
15530:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
15531:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
15532:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
15533:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
15534:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
15535:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
15536:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
15537:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
15538:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
15539:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
15540:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
15541:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
15542:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
15543:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
15544:   SIRegister_TLDAPAttribute(CL);
15545:   SIRegister_TLDAPAttributeList(CL);
15546:   SIRegister_TLDAPResult(CL);
15547:   SIRegister_TLDAPResultList(CL);
15548:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace );
15549:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree );
15550:   AddTypeS(TLDAPSearchAliases,(SA_NeverDeref,SA_InSearching,SA_FindingBaseObj,SA_Always);
15551:   SIRegister_TLDAPSend(CL);
15552:  Func LDAPResultDump( const Value : TLDAPResultList) : Ansistr;
15553: end;
15554:
15555:
15556: Proc SIRegister_slogsend(CL: TPSPascalCompiler);
15557: begin
15558:  Const('cSysLogProtocol','String '514;
15559:  Const('FCL_Kernel','LongInt'( 0);
15560:  Const('FCL_UserLevel','LongInt'( 1);
```

```
15561:  Const('FCL_MailSystem','LongInt'( 2);
15562:  Const('FCL_System','LongInt'( 3);
15563:  Const('FCL_Security','LongInt'( 4);
15564:  Const('FCL_Syslogd','LongInt'( 5);
15565:  Const('FCL_Printer','LongInt'( 6);
15566:  Const('FCL_News','LongInt'( 7);
15567:  Const('FCL_UUCP','LongInt'( 8);
15568:  Const('FCL_Clock','LongInt'( 9);
15569:  Const('FCL_Authorization','LongInt'( 10);
15570:  Const('FCL_FTP','LongInt'( 11);
15571:  Const('FCL_NTP','LongInt'( 12);
15572:  Const('FCL_LogAudit','LongInt'( 13);
15573:  Const('FCL_LogAlert','LongInt'( 14);
15574:  Const('FCL_Time','LongInt'( 15);
15575:  Const('FCL_Local0','LongInt'( 16);
15576:  Const('FCL_Local1','LongInt'( 17);
15577:  Const('FCL_Local2','LongInt'( 18);
15578:  Const('FCL_Local3','LongInt'( 19);
15579:  Const('FCL_Local4','LongInt'( 20);
15580:  Const('FCL_Local5','LongInt'( 21);
15581:  Const('FCL_Local6','LongInt'( 22);
15582:  Const('FCL_Local7','LongInt'( 23);
15583:   Type(TSyslogSeverity','(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15584:   SIRegister_TSyslogMessage(CL);
15585:   SIRegister_TSyslogSend(CL);
15586:  Func ToSysLog(const SyslogServer:str;Facil:Byte;Sever:TSyslogSeverity;const Content:str):Bool;
15587: end;
15588:
15589: Proc SIRegister_mimemess(CL: TPSPascalCompiler);
15590: begin
15591:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high );
15592:   SIRegister_TMessHeader(CL);
15593:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader;
15594:   SIRegister_TMimeMess(CL);
15595: end;
15596:
15597: Proc SIRegister_mimepart(CL: TPSPascalCompiler);
15598: begin
15599:   (FindClass('TOBJECT'),'TMimePart;
15600:   AddTypeS('THookWalkPart', 'Proc ( const Sender : TMimePart);
15601:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY );
15602:   AddTypeS(TMimeEncoding,(ME_7BIT,ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX );
15603:   SIRegister_TMimePart(CL);
15604:  Const('MaxMimeType','LongInt'( 25);
15605:  Func GenerateBoundary :Str;
15606: end;
15607:
15608: Proc SIRegister_mimeinln(CL: TPSPascalCompiler);
15609: begin
15610:  Func InlineDecode( const Value :Str; CP : TMimeChar) :Str;
15611:  Func InlineEncode( const Value :Str; CP, MimeP : TMimeChar) :Str;
15612:  Func NeedInline( const Value : Ansistr) :Bool;
15613:  Func InlineCodeEx( const Value :Str; FromCP : TMimeChar) :Str;
15614:  Func InlineCode( const Value :Str) :Str;
15615:  Func InlineEmailEx( const Value :Str; FromCP : TMimeChar) :Str;
15616:  Func InlineEmail( const Value :Str) :Str;
15617: end;
15618:
15619: Proc SIRegister_ftpsend(CL: TPSPascalCompiler);
15620: begin
15621:  Const('cFtpProtocol','String '21;
15622:  Const('cFtpDataProtocol','String '20;
15623:  Const('FTP_OK','LongInt'( 255);
15624:  Const('FTP_ERR','LongInt'( 254);
15625:   AddTypeS('TFTPStatus', 'Procedure(Sender:TObject; Response:Boolean; const Value :Str);
15626:   SIRegister_TFTPListRec(CL);
15627:   SIRegister_TFTPList(CL);
15628:   SIRegister_TFTPSend(CL);
15629:  Func FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass :Str) :Bool;
15630:  Func FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass :Str) :Bool;
15631:  Func FtpInterServerTransfer(const FromIP,FromPort,FromFile,FromUser,FromPass:str;const ToIP,ToPort,ToFile,
       ToUser,ToPass:str):Bool;
15632: end;
15633:
15634: Proc SIRegister_httpsend(CL: TPSPascalCompiler);
15635: begin
15636:  Const('cHttpProtocol','String '80;
15637:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED );
15638:   SIRegister_THTTPSend(CL);
15639:  Func HttpGetText( const URL :Str; const Response : TStrings) :Bool;
15640:  Func HttpGetBinary( const URL :Str; const Response : TStream) :Bool;
15641:  Func HttpPostBinary( const URL :Str; const Data : TStream) :Bool;
15642:  Func HttpPostURL( const URL, URLData :Str; const Data : TStream) :Bool;
15643:  Func HttpPostFile(const URL,FieldName,FileName:str;const Data:TStream;const ResultData:TStrings):Bool;
15644: end;
15645:
15646: Proc SIRegister_smtpsend(CL: TPSPascalCompiler);
15647: begin
15648:  Const('cSmtpProtocol','String '25;
```

```
15649:   SIRegister_TSMTPSend(CL);
15650:  Func SendToRaw(const MailFr,MailTo,SMTPHost:str;const MailData:TStrings;const Usrname,Passw:str):Bool;
15651:  Func SendTo(const MailFrom,MailTo,Subject,SMTPHost:str;const MailData:TStrings):Bool;
15652:  Func SendToEx(const MailFrom, MailTo, Subject, SMTPHost:str; const MailData:TStrings; const Username,
       Password :Str):Boolean;
15653: end;
15654:
15655: Proc SIRegister_snmpsend(CL: TPSPascalCompiler);
15656: begin
15657:  Const('cSnmpProtocol','String '161;
15658:  Const('cSnmpTrapProtocol','String '162;
15659:  Const('SNMP_V1','LongInt'( 0);
15660:  Const('SNMP_V2C','LongInt'( 1);
15661:  Const('SNMP_V3','LongInt'( 3);
15662:  Const('PDUGetRequest','LongWord')( $A0);
15663:  Const('PDUGetNextRequest','LongWord')( $A1);
15664:  Const('PDUGetResponse','LongWord')( $A2);
15665:  Const('PDUSetRequest','LongWord')( $A3);
15666:  Const('PDUTrap','LongWord')( $A4);
15667:  Const('PDUGetBulkRequest','LongWord')( $A5);
15668:  Const('PDUInformRequest','LongWord')( $A6);
15669:  Const('PDUTrapV2','LongWord')( $A7);
15670:  Const('PDUReport','LongWord')( $A8);
15671:  Const('ENoError',LongInt 0;
15672:  Const('ETooBig','LongInt')( 1);
15673:  Const('ENoSuchName','LongInt'( 2);
15674:  Const('EBadValue','LongInt'( 3);
15675:  Const('EReadOnly','LongInt'( 4);
15676:  Const('EGenErr','LongInt'( 5);
15677:  Const('ENoAccess','LongInt'( 6);
15678:  Const('EWrongType','LongInt'( 7);
15679:  Const('EWrongLength','LongInt'( 8);
15680:  Const('EWrongEncoding','LongInt'( 9);
15681:  Const('EWrongValue','LongInt'( 10);
15682:  Const('ENoCreation','LongInt'( 11);
15683:  Const('EInconsistentValue','LongInt'( 12);
15684:  Const('EResourceUnavailable','LongInt'( 13);
15685:  Const('ECommitFailed','LongInt'( 14);
15686:  Const('EUndoFailed','LongInt'( 15);
15687:  Const('EAuthorizationError','LongInt'( 16);
15688:  Const('ENotWritable','LongInt'( 17);
15689:  Const('EInconsistentName','LongInt'( 18);
15690:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv );
15691:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 );
15692:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES );
15693:   SIRegister_TSNMPMib(CL);
15694:   AddTypeS(TV3Sync', 'record EngineID : Ansistr; EngineBoots : Int; EngineTime : Int; EngineStamp :Card;
       end;
15695:   SIRegister_TSNMPRec(CL);
15696:   SIRegister_TSNMPSend(CL);
15697:  Func SNMPGet(const OID,Community,SNMPHost:Ansistr; var Value: Ansistr) :Bool;
15698:  Func SNMPSet(const OID,Community,SNMPHost,Value:Ansistr;ValueType:Int):Bool;
15699:  Func SNMPGetNext(var OID:Ansistr;const Community,SNMPHost:Ansistr;var Value:Ansistr):Bool;
15700:  Func SNMPGetTable(const BaseOID,Community,SNMPHost:Ansistr;const Value:TStrings):Bool;
15701:  Func SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15702:  Func SendTrap(const Dest,Source, Enterprise, Community : Ansistr; Generic, Specific, Seconds : Int; const
       MIBName, MIBValue : Ansistr; MIBtype : Int) : Int;
15703:  Func RecvTrap(var Dest,Source,Enterprise,Community:Ansistr;var Generic,Specific,Seconds Int; const
       MIBName,MIBValue : TStringList): Int;
15704: end;
15705:
15706: Proc SIRegister_NetWork(CL: TPSPascalCompiler);
15707: begin
15708:  Func GetDomainName2: Ansistr;
15709:  Func GetDomainController( Domain : Ansistr) : Ansistr;
15710:  Func GetDomainUsers( Controller : Ansistr) : Ansistr;
15711:  Func GetDomainGroups( Controller : Ansistr) : Ansistr;
15712:  Func GetDateTime( Controller : Ansistr) : TDateTime;
15713:  Func GetFullName2( Controller, UserID : Ansistr) : Ansistr;
15714: end;
15715:
15716: Proc SIRegister_wwSystem(CL: TPSPascalCompiler);
15717: begin
15718:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD );
15719:   TwwDateTimeSelection,''(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15720:  Func wwStrToDate( const S :Str) :Bool;
15721:  Func wwStrToTime( const S :Str) :Bool;
15722:  Func wwStrToDateTime( const S :Str) :Bool;
15723:  Func wwStrToTimeVal( const S :Str) : TDateTime;
15724:  Func wwStrToDateVal( const S :Str) : TDateTime;
15725:  Func wwStrToDateTimeVal( const S :Str) : TDateTime;
15726:  Func wwStrToInt( const S :Str) :Bool;
15727:  Func wwStrToFloat( const S :Str) :Bool;
15728:  Func wwGetDateOrder( const DateFormat :Str) : TwwDateOrder;
15729:  Func wwNextDay( Year, Month, Day : Word) : Int;
15730:  Func wwPriorDay( Year, Month, Day : Word) : Int;
15731:  Func wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) :Bool;
15732:  Func wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) :Bool;
15733:  Func wwGetDateTimeCursorPosition(SelStart:int;Text:str;TimeOnly:Bool):TwwDateTimeSelection;
```

```
15734:  Func wwGetTimeCursorPosition( SelStart : Int; Text :Str) : TwwDateTimeSelection;
15735:  Func wwScanDate( const S :Str; var Date : TDateTime) :Bool;
15736:  Func wwScanDateEpoch(const S:Str; var Date : TDateTime; Epoch : Int):Bool;
15737:  Proc wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15738:  Func wwStrToFloat2(const S:Str; var FloatValue:Extended;DisplayFormat:str):bool;
15739: end;
15740:
15741: unit uPSI_Themes;
15742: Func ThemeServices : TThemeServices;
15743: Func ThemeControl( AControl : TControl) :Bool;
15744: Func UnthemedDesigner( AControl : TControl) :Bool;
15745: Proc SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15746: begin
15747:  Func GetBindingkeyAccessPoint(const Operator:Str; const key:Str):Str;
15748: end;
15749: Unit uPSC_menus;
15750:  Func StripHotkey( const Text :Str) :Str;
15751:  Func GetHotkey( const Text :Str) :Str;
15752:  Func AnsiSameCaption( const Text1, Text2 :Str) :Bool;
15753:  Func IsAltGRPressed :Bool;
15754:
15755: Proc SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15756: begin
15757:   TCommandEvent,Procedure(Thread:TIdPeerThread; const Tag,CmdStr:str;var Handled:Bool);
15758:   SIRegister_TIdIMAP4Server(CL);
15759: end;
15760:
15761: Proc SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15762: begin
15763:  'HASH_SIZE','LongInt'( 256);
15764:   CL.FindClass('TOBJECT'),'EVariantSymbolTable;
15765:   TypeS('TSymbolType', '( stInt, stFloat, stDate, stString );
15766:   //TypeS('PSymbol', '^TSymbol // will not work;
15767:   TypeS('TSymbol', 'record Name :Str; BlockLevel : Int; HashValue'
15768:    +' : Int; Value : Variant; end;
15769:   //TypeS('PSymbolArray', '^TSymbolArray // will not work;
15770:   SIRegister_TVariantSymbolTable(CL);
15771: end;
15772:
15773: Proc SIRegister_udf_glob(CL: TPSPascalCompiler);
15774: begin
15775:   SIRegister_TThreadLocalVariables(CL);
15776:  Func MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar;
15777:  //Func MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD;
15778:  Func ThreadLocals : TThreadLocalVariables;
15779:  Proc WriteDebug( sz :Str);
15780:  ConstantN('UDF_SUCCESS','LongInt'( 0);
15781:  'UDF_FAILURE','LongInt'( 1);
15782:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15783:  TypeS('mTByteArray', 'array of byte;;
15784:  Func ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15785:  Func ChangeOEPFromFile(sFile:str; sDestFile:str):Boolean;
15786:  Proc CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget:Str);
15787:  Func IsNetworkConnected:Bool;
15788:  Func IsInternetConnected:Bool;
15789:  Func IsCOMConnected:Bool;
15790:  Func IsNetworkOn:Bool;
15791:  Func IsInternetOn:Bool;
15792:  Func IsCOMOn:Bool;
15793:  Func SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15794:  TmrProc', 'Proc TmrProc(hWnd: HWND; uMsg: Int; idEvent: Int; dwTime: Int);;
15795:  Func SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT;
15796:  Func KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL;
15797:  Func wIsWindowUnicode( hWnd : HWND) : BOOL;
15798:  Func wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL;
15799:  Func wIsWindowEnabled( hWnd : HWND) : BOOL;
15800:  Func GetMenu( hWnd : HWND) : HMENU;
15801:  Func SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL;
15802: end;
15803:
15804: Proc SIRegister_SockTransport(CL: TPSPascalCompiler);
15805: begin
15806:   SIRegister_IDataBlock(CL);
15807:   SIRegister_ISendDataBlock(CL);
15808:   SIRegister_ITransport(CL);
15809:   SIRegister_TDataBlock(CL);
15810:   //TypeS('PIntArray', '^TIntArray // will not work;
15811:   //TypeS('PVariantArray', '^TVariantArray // will not work;
15812:   TypeS('TVarFlag', '( vfByRef, vfVariant );
15813:   TypeS('TVarFlags', 'set of TVarFlag;
15814:   SIRegister_TCustomDataBlockInterpreter(CL);
15815:   SIRegister_TSendDataBlock(CL);
15816:   'CallSig','LongWord')( $D800);
15817:   'ResultSig','LongWord')( $D400);
15818:   'asMask','LongWord')( $00FF);
15819:   ClassN(CL.FindClass('TOBJECT'),'EInterpreterError;
15820:   ClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError;
15821:  Proc CheckSignature( Sig : Int);
15822: end;
```

```
15823:
15824: Proc SIRegister_WinInet(CL: TPSPascalCompiler);
15825: begin
15826:   //TypeS('HINTERNET', '___Pointer;
15827:   TypeS('HINTERNET1', 'THANDLE;
15828:   TypeS('HINTERNET', 'Int;
15829:   TypeS('HINTERNET2', '___Pointer;
15830:   //TypeS('PHINTERNET', '^HINTERNET // will not work;
15831:   //TypeS('LPHINTERNET', 'PHINTERNET;
15832:   TypeS('INTERNET_PORT', 'Word;
15833:   //TypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work;
15834:   //TypeS('LPINTERNET_PORT', 'PINTERNET_PORT;
15835:   Func InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15836:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15837:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15838:  Func InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
15839:  Func InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
        dwUrlLength:DWORD):BOOL;
15840:  Func InternetCloseHandle(hInet : HINTERNET) : BOOL;
15841:  Func
        InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;lpszPassword:PChar;d
15842:  Func InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15843:  ;dwContext:DWORD):HINTERNET;
15844:  Func InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
        lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15845:  Func InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
        dwContext:DWORD):BOOL;
15846:  Func InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL;
15847:  Func
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15848:  Func InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD;
15849:  Func InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL;
15850:  Func InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL;
15851:  Func InternetAutodialHangup( dwReserved : DWORD) : BOOL;
15852:  Func InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL;
15853:  Func InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
        lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15854:  Func InternetCombineUrl(lpszBaseUrl,lpszRelativeUrl:PChar;lpszBuffer:PChar;var lpdwBufferLength:Func
        InternetCloseHandle(hInet:HINTERNET):BOOL;
15855:  Func InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15856:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15857:  Func InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
        dwContext:DWORD):BOOL;
15858:  Func FtpFindFirstFile(hConnect:HINTERNET;lpszSearchFile:PChar;var lpFindFileData:TWin32FindData;dwFlags:
        DWORD; dwContext:DWORD): HINTERNET;
15859:  Func
        WFtpGetFile(hConnect:HINTERNET;lpszRemoteFile:PChar;lpszNewFile:PChar;fFailIfExists:BOOL;dwFlagsAndAttributes:DWORD;dw
15860:  Func WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15861:  Func FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL;
15862:  Func FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15863:  Func FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15864:  Func FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL;
15865:  Func FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL;
15866:  Func FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL;
15867:  Func FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15868:  Func FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15869:  Func IS_GOPHER_FILE( GopherType : DWORD) : BOOL;
15870:  Func IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL;
15871:  Func IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL;
15872:  Func IS_GOPHER_ERROR( GopherType : DWORD) : BOOL;
15873:  Func IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL;
15874:  Func IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL;
15875:  Func IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL;
15876:  Func IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL;
15877:  Func IS_GOPHER_ASK( GopherType : DWORD) : BOOL;
15878:  Func IS_GOPHER_PLUS( GopherType : DWORD) : BOOL;
15879:  Func IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL;
15880:  Func
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorStr:PChar;dwGopherTyp
        lpdwBufferLength:DWORD):BOOL;
15881:  Func GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL;
15882:  Func
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15883:  Func
        HttpOpenRequest(hConnect:HINTERNET;lpszVerb:PChar;lpszObjectName:PChar;lpszVersion:PChar;lpszReferrer:
        PChar;lplpszAcceptTypes:PLPSTR;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15884:  Func HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15885:  Func HttpSendRequest(hRequest:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;lpOptional:Tobject;
        dwOptionalLength:DWORD):BOOL;
15886:  Func InternetGetCookie(lpszUrl,lpszCookName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15887:  Func InternetAttemptConnect( dwReserved : DWORD) : DWORD;
15888:  Func InternetCheckConnection(lpszUrl:PChar; dwFlags:DWORD; dwReserved:DWORD) : BOOL;
15889:  Func InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15890:  Func InternetConfirmZoneCrossing(hWnd:HWND;szUrlPrev,szUrlNew:LPSTR;bPost:BOOL): DWORD;
15891:  Func CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64;
15892:  Func DeleteUrlCacheGroup(GroupId: Int64; dwFlags : DWORD; lpReserved : TObject) : Bool;
15893:  Func FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15894:  Func FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo;var
        lpdwNextCacheEntryInfoBufferSize:DWORD):BOOL;
```

```
15895:  Func FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15896:  Func DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15897:  Func
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15898:  Func InternetSetDialState(lpszConnectoid:PChar; dwState:DWORD;dwReserved: DWORD): BOOL;
15899: end;
15900:
15901: Proc SIRegister_Wwstr(CL: TPSPascalCompiler);
15902: begin
15903:   AddTypeS('strCharSet', 'set of char;
15904:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15905:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption;
15906:  Proc strBreakApart( s :Str; delimeter :Str; parts : TStrings);
15907:  Func strGetToken( s :Str; delimeter :Str; var APos : Int) :Str;
15908:  Proc strStripPreceding( var s :Str; delimeter : strCharSet);
15909:  Proc strStripTrailing( var s :Str; delimeter : strCharSet);
15910:  Proc strStripWhiteSpace( var s :Str);
15911:  Func strRemoveChar( str :Str; removeChar : char) :Str;
15912:  Func wwstrReplaceChar( str :Str; removeChar, replaceChar : char) :Str;
15913:  Func strReplaceCharWithStr(str:str; removeChar:char;replaceStr:Str) :Str;
15914:  Func wwEqualStr( s1, s2 :Str) :Bool;
15915:  Func strCount( s :Str; delimeter : char) : Int;
15916:  Func strWhiteSpace : strCharSet;
15917:  Func wwExtractFileNameOnly( const FileName :Str) :Str;
15918:  Func wwGetWord(s:str;var APos:Int;Options:TwwGetWordOptions;DelimSet:strCharSet):str;
15919:  Func strTrailing( s :Str; delimeter : char) :Str;
15920:  Func strPreceding( s :Str; delimeter : char) :Str;
15921:  Func wwstrReplace( s, Find, Replace :Str) :Str;
15922: end;
15923:
15924:  Ex.:tex2:= 'Finally, we recognize that the 8% is calculated over all states of the world.'
15925:       delimset:= [',','?',';','.',':','-','=','{','}','[',']','(',')','>','<','%'];
15926:       while length(tex2)>0 do begin
15927:         k:=1;
15928:         s:= wwGetWord(tex2, k, [wwgwStripQuotes, wwgwSpacesInWords], delimSet);
15929:         delete(tex2,1,k);
15930:         write(' '+s)
15931:       end;
15932:
15933: Proc SIRegister_DataBkr(CL: TPSPascalCompiler);
15934: begin
15935:   SIRegister_TRemoteDataModule(CL);
15936:   SIRegister_TCRemoteDataModule(CL);
15937:  Proc RegisterPooled(const ClassID:str; Max,Timeout : Int; Singleton :Bool);
15938:  Proc UnregisterPooled( const ClassID :Str);
15939:  Proc EnableSocketTransport( const ClassID :Str);
15940:  Proc DisableSocketTransport( const ClassID :Str);
15941:  Proc EnableWebTransport( const ClassID :Str);
15942:  Proc DisableWebTransport( const ClassID :Str);
15943: end;
15944:
15945: Proc SIRegister_Mathbox(CL: TPSPascalCompiler);
15946: begin
15947:  Func mxArcCos( x : Real) : Real;
15948:  Func mxArcSin( x : Real) : Real;
15949:  Func Comp2Str( N : Comp) :Str;
15950:  Func Int2StrPad0( N : LongInt; Len : Int) :Str;
15951:  Func Int2Str( N : LongInt) :Str;
15952:  Func mxIsEqual( R1, R2 : Double) :Bool;
15953:  Func LogXY( x, y : Real) : Real;
15954:  Func Pennies2Dollars( C : Comp) :Str;
15955:  Func mxPower( X : Int; Y : Int) : Real;
15956:  Func Real2Str( N : Real; Width, Places : Int) :Str;
15957:  Func mxStr2Comp( MyString :Str) : Comp;
15958:  Func mxStr2Pennies( S :Str) : Comp;
15959:  Func Str2Real( MyString :Str) : Real;
15960:  Func XToTheY( x, y : Real) : Real;
15961: end;
15962:
15963: //*************************Cindy Functions!******************************
15964: Proc SIRegister_cyIndy(CL: TPSPascalCompiler);
15965: begin
15966:  TypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15967:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15968:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification);
15969:  MessagePlainText','String 'text/plain;
15970:  ConstantN('MessagePlainText_Attach','String 'multipart/mixed;
15971:  MessageAlterText_Html','String 'multipart/alternative;
15972:  MessageHtml_Attach','String 'multipart/mixed;
15973:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html";
15974:  MessageAlterText_Html_Attach','String 'multipart/mixed;
15975:  MessageAlterText_Html_RelatedAttach,String)('multipart/related;type="multipart/alternative"';
15976:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed;
15977:  MessageReadNotificatin String).(multipart/report;report-type="disposition-notification"';
15978:  Func ForceDecodeHeader( aHeader :Str) :Str;
15979:  Func Base64_EncodeString( Value :Str; const aEncoding : TEncoding) :Str;
15980:  Func Base64_DecodeToString(Value:Str; const aBytesEncoding:TEncoding):Str;;
15981:  Func Base64_DecodeToBytes( Value :Str) : TBytes;;
15982:  Func IdHttp_DownloadFile(aSrcUrlFile,aDestFile:str;const OnWorkEvent:TWorkEvent):Bool;
```

```
15983:  Func Get_MD5( const aFileName :Str) :Str;
15984:  Func Get_MD5FromString( const aString :Str) :Str;
15985: end;
15986:
15987: Proc SIRegister_cySysUtils(CL: TPSPascalCompiler);
15988: begin
15989:  Func IsFolder( SRec : TSearchrec) :Bool;
15990:  Func isFolderReadOnly( Directory :Str) :Bool;
15991:  Func DirectoryIsEmpty( Directory :Str) :Bool;
15992:  Func DirectoryWithSubDir( Directory :Str) :Bool;
15993:  Proc GetSubDirs( FromDirectory :Str; aList : TStrings);
15994:  Func DiskFreeBytes( Drv : Char) : Int64;
15995:  Func DiskBytes( Drv : Char) : Int64;
15996:  Func GetFileBytes( Filename :Str) : Int64;
15997:  Func GetFilesBytes( Directory, Filter :Str) : Int64;
15998:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege;
15999:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege;
16000:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege;
16001:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege;
16002:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege;
16003:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege;
16004:  SE_TCB_NAME','String 'SeTcbPrivilege;
16005:  SE_SECURITY_NAME','String 'SeSecurityPrivilege;
16006:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege;
16007:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege;
16008:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege;
16009:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege;
16010:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege;
16011:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege;
16012:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege;
16013:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege;
16014:  SE_BACKUP_NAME','String 'SeBackupPrivilege;
16015:  SE_RESTORE_NAME','String 'SeRestorePrivilege;
16016:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege;
16017:  SE_DEBUG_NAME','String 'SeDebugPrivilege;
16018:  SE_AUDIT_NAME','String 'SeAuditPrivilege;
16019:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege;
16020:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege;
16021:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege;
16022:  SE_UNDOCK_NAME','String 'SeUndockPrivilege;
16023:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege;
16024:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege;
16025:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege;
16026: end;
16027:
16028: Proc SIRegister_cyWinUtils(CL: TPSPascalCompiler);
16029: begin
16030:   Type(TWindowsVersion','( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
16031:    +Me, wvWinNt3, wvWinNt4,wvWin2000,wvWinXP,wvWinVista, wvWin7,wvWin8 wvWin8_Or_Upper );
16032:  Func ShellGetExtensionName( FileName :Str) :Str;
16033:  Func ShellGetIconIndex( FileName :Str) : Int;
16034:  Func ShellGetIconHandle( FileName :Str) : HIcon;
16035:  Proc ShellThreadCopy( App_Handle : THandle; fromFile :Str; toFile :Str);
16036:  Proc ShellThreadMove( App_Handle : THandle; fromFile :Str; toFile :Str);
16037:  Proc ShellRenameDir( DirFrom, DirTo:Str);
16038:  Func cyShellExecute(Operation,FileName,Parameters,Directory:str;ShowCmd:Int):Card;
16039:  Proc cyShellExecute1(ExeFilename,Params,ApplicName,ApplicationClass:str;Restore:Bool);
16040:  Proc ShellExecuteAsModal( ExeFilename, ApplicationName, Directory :Str);
16041:  Proc ShellExecuteExAsAdmin( hWnd : HWND; Filename :Str; Parameters :Str);
16042:  Func ShellExecuteEx(aFileName:str;const Parameters:str;const Directory:str;const
       WaitCloseCompletion:bool):Bool;
16043:  Proc RestoreAndSetForegroundWindow( Hnd : Int);
16044:  Func RemoveDuplicatedPathDelimiter( Str :Str) :Str;
16045:  Func cyFileTimeToDateTime( _FT : TFileTime) : TDateTime;
16046:  Func GetModificationDate( Filename :Str) : TDateTime;
16047:  Func GetCreationDate( Filename :Str) : TDateTime;
16048:  Func GetLastAccessDate( Filename :Str) : TDateTime;
16049:  Func FileDelete( Filename :Str) :Bool;
16050:  Func FileIsOpen( Filename :Str) :Bool;
16051:  Proc FilesDelete( FromDirectory :Str; Filter : ShortString);
16052:  Func DirectoryDelete( Directory :Str) :Bool;
16053:  Func GetPrinters( PrintersList : TStrings) : Int;
16054:  Proc SetDefaultPrinter( PrinterName :Str);
16055:  Proc ShowDefaultPrinterWindowProperties( FormParent_Handle : Int);
16056:  Func WinToDosPath( WinPathName :Str) :Str;
16057:  Func DosToWinPath( DosPathName :Str) :Str;
16058:  Func cyGetWindowsVersion : TWindowsVersion;
16059:  Func NTSetPrivilege( sPrivilege :Str; bEnabled :Bool) :Bool;
16060:  Proc WindowsShutDown( Restart :Bool);
16061:  Proc CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,FileIcon:str;NumIcone:Int);
16062:  Proc GetWindowsFonts( FontsList : TStrings);
16063:  Func GetAvailableFilename( DesiredFileName :Str):Str;
16064: end;
16065:
16066: Proc SIRegister_cyStrUtils(CL: TPSPascalCompiler);
16067: begin
16068:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey );
16069:   Type(TStrLocateOptions', 'set of TStrLocateOption;
16070:   Type(TStringRead', '( srFromLeft, srFromRight );
```

```
16071:   Type(TStringReads', 'set of TStringRead;
16072:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive );
16073:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar );
16074:   Type(TWordsOptions', 'set of TWordsOption;
16075:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther );
16076:   Type(TCarTypes', 'set of TCarType;
16077:   //TypeS('TUnicodeCategories', 'set of TUnicodeCategory;
16078:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
16079:  Func Char_GetType( aChar : Char) : TCarType;
16080:  Func SubString_Count( Str :Str; Separator : Char) : Int;
16081:  Func SubString_AtPos( Str :Str; Separator : Char; SubStringIndex : Word) : Int;
16082:  Func SubString_Get( Str :Str; Separator : Char; SubStringIndex : Word) :Str;
16083:  Func SubString_Length( Str :Str; Separator : Char; SubStringIndex : Word) : Int;
16084:  Proc SubString_Add( var Str :Str; Separator : Char; Value :Str);
16085:  Proc SubString_Insert(var Str:str;Separator:Char;SubStrIndex:Word;Value:str);
16086:  Proc SubString_Edit(var Str:str;Separator:Char;SubStringIndex:Word; NewValue:str);
16087:  Func SubString_Remove(var Str:str; Separator:Char; SubStringIndex : Word) :Bool;
16088:  Func SubString_Locate(Str:str;Separator:Char;SubString:Str;Options:TStrLocateOptions):Int;
16089:  Func SubString_Ribbon(Str:str; Separator:Char;Current: Word; MoveBy:Int):Int;;
16090:  Func SubString_Ribbon1(Str:str;Separator:Char;Current:str;MoveBy:Int):str;;
16091:  Func String_Quote( Str :Str) :Str;
16092:  Func String_GetCar(Str:str; Position : Word; ReturnCarIfNotExists : Char) : Char;
16093:  Func String_ExtractCars(fromStr:Str;CarTypes:TCarTypes;IncludeCars,ExcludeCars:Str):str;
16094:  Func String_GetWord( Str :Str; StringRead : TStringRead) :Str;
16095:  Func String_GetInt( Str :Str; StringRead : TStringRead) :Str;
16096:  Func String_ToInt( Str :Str) : Int;
16097:  Func String_Uppercase( Str :Str; Options : TWordsOptions):Str;
16098:  Func String_Lowercase( Str :Str; Options : TWordsOptions):Str;
16099:  Func String_Reverse( Str :Str) :Str;
16100:  Func String_Pos(SubStr:str;Str:str;fromPos:Int;CaseSensitive:TCaseSensitiv)Int;
16101:  Func
        String_Pos1(SubStr:str;Str:str;StringRead:TStringRead;Occurrence:Word;CaseSensitive:TCaseSensitive):Int
16102:  Func String_Copy( Str :Str; fromIndex : Int; toIndex : Int) :Str;;
16103:  Func String_Copy1(Str:str;StringRead:TStringRead;UntilFind:Str;_Inclusive:Boo):str;
16104:  Func String_Copy2(Str:str;Between1:str;Between1MustExist:Boolean;Between2:str;
        Between2MustExist:Boolean;CaseSensitive:TCaseSensitive):str;
16105:  Func String_Delete( Str :Str; fromIndex : Int; toIndex : Int) :Str;;
16106:  Func String_Delete1(Str:str; delStr:str; CaseSensitive:TCaseSensitive):str;;
16107:  Func String_BoundsCut( Str :Str; CutCar : Char; Bounds : TStringReads) :Str;
16108:  Func String_BoundsAdd( Str :Str; AddCar : Char; ReturnLength : Int) :Str;
16109:  Func String_Add(Str:str;StringRead:TStringRead;aCar:Char;ReturnLength:Int):str;
16110:  Func String_End( Str :Str; Cars : Word) :Str;
16111:  Func String_Subst(OldStr:str;NewStr:str;Str:str;
        CaseSensitive:TCaseSensitive;AlwaysFindFromBeginning:Bool):Str;
16112:  Func String_SubstCar( Str :Str; Old, New : Char) :Str;
16113:  Func String_Count(Str:Str; SubStr:Str; CaseSenSitive : TCaseSensitive) : Int;
16114:  Func String_SameCars(Str1,Str2:str;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Int;
16115:  Func String_IsNumbers( Str :Str) :Bool;
16116:  Func SearchPos( SubStr :Str; Str :Str; MaxErrors : Int) : Int;
16117:  Func StringToCsvCell( aStr :Str) :Str;
16118: end;
16119:
16120: Proc SIRegister_cyDateUtils(CL: TPSPascalCompiler);
16121: begin
16122:  Func LongDayName( aDate : TDate) :Str;
16123:  Func LongMonthName( aDate : TDate) :Str;
16124:  Func ShortYearOf( aDate : TDate) : byte;
16125:  Func DateToStrYYYYMMDD( aDate : TDate) :Str;
16126:  Func StrYYYYMMDDToDate( aStr :Str) : TDate;
16127:  Func SecondsToMinutes( Seconds : Int) : Double;
16128:  Func MinutesToSeconds( Minutes : Double) : Int;
16129:  Func MinutesToHours( Minutes : Int) : Double;
16130:  Func HoursToMinutes( Hours : Double) : Int;
16131:  Func ShortTimeStringToTime(ShortTimeStr:str;const ShortTimeFormat:str): TDateTime;
16132:  Proc cyAddMonths( var aMonth, aYear : Word; Months : Int);
16133:  Func MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime;
16134:  Func GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;;
16135:  Func GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:str;const ShortTimeFormat:str):Int64;
16136:  Func GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;;
16137:  Func IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime;var
        RsltBegin:TDateTime;RsltEnd:TDateTime):Bool;
16138:  Func IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
16139:  Func TryToEncodeDate( Year, Month, Day : Int; var RsltDate : TDateTime) :Bool;
16140: end;
16141:
16142: Proc SIRegister_cyObjUtils(CL: TPSPascalCompiler);
16143: begin
16144:   Type(TStringsSortType','( stNone, stStringSensitive, stStringInsensitive, stExtended );
16145:   Type(TStringsValueKind','( skStringSensitive, skStringInsensitive, skExtended );
16146:   Type(TcyLocateOption','( lCaseInsensitive, lPartialKey );
16147:   Type(TcyLocateOptions','set of TcyLocateOption;
16148:  Func StringsLocate(aList: TStrings; Value:str; Options:TcyLocateOptions): Int;;
16149:  Func StringsLocate1(aList:TStrings; Value:Str; ValueKind:TStringsValueKind):Int;;
16150:  Func StringsAdd(aList:TStrings;Value:str;Unique:Boolean;SortType:TStringsSortType): Int;
16151:  Proc StringsReplace(aList:TStrings;OldStr:str; NewStr:str;ValueKind : TStringsValueKind);
16152:  Proc StringsSort( aList : TStrings; SortType : TStringsSortType);
16153:  Func TreeNodeLocate( ParentNode : TTreeNode; Value :Str) : TTreeNode;
16154:  Func TreeNodeLocateOnLevel(TreeView:TTreeView;OnLevel:Int;Value:str): TTreeNode;
16155:  Func TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Int):TTreeNode;
```

```
16156:  Func TreeNodeGetParentOnLevel(ChildNode: TTreeNode; ParentLevel : Int) : TTreeNode;
16157:  Proc TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
        CopySubChildren:Bool;
16158:  Proc RichEditSetStr( aRichEdit : TRichEdit; FormatedString :Str);
16159:  Proc RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:str;Flags:TReplaceFlags);
16160:  Func GetTopMostControlAtPos(FromControl:TWinControl; aControlPoint : TPoint): TControl;
16161:  Proc cyCenterControl( aControl : TControl);
16162:  Func GetLastParent( aControl : TControl) : TWinControl;
16163:  Func GetControlBitmap( aControl : TWinControl) : TBitmap;
16164:  Func GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap;
16165: end;
16166:
16167: Proc SIRegister_cyBDE(CL: TPSPascalCompiler);
16168: begin
16169:  Func TablePackTable( Tab : TTable) :Bool;
16170:  Func TableRegenIndexes( Tab : TTable) :Bool;
16171:  Func TableShowDeletedRecords( Tab : TTable; Show :Bool) :Bool;
16172:  Func TableUndeleteRecord( Tab : TTable) :Bool;
16173:  Func TableAddIndex(Tab:TTable;FieldName Str;FieldExpression:Str;IOpt:TIndexOptions):Bool;
16174:  Func TableDeleteIndex( Tab : TTable; IndexFieldName :Str) :Bool;
16175:  Func TableEmptyTable( Tab : TTable) :Bool;
16176:  Func TableFindKey( aTable : TTable; Value :Str) :Bool;
16177:  Proc TableFindNearest( aTable : TTable; Value :Str);
16178:  Func
        TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:str;IdxName:ShortString;ReadOnly:Bool):TTable;
16179:  Func
        TableOpen(Tab:TTable;FileName:str;IndexFieldName:str;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
16180:  Func DateToBDESQLDate( aDate : TDate; const DateFormat :Str) :Str;
16181: end;
16182:
16183: Proc SIRegister_cyClasses(CL: TPSPascalCompiler);
16184: begin
16185:   SIRegister_TcyRunTimeDesign(CL);
16186:   SIRegister_TcyShadowText(CL);
16187:   SIRegister_TcyBgPicture(CL);
16188:   SIRegister_TcyGradient(CL);
16189:   SIRegister_tcyBevel(CL);
16190:   //TypeS('TcyBevelClass', 'class of tcyBevel;
16191:   SIRegister_tcyBevels(CL);
16192:   SIRegister_TcyImagelistOptions(CL);
16193:  Proc cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture);
16194: end;
16195:
16196: Proc SIRegister_cyGraphics(CL: TPSPascalCompiler);
16197: begin
16198:  Proc cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; adgradOrientation :
        TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode; Maxdegrade : Byte;'+
16199:   SpeedPercent : Int; const AngleClipRect :Bool; const AngleBuffer : TBitmap);
16200:  Proc cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
16201:  Proc cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromCol,toColor:TColor;MaxDegrad:byte);
16202:  Proc cyGradientFillShape(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:
        Byte;toRect:TRect;OrientationShape:TDgradOrientationShape);
16203:  Proc cyGradientFillAngle(aCanvas:TCanvas;aRect TRect;fromColor,toColor:TColor;MaxDegrad:
        Byte;AngleDegree:Word;const ClipRect:Boolean;const Buffer:TBitmap);
16204:  Proc DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Int);
16205:  Proc cyFrame(aCanvas:TCanvas; var InsideRect:TRect;Color:TColor;const Width:Int);;
16206:  Proc cyFrame1(Canvas:TCanvas;var InsideRect:TRect;LeftColor,TopColor,ightColor,BottomColor:TColor;const
        Width:Int;const RoundRect:bool);;
16207:  Proc cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,BottomRightColor:TColor;Width:Int;const
        DrawLeft:Boolean;const DrawTop:Bool;const DrawRight:Bool;const DrawBottom:Bool;const RoundRect:bool;
16208:  Proc cyDrawButtonFace(Canvas:TCanvas; var Rect:Rect;GradientColor1,
        GradientColor2:TColor;aState:TButtonState;Focused,Hot:Bool);
16209:  Proc cyDrawButton(Canvas:TCanvas;Caption:str;ARect:TRect;radientColor1,GradientColor2:TColor;aState
        TButtonState;Focused,Hot : Bool);
16210:  Proc cyDrawSpeedButtonFace(Canvas:TCanvas;var Rect:TRect;GradientColor1,
        GradientColor2:TColor;aState:TButtonState;Focused,Hot:Bool);
16211:  Proc cyDrawSpeedButton(Canvas:TCanvas;Caption:str;ARect:TRect;GradientColor1,
        GradientColor2:TColor;aState: TButtonState;Focused,Hot:oolean);
16212:  Proc cyDrawCheckBox(Canvas:TCanvas;IsChecked:Boolean;ARect:TRect;const BgColor:TColor;const
        DarkFrameColor:TColor;const LightFrameColor:TColor;const MarkColor:TColor);
16213:  Proc
        cyDrawSingleLineText(Canvas:TCanvas;Text:str;ARect:TRect;Alignment:TAlignment;TextLayout:TTextLayout; const
        IndentX:Int;const IndentY:Int);
16214:  Func DrawTextFormatFlags(aTextFormat:LongInt;Alignment
        TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
16215:  Func DrawTextFormatFlags1(aTextFormat:LongInt;Alignment:TAlignment;Layout:TTextLayout;
        WordWrap:Boolean;CaptionRender:TCaptionRender):LongInt;;
16216:  Proc cyDrawText(CanvasHandl:Card;Text:str;var Rect:TRect;TextFormat:LongInt);
16217:  Func cyCreateFontIndirect(fromFont : TFont; Angle : Double) : TFont;;
16218:  Func cyCreateFontIndirect1(fromFont: TFont;CaptionOrientation:TCaptionOrientation):TFont;
16219:  Proc cyDrawVerticalText(Canvas : TCanvas; Text :Str; var Rect : TRect; TextFormat : Longint;
        CaptionOrientation:TCaptionOrientation;Alignment:TAlignment;Layout: TextLayout);
16220:  Func DrawLeftTurnPageEffect(Canvas:TCanvas;PageColor:TColor;PageRect:TRect;PercentDone:Int; const
        OnlyCalcFoldLine:Boolean):TLineCoord;
16221:  Func DrawRightTurnPageEffect(Canvas:TCanvas;PageColor:TColor;PageRect:TRect;PercentDone:Int;const
        OnlyCalcFoldLine:Boolean):TLineCoord;
16222:  Func PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean;
16223:  Func IconIsTransparentAtPos(aIcon : TIcon; aPoint : TPoint) :Bool;
16224:  Func MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) :Bool;
```

```
16225:  Func PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) :Bool;
16226:  Proc DrawCanvas(Destination:TCanvas;DestRect:TRect;Source:TCanvas;SourceRect:TRect);
16227:  Proc
        DrawCanvas1(Destination:TCanvas;DestRect:TRect;Src:TCanvas;SrcRect:TRect;TransparentColor:TColor;const
        aStyle:TBgStyle;const aPosition:TBgPosition;const IndentX:Int;const IndentY:Int;const IntervalX:Int;const
        IntervalY:Int;const RepeatX:Int;const RepeatY:Int);
16228:  Proc
        DrawGraphic(Destination:TCanvas;DestRect:TRect;aGraphic:TGraphic;SrcRect:TRect;TransparentColor:TColor;const
        aStyle:TBgStyle;const aPosition:TBgPosition;const IndentX:Int; const IndentY:Int;const IntervalX:Int;const
        IntervalY:Int;const RepeatX:Int;const RepeatY:Int;
16229:  Proc DrawGraphic1(Destination:TCanvas; DestRect:TRect;aGraphic : TGraphic; Transparent :Bool; const
        aStyle : TBgStyle; const aPosition:TBgPosition;const IndentX:Int;const IndentY:Int;const
        IntervalX:Int;const IntervalY:Int;const RepeatX:Int;const RepeatY:Int);
16230:  Proc DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap);
16231:  Func ValidGraphic( aGraphic : TGraphic) :Bool;
16232:  Func ColorSetPercentBrightness( Color : TColor; PercentLight : Int) : TColor;
16233:  Func ColorModify( Color : TColor; incR, incG, incB : Int) : TColor;
16234:  Func ColorSetPercentContrast( Color : TColor; IncPercent : Int) : TColor;
16235:  Func ColorSetPercentPale( Color : TColor; IncPercent : Int) : TColor;
16236:  Func MediumColor( Color1, Color2 : TColor) : TColor;
16237:  Func ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect;
16238:  Func ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect;
16239:  Func CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect;
16240:  Proc InflateRectPercent( var aRect : TRect; withPercent : Double);
16241:  Func GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect;
16242:  Func GetProportionalRect( fromRect, InsideRect : TRect) : TRect;
16243:  Func PointInRect( const aPt : TPoint; const aRect : TRect) :Bool;
16244:  Func PointInEllispe( const aPt : TPoint; const aRect : TRect) :Bool;
16245:  Func CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText :Str) : Int;
16246: end;
16247:
16248: Proc SIRegister_cyTypes(CL: TPSPascalCompiler);
16249: begin
16250:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight );
16251:   Type(TGlyphLayout', '( glTop, glCenter, glBottom );
16252:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome );
16253:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis );
16254:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed );
16255:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
16256:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft);
16257:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional );
16258:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext );
16259:   Type(TDgradOrientation','( dgdVertical,dgdHorizontal,dgdAngle,dgdRadial,dgdRectangle );
16260:   Type(TDgradOrientationShape', '( osRadial, osRectangle );
16261:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
        bmInvertReverseFromColor);
16262:   Type(TRunTimeDesignJob','(rjNothing,rjMove,rjResizeTop,rjResizeBottom,rjResizeLeft,rjResizeTopLeft,
        rjResizeBottomLeft,rjResizeRight,rjResizeTopRight,rjResizeBottomRight );
16263:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end;
16264:  cCaptionOrientationWarning,String)(Note text orientation doesnt work with all fonts!;
16265: end;
16266:
16267: Proc SIRegister_WinSvc(CL: TPSPascalCompiler);
16268: begin
16269:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive;
16270:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive;
16271:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA;
16272:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed;
16273:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed;
16274:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA;
16275:  Const SC_GROUP_IDENTIFIERA','String'). '+;
16276:  Const SC_GROUP_IDENTIFIERW','String') '+;
16277:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA;
16278:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
16279:  Const SERVICE_ACTIVE','LongWord')( $00000001);
16280:  Const SERVICE_INACTIVE','LongWord $00000002);
16281:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
16282:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
16283:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
16284:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
16285:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
16286:  Const SERVICE_STOPPED','LongWord $00000001);
16287:  Const SERVICE_START_PENDING','LongWord $00000002);
16288:  Const SERVICE_STOP_PENDING','LongWord $00000003);
16289:  Const SERVICE_RUNNING','LongWord $00000004);
16290:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
16291:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
16292:  Const SERVICE_PAUSED','LongWord $00000007);
16293:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
16294:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
16295:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
16296:  Const SC_MANAGER_CONNECT','LongWord $0001);
16297:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
16298:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
16299:  Const SC_MANAGER_LOCK','LongWord $0008);
16300:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
16301:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
16302:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
16303:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
```

```
16304:  Const SERVICE_QUERY_STATUS','LongWord $0004);
16305:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
16306:  Const SERVICE_START','LongWord $0010);
16307:  Const SERVICE_STOP','LongWord $0020);
16308:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
16309:  Const SERVICE_INTERROGATE','LongWord $0080);
16310:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
16311:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
16312:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
16313:  Const SERVICE_ADAPTER','LongWord $00000004);
16314:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
16315:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
16316:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
16317:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
16318:  Const SERVICE_BOOT_START','LongWord $00000000);
16319:  Const SERVICE_SYSTEM_START','LongWord $00000001);
16320:  Const SERVICE_AUTO_START','LongWord $00000002);
16321:  Const SERVICE_DEMAND_START','LongWord $00000003);
16322:  Const SERVICE_DISABLED','LongWord $00000004);
16323:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
16324:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
16325:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
16326:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
16327:   TypeS('SC_HANDLE', 'THandle;
16328:   //TypeS('LPSC_HANDLE', '^SC_HANDLE // will not work;
16329:   TypeS('SERVICE_STATUS_HANDLE', 'DWORD;
16330:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
16331:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
16332:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end;
16333:   Const SERVICE_STATUS', '_SERVICE_STATUS;
16334:   Const TServiceStatus', '_SERVICE_STATUS;
16335:   TypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
16336:    +'playName : PChar; ServiceStatus : TServiceStatus; end;
16337:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA;
16338:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA;
16339:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA;
16340:   TEnumServiceStatus', 'TEnumServiceStatusA;
16341:   SC_LOCK', '___Pointer;
16342:   _QUERY_SERVICE_LOCK_STATUSA,record fIsLocked:DWORD;lpLockOner:PChar;dwLockDuration:DWORD;end;
16343:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA;
16344:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA;
16345:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA;
16346:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA;
16347:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW;
16348:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA;
16349:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
16350:    +ype : DWORD; dwErrorControl:DWORD;lpBinaryPathName: PChar; lpLoadO'
16351:   +rderGroup:PChar;dwTagId:DWORD;lpDependencies:PChar;lpServiceStartName:PChar;lpDisplayName:PChar;end;
16352:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA;
16353:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA;
16354:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA;
16355:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA;
16356:   TQueryServiceConfig', 'TQueryServiceConfigA;
16357:  Func CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL;
16358:  Func ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
16359:  Func CreateService(hSCManager:SC_HANDLE;lpServiceName,lpDisplayName:PChar;dwDesiredAccess,dwServiceType,
        dwStartType,dwErrorControl:DWORD;lpBinaryPathName,lpLoadOrderGroup:PChar;
16360:  +' lpdwTagId: DWORD;lpDependencies,lpServiceStartName, lpPassword : PChar): SC_HANDLE;
16361:  Func CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
        dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
16362:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE;
16363:  Func DeleteService( hService : SC_HANDLE) : BOOL;
16364:  Func EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD; var
        lpServices:TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned: DWORD) : BOOL;
16365:  Func EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState:DWORD;var lpServices:
        TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD):BOOL;
16366:  Func GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var lpcchBuffer:DWORD):BOOL;
16367:  Func GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
        lpcchBuffer:DWORD):BOOL;
16368:  Func LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK;
16369:  Func NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL;
16370:  Func OpenSCManager(lpMachineName,lpDatabaseName:PChar;dwDesiredAccess:DWORD):SC_HANDLE;
16371:  Func OpenService(hSCManager:SC_HANDLE;lpServiceNam:PChar;dwDesiredAccess:DWORD):SC_HANDLE;
16372:  Func QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
        cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL;
16373:  Func QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
16374:  Func SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
16375:  Func StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
16376:  Func UnlockServiceDatabase(ScLock : SC_LOCK) : BOOL;
16377: end;
16378:
16379: Proc SIRegister_JvPickDate(CL: TPSPascalCompiler);
16380: begin
16381:  Func SelectDate(Sender:TWinControl; var Date:TDateTime; const DlgCaption:TCaption; AStartOfWeek:
        TDayOfWeekName;AWeekends:TDaysOfWeek;AWeekendColor:TColor;BtnHints:TStrings;MinDate:TDateTime;MaxDate:TDateTime):Bool;
16382:  Func SelectDateStr(Sender:TWinControl;var StrDate:str;const
        DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
16383:  Func PopupDate(var Date:TDateTime;Edit:TWinControl;MinDate:TDateTime;MaxDate:TDateTime):Bool;
16384:  Func
        CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):TWinControl;
```

```
16385:  Proc SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
       AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
16386:  Func CreateNotifyThread(const FolderName:str;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
16387: end;
16388:
16389: Proc SIRegister_JclNTFS2(CL: TPSPascalCompiler);
16390: begin
16391:   ClassN(CL.FindClass('TOBJECT'),'EJclNtfsError;
16392:   TypeS(TFileCompressionState,(fcNoCompression,fcDefaultCompression,fcLZNT1Compression);
16393:  Func NtfsGetCompression2( const FileName : TFileName; var State : Short) :Bool;;
16394:  Func NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;;
16395:  Func NtfsSetCompression2( const FileName : TFileName; const State : Short) :Bool;
16396:  Proc NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState);
16397:  Proc NtfsSetDirectoryTreeCompression2(const Directy:str;const State:TFileCompressionState);
16398:  Proc NtfsSetDefaultFileCompression2(const Directory:Str;const State:TFileCompressionState);
16399:  Proc NtfsSetPathCompression2(const Path:str;const State:TFileCompressState;Recurve:Bool);
16400:  Func NtfsSetSparse2( const FileName :Str) :Bool;
16401:  Func NtfsZeroDataByHandle2(const Handle: THandle; const First, Last : Int64) :Bool;
16402:  Func NtfsZeroDataByName2(const FileName:Str;const First,Last: Int64) :Bool;
16403:  Func NtfsSparseStreamsSupported2( const Volume :Str) :Bool;
16404:  Func NtfsGetSparse2( const FileName :Str) :Bool;
16405:  Func NtfsDeleteReparsePoint2( const FileName :Str; ReparseTag : DWORD) :Bool;
16406:  Func NtfsSetReparsePoint2(const FileName:str;var ReparseData,Size : Longword) : Bool);
16407:  Func NtfsGetReparseTag2( const Path :Str; var Tag : DWORD) :Bool;
16408:  Func NtfsReparsePointsSupported2( const Volume :Str) :Bool;
16409:  Func NtfsFileHasReparsePoint2( const Path :Str) :Bool;
16410:  Func NtfsIsFolderMountPoint2( const Path :Str) :Bool;
16411:  Func NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) :Bool;
16412:  Func NtfsMountVolume2(const Volume:WideChar;const MountPoint:WideString) :Bool;
16413:   TypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter );
16414:  Func NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) :Bool;
16415:  Func NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) :Bool;
16416:  Func NtfsOpLockBreakAcknowledge2(Handle:THandle; Overlapped: TOverlapped) :Bool;
16417:  Func NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) :Bool;
16418:  Func NtfsRequestOpLock2(Handle:THandle; Kind:TOpLock; Overlapped: TOverlapped):Boolean;
16419:  Func NtfsCreateJunctionPoint2( const Source, Destination :Str) :Bool;
16420:  Func NtfsDeleteJunctionPoint2( const Source :Str) :Bool;
16421:  Func NtfsGetJunctionPointDestination2(const Source:str;var Destination:Str):Bool;
16422:   TypeS(TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
16423:    +urity,siAlternate,siHardLink,siProperty, siObjectIdentifier,siReparsePoints,siSparseFile);
16424:   TypeS(TStreamIds', 'set of TStreamId;
16425:   TInternalFindStreamData','record FileHandle:THandle;Context:TObject; StreamIds:TStreamIds;end;
16426:   TypeS(TFindStreamData', 'record Internal : TInternalFindStreamData; At'
16427:   +tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end;
16428:  Func NtfsFindFirstStream2(const FileName:str;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
16429:  Func NtfsFindNextStream2( var Data : TFindStreamData) :Bool;
16430:  Func NtfsFindStreamClose2( var Data : TFindStreamData) :Bool;
16431:  Func NtfsCreateHardLink2( const LinkFileName, ExistingFileName :Str) :Bool;
16432:  Func NtfsCreateHardLinkA2(const LinkFileName,ExistingFileName:Ansistr):Bool;
16433:  Func NtfsCreateHardLinkW2(const LinkFileName,ExistingFileName: WideString):Bool;
16434:   TypeS('TNtfsHardLinkInfo', 'record LinkCount :Card; FileIndex : Int64; end;
16435:  Func NtfsGetHardLinkInfo2(const FileName:str;var Info:TNtfsHardLinkInfo):Boolean;
16436:  Func NtfsFindHardLinks2(const Path:str;const FileIdxHigh,FIdxLow:Card;const List:TStrings):Bool
16437:  Func NtfsDeleteHardLinks2( const FileName :Str) :Bool;
16438:   FindClass('TOBJECT'),'EJclFileSummaryError;
16439:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite );
16440:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll );
16441:   TJclFileSummaryPropSetCallback', 'Func ( const FMTID : TGUID) :Bool;
16442:   ClassN(CL.FindClass('TOBJECT'),'TJclFileSummary;
16443:   SIRegister_TJclFilePropertySet(CL);
16444:   //TypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet;
16445:   SIRegister_TJclFileSummary(CL);
16446:   SIRegister_TJclFileSummaryInformation(CL);
16447:   SIRegister_TJclDocSummaryInformation(CL);
16448:   SIRegister_TJclMediaFileSummaryInformation(CL);
16449:   SIRegister_TJclMSISummaryInformation(CL);
16450:   SIRegister_TJclShellSummaryInformation(CL);
16451:   SIRegister_TJclStorageSummaryInformation(CL);
16452:   SIRegister_TJclImageSummaryInformation(CL);
16453:   SIRegister_TJclDisplacedSummaryInformation(CL);
16454:   SIRegister_TJclBriefCaseSummaryInformation(CL);
16455:   SIRegister_TJclMiscSummaryInformation(CL);
16456:   SIRegister_TJclWebViewSummaryInformation(CL);
16457:   SIRegister_TJclMusicSummaryInformation(CL);
16458:   SIRegister_TJclDRMSummaryInformation(CL);
16459:   SIRegister_TJclVideoSummaryInformation(CL);
16460:   SIRegister_TJclAudioSummaryInformation(CL);
16461:   SIRegister_TJclControlPanelSummaryInformation(CL);
16462:   SIRegister_TJclVolumeSummaryInformation(CL);
16463:   SIRegister_TJclShareSummaryInformation(CL);
16464:   SIRegister_TJclLinkSummaryInformation(CL);
16465:   SIRegister_TJclQuerySummaryInformation(CL);
16466:   SIRegister_TJclImageInformation(CL);
16467:   SIRegister_TJclJpegSummaryInformation(CL);
16468: end;
16469:
16470: Proc SIRegister_Jcl8087(CL: TPSPascalCompiler);
16471: begin
16472:   AddTypeS('T8087Precision','( pcSingle, pcReserved, pcDouble, pcExtended );
```

```
16473:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate );
16474:   T8087Infinity','( icProjective, icAffine );
16475:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
16476:   TypeS('T8087Exceptions', set of T8087Exception;
16477:  Func Get8087ControlWord : Word;
16478:  Func Get8087Infinity : T8087Infinity;
16479:  Func Get8087Precision : T8087Precision;
16480:  Func Get8087Rounding : T8087Rounding;
16481:  Func Get8087StatusWord( ClearExceptions :Bool) : Word;
16482:  Func Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity;
16483:  Func Set8087Precision( const Precision : T8087Precision) : T8087Precision;
16484:  Func Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding;
16485:  Func Set8087ControlWord( const Control : Word) : Word;
16486:  Func ClearPending8087Exceptions : T8087Exceptions;
16487:  Func GetPending8087Exceptions : T8087Exceptions;
16488:  Func GetMasked8087Exceptions : T8087Exceptions;
16489:  Func SetMasked8087Exceptions(Exceptions:T8087Exceptions;ClearBefore:Bool):T8087Exceptions;
16490:  Func Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions;
16491:  Func Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore:Bool):T8087Exceptions;
16492: end;
16493:
16494: Proc SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
16495: begin
16496:  Proc BoxMoveSelectedItems( SrcList, DstList : TWinControl);
16497:  Proc BoxMoveAllItems( SrcList, DstList : TWinControl);
16498:  Proc BoxDragOver(List:TWinControl;Sorce:TObject;X,Y:Int;State:TDragState;var Accept:Bool;Sorted:Bool;
16499:  Proc BoxMoveFocusedItem( List : TWinControl; DstIndex : Int);
16500:  Proc BoxMoveSelected( List : TWinControl; Items : TStrings);
16501:  Proc BoxSetItem( List : TWinControl; Index : Int);
16502:  Func BoxGetFirstSelection( List : TWinControl) : Int;
16503:  Func BoxCanDropItem(List:TWinControl; X,Y:Int; var DragIndex : Int) :Bool;
16504: end;
16505:
16506: Proc SIRegister_UrlMon(CL: TPSPascalCompiler);
16507: begin
16508:  //ConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context;
16509:  //ConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee;
16510:  ConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
16511:   type ULONG', 'Cardinal; LPCWSTR', 'PChar;  TypeS('LPWSTR', 'PChar;
16512:   LPSTR', 'PChar;
16513:   TBindVerb', 'ULONG;
16514:   TBindInfoF', 'ULONG;
16515:   TBindF', 'ULONG;
16516:   TBSCF', 'ULONG;
16517:   TBindStatus', 'ULONG;
16518:   TCIPStatus', 'ULONG;
16519:   TBindString', 'ULONG;
16520:   TPiFlags', 'ULONG;
16521:   TOIBdgFlags', 'ULONG;
16522:   TParseAction', 'ULONG;
16523:   TPSUAction', 'ULONG;
16524:   TQueryOption', 'ULONG;
16525:   TPUAF', 'ULONG;
16526:   TSZMFlags', 'ULONG;
16527:   TUrlZone', 'ULONG;
16528:   TUrlTemplate', 'ULONG;
16529:   TZAFlags', 'ULONG;
16530:   TUrlZoneReg', 'ULONG;
16531:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
16532:  ConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
16533:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
16534:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
16535:  const 'CF_NULL','LongInt').SetInt( 0);
16536:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
16537:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain;
16538:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext;
16539:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap;
16540:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript;
16541:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff;
16542:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic;
16543:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav;
16544:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav;
16545:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif;
16546:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg;
16547:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg;
16548:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff;
16549:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png;
16550:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp;
16551:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg;
16552:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf;
16553:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf;
16554:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi;
16555:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg;
16556:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals;
16557:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream;
16558:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream;
16559:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf;
16560:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff;
16561:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio;
```

```
16562:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm;
16563:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime;
16564:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo;
16565:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie;
16566:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html;
16567:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
16568:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
16569:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
16570:   Interface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding;
16571:   SIRegister_IPersistMoniker(CL);
16572:   SIRegister_IBindProtocol(CL);
16573:   SIRegister_IBinding(CL);
16574:  const BINDVERB_GET','LongWord').SetUInt( $00000000);
16575:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
16576:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
16577:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
16578:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
16579:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
16580:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
16581:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
16582:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
16583:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
16584:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
16585:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
16586:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16587:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16588:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16589:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
16590:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
16591:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
16592:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
16593:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
16594:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
16595:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
16596:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
16597:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
16598:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
16599:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
16600:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
16601:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16602:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16603:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16604:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16605:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
16606:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16607:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16608:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16609:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16610:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16611:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16612:  const BINDSTATUS_BEGINDOWNLOADCOMPONENTS,LongInt).SetInt(BINDSTATUS_ENDDOWNLOADDATA + 1);
16613:  const 'BINDSTATUS_INSTALLINGCOMPONENTS',LongInt).SetInt(BINDSTATUS_BEGINDOWNLOADCOMPONENTS+1);
16614:  const BINDSTATUS_ENDDOWNLOADCOMPONENTS,LongInt).SetInt(BINDSTATUS_INSTALLINGCOMPONENTS+1);
16615:  const BINDSTATUS_USINGCACHEDCOPY,LongInt).SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16616:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16617:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16618:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16619:  const BINDSTATUS_CACHEFILENAMEAVAILABLE,LongInt).SetInt(BINDSTATUS_MIMETYPEAVAILABLE+1);
16620:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt(BINDSTATUS_CACHEFILENAMEAVAILABLE+1);
16621:  const BINDSTATUS_ENDSYNCOPERATION',LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16622:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16623:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16624:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16625:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16626:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16627:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16628:  const BINDSTATUS_CLASSINSTALLLOCATION,LongInt).SetInt(BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE+1);
16629:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16630:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16631:  const BINDSTATUS_CONTENTDISPOSITIONATTACH,LongInt).SetInt(BINDSTATUS_LOADINGMIMEHANDLER+1);
16632:  const 'BINDSTATUS_FILTERREPORTMIMETYPE',LongInt).SetInt(BINDSTAT_CONTENTDISPOSITIONATTACH+1);
16633:  const BINDSTATUS_CLSIDCANINSTANTIATE,LongInt).SetInt(BINDSTATUS_FILTERREPORTMIMETYPE+ 1);
16634:  const BINDSTATUS_IUNKNOWNAVAILABLE,LongInt).SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16635:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16636:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16637:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16638:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16639:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16640:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt(BINDSTATUS_COOKIE_SENT + 1);
16641:  const BINDSTATUS_COOKIE_SUPPRESSED,LongInt).SetInt(BINDSTATUS_COMPACT_POLICY_RECEIVED+1);
16642:  const BINDSTATUS_COOKIE_STATE_UNKNOWN,LongInt).SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16643:  const BINDSTATUS_COOKIE_STATE_ACCEPT,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_UNKNOWN+ 1);
16644:  const BINDSTATUS_COOKIE_STATE_REJECT,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16645:  const BINDSTATUS_COOKIE_STATE_PROMPT,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_REJECT + 1);
16646:  const BINDSTATUS_COOKIE_STATE_LEASH,LongInt).SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16647:  const BINDSTATUS_COOKIE_STATE_DOWNGRADE,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_LEASH+1);
16648:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16649:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16650:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
```

```
16651: const'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED',LongInt).SetInt(BINDSTATUS_SESSION_COOKIE_RECEIVED+1);
16652: const'BINDSTATUS_SESSION_COOKIES_ALLOWED',LongInt).SetInt(BINDSTATUS_PERSISTENT_COOKIE_RECEIVED+1);
16653:  const BINDSTATUS_CACHECONTROL,LongInt).SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16654:  const BINDSTATUS_CONTENTDISPOSITIONFILENAME,LongInt).SetInt(BINDSTATUS_CACHECONTROL + 1);
16655: const'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt(BINDSTATUS_CONTENTDISPOSITIONFILENAME+1);
16656:  const BINDSTATUS_PUBLISHERAVAILABLE,LongInt).SetInt(BINDSTATUS_MIMETEXTPLAINMISMATCH+1);
16657:  const BINDSTATUS_DISPLAYNAMEAVAILABLE,LongInt).SetInt(BINDSTATUS_PUBLISHERAVAILABLE + 1);
16658:  // PBindInfo', '^TBindInfo // will not work;
16659:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16660:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16661:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16662:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16663:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end;
16664:   TBindInfo', '_tagBINDINFO;
16665:   BINDINFO', '_tagBINDINFO;}
16666:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16667:    +'criptor : DWORD; bInheritHandle : BOOL; end;
16668:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES;
16669:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES;
16670:   //PRemBindInfo', '^TRemBindInfo // will not work;
16671:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16672:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16673:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16674:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16675:    +'n; dwReserved : DWORD; end;
16676:   TRemBindInfo', '_tagRemBINDINFO;
16677:   RemBINDINFO', '_tagRemBINDINFO;}
16678:   //PRemFormatEtc', '^TRemFormatEtc // will not work;
16679: tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end;
16680:   TRemFormatEtc', 'tagRemFORMATETC;
16681:   RemFORMATETC', 'tagRemFORMATETC;
16682:   SIRegister_IBindStatusCallback(CL);
16683:   SIRegister_IAuthenticate(CL);
16684:   SIRegister_IHttpNegotiate(CL);
16685:   SIRegister_IWindowForBindingUI(CL);
16686:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16687:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16688:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16689:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16690:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16691:  const CIP_TRUST_VERIFICATION_COMPONENT_MISSING,LongInt).SetInt( CIP_NAME_CONFLICT + 1);
16692:  const CIP_EXE_SELF_REGISTERATION_TIMEOUT,LongInt(CIP_TRUST_VERIFICATION_COMPONET_MISSING+1);
16693:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16694:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16695:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16696:   SIRegister_ICodeInstall(CL);
16697:   SIRegister_IWinInetInfo(CL);
16698:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16699:   SIRegister_IHttpSecurity(CL);
16700:   SIRegister_IWinInetHttpInfo(CL);
16701:   SIRegister_IBindHost(CL);
16702:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16703:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16704:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16705:  Func URLOpenStream(p1:IUnknown;p2:PChar; p3:DWORD; p4:IBindStatusCallback): HResult;
16706:  Func URLOpenPullStream(p1:IUnknown;p2:PChar;p3:DWORD; BSC:IBindStatusCallback):HResult;
16707:  Func
       URLDownloadToFile(Caller:IUnknown;URL:PChar;FileName:PChar;Reserved:DWORD;StatusCB:IBindStatusCallback):
       HResult;
16708:  Func URLDownloadToCacheFile(p1:IUnknown;p2:PChar;p3:PChar;p4:DWORD;p5:DWORD;p6:IBindStatusCallback):
       HResult;
16709:  Func URLOpenBlockingStream(p1:IUnknown;p2:PChar;out p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult;
16710:  Func HlinkGoBack( unk : IUnknown) : HResult;
16711:  Func HlinkGoForward( unk : IUnknown) : HResult;
16712:  Func HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult;
16713: // Func HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult;
16714:  SIRegister_IInternet(CL);
16715:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16716:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16717:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16718:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16719:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16720:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16721:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16722:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16723:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16724:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16725:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16726:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16727:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16728:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16729:   //POLEStrArray', '^TOLESTRArray // will not work;
16730:   SIRegister_IInternetBindInfo(CL);
16731:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16732:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16733:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16734:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16735:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16736:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
```

```
16737:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16738:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16739:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16740:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16741:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16742:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16743:   //PProtocolData', '^TProtocolData // will not work;
16744:   _tagPROTOCOLDATA,record grfFlags:DWORD; dwState:DWORD; pData:TObject;cbData:ULONG; end;
16745:   TProtocolData', '_tagPROTOCOLDATA;
16746:   PROTOCOLDATA', '_tagPROTOCOLDATA;
16747:   Interface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink,'IInternetProtocolSink;
16748:   SIRegister_IInternetProtocolRoot(CL);
16749:   SIRegister_IInternetProtocol(CL);
16750:   SIRegister_IInternetProtocolSink(CL);
16751:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16752:   SIRegister_IInternetSession(CL);
16753:   SIRegister_IInternetThreadSwitch(CL);
16754:   SIRegister_IInternetPriority(CL);
16755:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16756:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16757:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16758:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16759:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16760:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16761:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16762:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16763:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16764:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16765:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16766:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16767:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16768:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16769:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16770:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16771:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16772:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16773:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16774:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16775:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16776:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16777:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16778:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16779:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16780:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16781:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16782:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16783:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16784:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16785:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16786:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16787:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16788:   SIRegister_IInternetProtocolInfo(CL);
16789:   IOInet', 'IInternet;
16790:   IOInetBindInfo', 'IInternetBindInfo;
16791:   IOInetProtocolRoot', 'IInternetProtocolRoot;
16792:   IOInetProtocol', 'IInternetProtocol;
16793:   IOInetProtocolSink', 'IInternetProtocolSink;
16794:   IOInetProtocolInfo', 'IInternetProtocolInfo;
16795:   IOInetSession', 'IInternetSession;
16796:   IOInetPriority', 'IInternetPriority;
16797:   IOInetThreadSwitch', 'IInternetThreadSwitch;
16798:  Func
       CoInternetParseUrl(pwzUrl:LPCWSTR;ParseAction:TParseAction;dwFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD;
       var pcchResult:DWORD;dwReserved:DWORD):HResult;
16799:  Func CoInternetCombineUrl(pwzBaseUrl,
       pwzRelativeUrl:LPCWSTR;dwCombineFlags:DWORD;pszResult:LPWSTR;cchResult: DWORD;var
       pcchResult:DWORD;dwReserved:DWORD):HResult;
16800:  Func CoInternetCompareUrl(pwzUrl1, pwzUrl2 LPCWSTR;dwFlags DWORD): HResult;
16801:  Func CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD;dwReserved:DWORD):HResult;
16802:  Func
       CoInternetQueryInfo(pwzUrl:LPCWSTR;QueryOptions:TQueryOption;dwQueryFlags:DWORD;pvBuffer:TObject;cbBuffer
       :DWORD;var pcbBuffer:DWORD;dwReserved DWORD):HResult;
16803:  Func CoInternetGetSession(dwSessionMode:DWORD;var
       pIInternetSession:IInternetSes;dwReserved:DWORD):HResult;
16804:  Func CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16805:  Func
       OInetParseUrl(pwzUrl:LPCWSTR;ParseAction:TParseAction;dwFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD; var
       pcchResult:DWORD;dwReserved:DWORD):HResult;
16806:  Func OInetCombineUrl(pwzBaseUrl,
       pwzRelativeUrl:LPCWSTR;dwCombineFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD;var
       pcchResult:DWORD;dwReserved:DWORD):HResult;
16807:  Func OInetCompareUrl(pwzUrl1,pwzUrl2  LPCWSTR;dwFlags: DWORD): Hresult;
16808:  Func OInetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags:DWORD;dwReserved:DWORD): HResult;
16809:  Func
       OInetQueryInfo(pwzUrl:LPCWSTR;QueryOptions:TQueryOption;dwQueryFlags:DWORD;pvBuffer:TObject;cbBuffer: DWORD;var
       pcbBuffer:DWORD;dwReserved:DWORD):HResult;
16810:  Func OInetGetSession(dwSessionMode:DWORD;var pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16811:   //Func CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult;
```

```
16812:  //Proc ReleaseBindInfo( const bindinfo : TBindInfo);
16813: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16814: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16815:  //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16816:  //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16817:  //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16818:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16819:   SIRegister_IInternetSecurityMgrSite(CL);
16820:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16821:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16822:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16823:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16824:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16825:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16826:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16827:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16828:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16829:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16830:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16831:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16832:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16833:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16834:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16835:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16836:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16837:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16838:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16839:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16840:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16841:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16842:  const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16843:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16844:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16845:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16846:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16847:   SIRegister_IInternetSecurityManager(CL);
16848:   SIRegister_IInternetHostSecurityManager(CL);
16849:   SIRegister_IInternetSecurityManagerEx(CL);
16850:  const URLACTION_MIN','LongWord').SetUInt( $00001000);
16851:  const URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16852:  const URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16853:  const URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16854:  const URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16855:  const URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16856:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16857:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16858:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16859:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16860:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16861:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16862:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16863:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16864:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16865:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16866:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16867:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16868:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16869:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16870:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16871:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16872:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16873:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16874:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16875:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16876:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16877:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16878:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16879:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16880:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16881:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16882:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16883:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16884:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16885:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16886:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16887:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16888:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16889:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16890:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16891:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16892:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16893:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16894:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16895:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16896:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16897:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16898:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16899:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16900:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
```

```
const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
Func GetUrlPolicyPermissions( dw : DWORD) : DWORD;
Func SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD;
const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
 //PZoneAttributes', '^TZoneAttributes // will not work;
_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;
  { _ZONEATTRIBUTES = packed record
    cbSize: ULONG;
    szDisplayName: array [0..260 - 1] of WideChar;
    szDescription: array [0..200 - 1] of WideChar;
    szIconPath: array [0..260 - 1] of WideChar;
    dwTemplateMinLevel: DWORD;
    dwTemplateRecommended: DWORD;
    dwTemplateCurrentLevel: DWORD;
```

```
16988:     dwFlags: DWORD;
16989:   end; }
16990:   TZoneAttributes', '_ZONEATTRIBUTES;
16991:   ZONEATTRIBUTES', '_ZONEATTRIBUTES;
16992:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16993:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16994:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16995:   SIRegister_IInternetZoneManager(CL);
16996:   SIRegister_IInternetZoneManagerEx(CL);
16997:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16998:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16999:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
17000:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
17001:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
17002:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
17003:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
17004:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
17005:   //PCodeBaseHold', '^TCodeBaseHold // will not work;
17006:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
17007:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle:DWORD; end;
17008:   TCodeBaseHold', '_tagCODEBASEHOLD;
17009:   CODEBASEHOLD', '_tagCODEBASEHOLD;
17010:   //PSoftDistInfo', '^TSoftDistInfo // will not work;
17011:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
17012:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
17013:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
17014:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
17015:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end;
17016:   TSoftDistInfo', '_tagSOFTDISTINFO;
17017:   SOFTDISTINFO', '_tagSOFTDISTINFO;
17018:   SIRegister_ISoftDistExt(CL);
17019:  Func GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult;
17020:  Func SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
       dwAdvertisedVersionLS : DWORD) : HResult;
17021:   SIRegister_IDataFilter(CL);
17022:  // PProtocolFilterData', '^TProtocolFilterData // will not work;
17023:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
17024:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
17025:    +'terFlags : DWORD; end;
17026:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA;
17027:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA;
17028:   //PDataInfo', '^TDataInfo // will not work;
17029:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
17030:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end;
17031:   TDataInfo', '_tagDATAINFO;
17032:   DATAINFO', '_tagDATAINFO;
17033:   SIRegister_IEncodingFilterFactory(CL);
17034:  Func IsLoggingEnabled( pszUrl : PChar) : BOOL;
17035:  //Func IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL;
17036:  //Func IsLoggingEnabledW( pszUrl : PWideChar) : BOOL;
17037:   //PHitLoggingInfo', '^THitLoggingInfo // will not work;
17038:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
17039:    +rlName:LPSTR;StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo:LPSTR; end;
17040:   THitLoggingInfo', '_tagHIT_LOGGING_INFO;
17041:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO;
17042:  Func WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL;
17043: end;
17044:
17045: Proc SIRegister_DFFUtils(CL: TPSPascalCompiler);
17046: begin
17047:  Proc reformatMemo( const m : TCustomMemo);
17048:  Proc SetMemoMargins( m : TCustomMemo; const L, T, R, B : Int);
17049:  Proc MoveToTop( memo : TMemo);
17050:  Proc ScrollToTop( memo : TMemo);
17051:  Func LineNumberClicked( memo : TMemo) : Int;
17052:  Func MemoClickedLine( memo : TMemo) : Int;
17053:  Func ClickedMemoLine( memo : TMemo) : Int;
17054:  Func MemoLineClicked( memo : TMemo) : Int;
17055:  Func LinePositionClicked( Memo : TMemo) : Int;
17056:  Func ClickedMemoPosition( memo : TMemo) : Int;
17057:  Func MemoPositionClicked( memo : TMemo) : Int;
17058:  Proc AdjustGridSize( grid : TDrawGrid);
17059:  Proc DeleteGridRow( Grid : TStringGrid; const ARow : Int);
17060:  Proc InsertgridRow( Grid : TStringGrid; const ARow : Int);
17061:  Proc Sortgrid( Grid : TStringGrid; const SortCol : Int);;
17062:  Proc Sortgrid1(Grid:TStringGrid; const SortCol : Int; sortascending :Bool);;
17063:  Proc sortstrDown( var s :Str);
17064:  Proc sortstrUp( var s :Str);
17065:  Proc rotatestrleft( var s :Str);
17066:  Func dffstrtofloatdef( s :Str; default : extended) : extended;
17067:  Func deblank( s :Str) :Str;
17068:  Func IntToBinaryString( const n : Int; MinLength : Int) :Str;
17069:  Proc FreeAndClearListBox( C : TListBox);;
17070:  Proc FreeAndClearMemo( C : TMemo);;
17071:  Proc FreeAndClearStringList( C : TStringList);;
17072:  Func dffgetfilesize( f : TSearchrec) : int64;
17073: end;
17074:
17075: Proc SIRegister_MathsLib(CL: TPSPascalCompiler);
```

```
17076: begin
17077:   TypeS('intset', set of byte;
17078:   TPoint64', 'record x : int64; y : int64; end;
17079:  Func GetNextPandigital( size : Int; var Digits : array of Int) :Bool;
17080:  Func IsPolygonal( T : int64; var rank : array of Int) :Bool;
17081:  Func GeneratePentagon( n : Int) : Int;
17082:  Func IsPentagon( p : Int) :Bool;
17083:  Func isSquare( const N : int64) :Bool;
17084:  Func isCube( const N : int64) :Bool;
17085:  Func isPalindrome( const n : int64) :Bool;;
17086:  Func isPalindrome1( const n : int64; var len : Int) :Bool;;
17087:  Func GetEulerPhi( n : int64) : int64;
17088:  Func dffIntPower( a, b : int64) : int64;;
17089:  Func IntPower1( a : extended; b : int64) : extended;;
17090:  Func gcd2( a, b : int64) : int64;
17091:  Func GCDMany( A : array of Int) : Int;
17092:  Func LCMMany( A : array of Int) : Int;
17093:  Proc ContinuedFraction(A: array of int64;const wholepart:int;var numerator,denominator:int64);
17094:  Func dffFactorial( n : int64) : int64;
17095:  Func digitcount( n : int64) : Int;
17096:  Func nextpermute( var a : array of Int) :Bool;
17097:  Func convertfloattofractionstring(N:extended;maxdenom:Int;multipleof:boolean):str;
17098:  Func convertStringToDecimal( s :Str; var n : extended) :Bool;
17099:  Func InttoBinaryStr( nn : Int) :Str;
17100:  Func StrtoAngle( const s :Str; var angle : extended) :Bool;
17101:  Func AngleToStr( angle : extended) :Str;
17102:  Func AngleToStr2( angle : extended; decimal:boolean) :Str;
17103:  Func deg2rad( deg : extended) : extended;
17104:  Func rad2deg( rad : extended) : extended;
17105:  Func GetLongToMercProjection( const long : extended) : extended;
17106:  Func GetLatToMercProjection( const Lat : Extended) : Extended;
17107:  Func GetMercProjectionToLong( const ProjLong : extended) : extended;
17108:  Func GetMercProjectionToLat( const ProjLat : extended) : extended;
17109:   SIRegister_TPrimes(CL);
17110:   //RIRegister_TPrimes(CL);
17111:  //ConstantN('deg','LongInt').SetInt( char( 176));
17112:  ConstantN('minmark','LongInt').SetInt(( 180));
17113:  Func Random64( const N : Int64) : Int64;;
17114:  Proc Randomize64;
17115:  Func Random641 : extended;;
17116:  Proc GetParens( Variables:Str; OpChar:char; var list:TStringlist)//DFFUtils
17117:  add 4.7.4.62
17118:  Func MercScaling(Long0,lat0, long1, lat1:extended; x0,y0,x1,y1:integer) :TMercScalingRec');
17119:  Func LonglatToPlotPt(Long,lat:Extended; Scalerec:TMercScalingrec):TPoint');
17120:  Func PlotPtToLonglat(PlotPt:TPoint; Scalerec:TMercScalingrec):TRealPoint');
17121:  Func SphericalEarthDistance(lat1,lon1,lat2,lon2:extended; Units:integer):extended');
17122: end;
17123:
17124: Proc SIRegister_UGeometry(CL: TPSPascalCompiler);
17125: begin
17126:   TrealPoint', 'record x : extended; y : extended; end;
17127:   Tline', 'record p1 : TPoint; p2 : TPoint; end;
17128:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end;
17129:   TCircle', 'record cx : Int; cy : Int; r : Int; end;
17130:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end;
17131:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError );
17132:  Func realpoint( x, y : extended) : TRealPoint;
17133:  Func dist( const p1, p2 : TrealPoint) : extended;
17134:  Func intdist( const p1, p2 : TPoint) : Int;
17135:  Func dffLine( const p1, p2 : TPoint) : Tline;;
17136:  Func Line1( const p1, p2 : TRealPoint) : TRealline;;
17137:  Func dffCircle( const cx, cy, R : Int) : TCircle;;
17138:  Func Circle1( const cx, cy, R : extended) : TRealCircle;;
17139:  Func GetTheta( const L : TLine) : extended;;
17140:  Func GetTheta1( const p1, p2 : TPoint) : extended;;
17141:  Func GetTheta2( const p1, p2 : TRealPoint) : extended;;
17142:  Proc Extendline( var L : TLine; dist : Int);;
17143:  Proc Extendline1( var L : TRealLine; dist : extended);;
17144:  Func Linesintersect( line1, line2 : TLine) :Bool;
17145:  Func ExtendedLinesIntersect(Line1,Line2:TLine;const extendlines:bool;var IP:TPoint):bool;
17146:  Func ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
17147:  Func Intersect(L1,L2: TLine; var pointonborder:boolean; var IP : TPoint) :Bool;
17148:  Func PointPerpendicularLine( L : TLine; P : TPoint) : TLine;
17149:  Func PerpDistance( L : TLine; P : TPoint) : Int;
17150:  Func AngledLineFromLine(L:TLine;P:TPoint;Dist:extended; alpha : extended) : TLine;;
17151:  Func AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordiates:bool):TLine;
17152:  Func PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult;
17153:  Func PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var Clockwise:bool):Int;
17154:  Proc InflatePolygon(const points:array of Tpoint;var points2:array of TPoint;var area:Int;const
       screenCoordinates:bool; const inflateby : Int);
17155:  Func PolyBuiltClockwise(const points:array of TPoint;const screencoordies:bool):bool;
17156:  Func DegtoRad( d : extended) : extended;
17157:  Func RadtoDeg( r : extended) : extended;
17158:  Proc TranslateLeftTo( var L : TLine; newend : TPoint);;
17159:  Proc TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);;
17160:  Proc RotateRightEndBy( var L : TLine; alpha : extended);
17161:  Proc RotateRightEndTo( var L : TLine; alpha : extended);;
17162:  Proc RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);;
17163:  Func CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) :Bool;;
```

```
17164:  Func CircleCircleIntersect1(c1,c2:TRealCircle; var IP1,Ip2  TRealPoint) :Bool;;
17165:  Func PointCircleTangentLines(const C:TCircle;const P:TPoint;var L1,L2:TLine):bool;
17166: Func CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
17167: end;
17168:
17169:
17170: Proc SIRegister_UAstronomy(CL: TPSPascalCompiler);
17171: begin
17172:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat );
17173:   TDTType', '( ttLocal, ttUT, ttGST, ttLST );
17174:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end;
17175:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl:TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
17176:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
17177:    + TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
17178:    +arth : extended; Phase : extended; end;
17179:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
17180:    +Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end;
17181:   TEclipseRec', 'record Msg :Str; Status : Int; FirstConta'
17182:    +ct: TDatetime;LastContact:TDateTime;Magnitude:Extended;MaxeclipseUTime:TDateTime;end;
17183:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO );
17184:   TPlanetRec', 'record AsOf : TDateTime; Name :Str; MeanLon : '
17185:    +extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
17186:    +ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
17187:    +jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end;
17188:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended;
       UncorrectedEarthDistance:extended;GeoEclLonLat:TRpoint;CorrectedEarthDistance:extended;ApparentRaDecl:TRPoint;end;
17189:   SIRegister_TAstronomy(CL);
17190:  Func AngleToStr( angle : extended) :Str;
17191:  Func AngleToStr2( angle : extended; decimal:boolean) :Str;
17192:  Func StrToAngle( s :Str; var angle : extended) :Bool;
17193:  Func HoursToStr24( t : extended) :Str;
17194:  Func RPoint( x, y : extended) : TRPoint;
17195:  Func getStimename( t : TDTType) :Str;
17196: end;
17197:
17198: Proc SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
17199: begin
17200:   TCardValue', 'Int;
17201:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts );
17202:   TShortSuit', '( cardS, cardD, cardC, cardH );
17203: Type(TDecks,Stand1,Standrd2,Fises1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
17204:   SIRegister_TCard(CL);
17205:   SIRegister_TDeck(CL);
17206: end;
17207:
17208: Proc SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
17209: begin
17210:   tMethodCall', 'Procedure;tVerboseCall', 'Proc ( s :Str);
17211:  // PTEdge', '^TEdge // will not work;
17212:   TEdge', 'record FromNodeIndex : Int; ToNodeIndex : Int; '
17213:    +Weight : Int; work : Int; Len : Int; Highlight :Bool; end;
17214:   SIRegister_TNode(CL);
17215:   SIRegister_TGraphList(CL);
17216: end;
17217:
17218: Proc SIRegister_UParser10(CL: TPSPascalCompiler);
17219: begin
17220:   ParserFloat', 'extended;
17221:   //PParserFloat', '^ParserFloat /will not work;
17222:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
17223:    +ulo, IntDiv,IntDIVZ,Intpower,realpower,square,third,fourth,FuncOneVar,FuncTwoVar );
17224:   //POperation', '^TOperation // will not work;
17225:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
17226:    +est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
17227:    +; Token : TDFFToken; end;
17228:   TMathProcedure','procedure(AnOperation: TDFFOperation);
17229:   (TOBJECT'),'EMathParserError;
17230:   TOBJECT'),'ESyntaxError;
17231:   'TOBJECT'),'EExpressionHasBlanks;
17232:   ('TOBJECT'),'EExpressionTooComplex;
17233:   (TOBJECT'),'ETooManyNestings;
17234:   ('TOBJECT'),'EMissMatchingBracket;
17235:   (TOBJECT'),'EBadName;
17236:   (CL.FindClass('TOBJECT'),'EParserInternalError;
17237:   (TExParserExceptionEvent', 'Proc ( Sender : TObject; E : Exception);
17238:   SIRegister_TCustomParser(CL);
17239:   SIRegister_TExParser(CL);
17240: end;
17241:
17242: Func isService:Bool;
17243: begin
17244:   result:= NOT(Application is TApplication);
17245:   {result:= Application is TServiceApplication;}
17246: end;
17247: Func isApplication:Bool;
17248: begin
```

```
17249:   result:= Application is TApplication;
17250: end;
17251: //SM_REMOTESESSION = $1000
17252: Func isTerminalSession:Bool;
17253: begin
17254:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
17255: end;
17256: writeln(inttostr(getmemoryload))
17257:
17258: Proc SIRegister_cyIEUtils(CL: TPSPascalCompiler);
17259: begin
17260:  TypeS('TwbPageSetup', record font :Str; footer :Str; header : '
17261:    +String; margin_bottom :Str; margin_left :Str; margin_right : Strin'
17262:    +g; margin_top :Str; Print_Background :Str; Shrink_To_Fit:str; end;
17263:  Func cyURLEncode( const S :Str) :Str;
17264:  Func MakeResourceURL(const ModulName:str;const ResName:PChar;const ResType:PChar):str;
17265:  Func MakeResourceURL1(const Modul:HMODULE;const ResName:PChar;const ResType:PChar):str;
17266:  Func cyColorToHtml( aColor : TColor) :Str;
17267:  Func HtmlToColor( aHtmlColor :Str) : TColor;
17268:  //Func GetStreamEncoding( aStream : TStream) : TEncoding;
17269:  // Func IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) :Bool;
17270:  Func AddHtmlUnicodePrefix( aHtml :Str) :Str;
17271:  Func RemoveHtmlUnicodePrefix( aHtml :Str) :Str;
17272:  Proc GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup);
17273:  Proc SetPageSetupToRegistry( awbPageSetup : TwbPageSetup);
17274:  ConstantN('IEBodyBorderless','String').SetString( 'none;
17275:   IEBodySingleBorder','String').SetString( ';
17276:   IEDesignModeOn','String').SetString( 'On;
17277:   IEDesignModeOff','String').SetString( 'Off;
17278:   cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
17279:   cHtmlUnicodePrefixChar','Char').SetString( #$FE);
17280: end;
17281:
17282:
17283: Proc SIRegister_UcomboV2(CL: TPSPascalCompiler);
17284: begin
17285:  ConstantN('UMaxEntries','LongInt').SetInt( 600);
17286:   TypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
17287:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
17288:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
17289:    +'inationsRepeat, CombinationsRepeatDown );
17290:   TypeS('TByteArray64', 'array[0..600 + 1] of int64;;
17291:   SIRegister_TComboSet(CL);
17292: end;
17293:
17294:  Proc SIRegister_cyBaseComm(CL: TPSPascalCompiler);
17295: begin
17296:   TcyCommandType', '( ctDeveloperDefined, ctUserDefined );
17297:   TStreamContentType', '( scDeveloperDefined, scUserDefined, scString );
17298:   TProcOnReceiveCommand', 'Proc (Sender: TObject; aCommand: Word; userParam : Int);
17299:   TProcOnReceiveString', 'Proc ( Sender : TObject; fromBaseCo'
17300:    +'mmHandle : THandle; aString :Str; userParam : Int);
17301:   TProcOnReceiveMemoryStream', 'Proc ( Sender : TObject; from'
17302:    +BaseCommHandle : THandle; aStream : TMemoryStream; userParam: Int);
17303:   SIRegister_TcyBaseComm(CL);
17304:  ConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
17305:  ConstantN('MsgResultOk','LongInt').SetInt( 99);
17306:  Func ValidateFileMappingName( aName :Str) :Str;
17307:  Proc makeCaption(leftSide, Rightside:str; form:TForm);
17308: end;
17309:
17310: Proc SIRegister_cyDERUtils(CL: TPSPascalCompiler);
17311: begin
17312:   TypeS('DERString', 'String;
17313:   TypeS('DERChar', 'Char;
17314:   TypeS('TElementsType', ( etText, etExpressionKeyWord, etNumbers, etInt'
17315:    +eger,etFloat,etPercentage,etwebSite,etWebMail,etMoney,etDate,etTextLine,etParagraph);
17316:   TypeS('TElementsTypes', 'set of TElementsType;
17317:   TypeS('DERNString', 'String;
17318:  const DERDecimalSeparator','String').SetString( '.;
17319:  const DERDefaultChars,String)(+°@/%-_.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ;
17320:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz;
17321:  Func isValidWebSiteChar( aChar : Char) :Bool;
17322:  Func isValidWebMailChar( aChar : Char) :Bool;
17323:  Func isValidwebSite( aStr :Str) :Bool;
17324:  Func isValidWebMail( aStr :Str) :Bool;
17325:  Func ValidateDate( aDERStr : DERString; var RsltFormat :Str) :Bool;
17326:  Func DERStrToDate( aDERStr, aFormat :Str) : TDate;
17327:  Func IsDERChar( aChar : Char) :Bool;
17328:  Func IsDERDefaultChar( aChar : Char) :Bool;
17329:  Func IsDERMoneyChar( aChar : Char) :Bool;
17330:  Func IsDERExceptionCar( aChar : Char) :Bool;
17331:  Func IsDERSymbols( aDERString :Str) :Bool;
17332:  Func StringToDERCharSet( aStr :Str) : DERString;
17333:  Func IsDERNDefaultChar( aChar : Char) :Bool;
17334:  Func IsDERNChar( aChar : Char) :Bool;
17335:  Func DERToDERNCharset( aDERStr : DERString) : DERNString;
17336:  Func DERExtractwebSite( aDERStr : DERString; SmartRecognition :Bool) :Str;
17337:  Func DERExtractWebMail( aDERStr : DERString) :Str;
```

```
17338:  Func DERExtractPhoneNr( aDERStr : DERString) :Str;
17339:  Func DERExecute(aDERStr:DERString;SmartNumbersRecognition,SmartWebsiteRecognition:Bool):TElementsType;
17340:  Func DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInt, RsltFloat, RsltPercentage, RsltwebSite,
        RsltWebMail,RsltMoney,RsltDate:str;SmartNumbersRecognition,SmartWebsiteRecognition:Bool):TElementsType;
17341:  Func RetrieveElementValue(aStr:str;SmartNumbersRecognition,
        SmartWebsiteRecognition:Bool;aElementsType:TElementsType):str;;
17342:  Proc RetrieveElementValue1(aStr:Str;SmartNumbersRecognition,SmartWebsiteRecognition:Bool;var RsltDERStr:
        DERString;var RsltElementType:TElementsType);;
17343: end;
17344:
17345: Proc SIRegister_cyImage(CL: TPSPascalCompiler);
17346: begin
17347:   pRGBQuadArray', '^TRGBQuadArray // will not work;
17348:  Func BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Int);
17349:  Proc BitmapSetPercentBrightness(Bmp: TBitmap; IncPercent:Int; RefreshBmp :Bool);
17350:  Proc BitmapSetPixelsBrightness(Bmp:TBitmap; IncPixels : Int; RefreshBmp:Bool);
17351:  Proc BitmapSetPercentContrast(Bmp:TBitmap; IncPercent : Int; RefreshBmp:Bool);
17352:  Proc BitmapSetPixelsContrast(Bmp: TBitmap; IncPixels : Int; RefreshBmp:Bool);
17353:  Proc BitmapNegative( Bmp : TBitmap; RefreshBmp :Bool);
17354:  Proc BitmapModifyRGB(Bmp:TBitmap;IncRed:Int;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
17355:  Proc BitmapReplaceColor(Bmp:TBitmap;OldColor:TColor;NewColor:TColor;RangeRed,RangeGreen,
        RangeBlue:Word;SingleDestinationColor:Bool;RefreshBmp:Bool);
17356:  Proc BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,PercentRange1Green,
         PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
        PercentRange2Blue:Double;SingleDestinationColor:Bool;RefreshBmp:Bool;;
17357:  Proc BitmapReplaceColors(Bmp:TBitmap; Array_OldPalette,Array_NewPalette:array of
        TColor;SingleDestinationColor,RefreshBmp:Bool);
17358: Proc BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp:Bool
17359:  Proc BitmapRotate(Bmp:TBitmap;AngleDegree: Word;AdjustSize:Bool;BkColor: TColor);
17360:  Proc BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Int;RefreshBmp:Bool;
17361:  Proc GraphicMirror(Source:TGraphic;Dest:TCanvas;Left,Top:Int;Horizontal,Vertical:Bool;
17362:  Proc GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Bool;
17363:  Func BitmapCreate(BmpWidth:Int;BmpHeight:Int;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
17364:  Proc BitmapSaveToJpegFile(Bmp : TBitmap; FileName :Str; QualityPercent : Word);
17365:  Proc JpegSaveToBitmapFile(JPEG : TJPEGImage; FileName :Str);
17366: end;
17367:
17368: Proc SIRegister_WaveUtils(CL: TPSPascalCompiler);
17369: begin
17370:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA );
17371:   TPCMChannel', '( cMono, cStereo );
17372:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz );
17373:   TPCMBitsPerSample', '( bs8Bit, bs16Bit );
17374:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
17375:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
17376:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
17377:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
17378:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
17379:    +'it48000Hz, Stereo16bit48000Hz );
17380:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
17381:    +nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end;
17382:    tWaveFormatEx', 'PWaveFormatEx;  HMMIO', 'Int;
17383:   TWaveDeviceFormats', 'set of TPCMFormat;
17384:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
17385:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound );
17386:   TypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport;
17387:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate );
17388:   TWaveOutOptions', 'set of TWaveOutOption;
17389:   TStreamOwnership2', '( soReference, soOwned );
17390:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx );
17391:  // PRawWave', '^TRawWave // will not work;
17392:   TRawWave', 'record pData : TObject; dwSize : DWORD; end;
17393:   ClassN(CL.FindClass('TOBJECT'),'EWaveAudioError;
17394:   ClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError;
17395:   ClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation;
17396:   TWaveAudioEvent', 'Proc (Sender : TObject);
17397:   TWaveAudioGetFormatEvent', 'Proc (Sender : TObject; var pW'
17398:    +aveFormat : PWaveFormatEx; var FreeIt : Bool);
17399:   TWaveAudioGetDataEvent', 'Func (Sender : TObject; const Buf'
17400:    +fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD;
17401:   TWaveAudioGetDataPtrEvent', 'Func ( Sender : TObject; var Bu'
17402:    +ffer : TObject; var NumLoops : DWORD; var FreeIt :Bool) : DWORD;
17403:   TWaveAudioDataReadyEvent', 'Proc ( Sender : TObject; const '
17404:    +Buffer : TObject; BufferSize : DWORD; var FreeIt :Bool);
17405:   TWaveAudioLevelEvent', 'Proc ( Sender : TObject; Level : Int);
17406:   TWaveAudioFilterEvent,Procedure(Sender:TObject;const Buffer:TObject; BufferSize:DWORD);
17407:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
17408:  CreateWaveAudio(mmIO:HMMIO;const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
17409:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo);
17410: GetStreamWaveAudioInfo(Stream:TStream;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
17411:  CreateStreamWaveAudio(Stream:TStream;const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):HMMIO;
17412:  OpenStreamWaveAudio( Stream : TStream) : HMMIO;
17413:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD;
17414:  GetAudioFormat( FormatTag : Word) :Str;
17415:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) :Str;
17416:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD;
17417:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD;
17418:  GetWaveAudioPeakLevel(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx): Int;
17419:  InvertWaveAudio(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx):Bool;
```

```
17420:  SilenceWaveAudio(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx):Bool;
17421: ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
17422:  MixWaveAudio(const RawWaves:TRawWave;Count:Int;const
       pWaveFormat:PWaveFormatEx;Buffer:TObject;BufferSize:DWORD):Bool;
17423:  ConvertWaveFormat(const srcFormat:PWaveFormatEx;srcData:TObject;srcDataSize:DWORD;const
       dstFormat:PWaveFormatEx; var dstData:TObject;var dstDataSize:DWORD):Bool;
17424:  SetPCMAudioFormat(const pWaveFormat:PWaveFormatEx;Channels:TPCMChannel;SamplesPerSec:TPCMSamplesPerSec;
       BitsPerSample:TPCMBitsPerSample);
17425:  Proc SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat:TPCMFormat);
17426:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat;
17427:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position:DWORD):DWORD;
17428:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) :Str;
17429:  WaitForSyncObject(SyncObject : THandle; Timeout : DWORD) : DWORD;
17430:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT;
17431: end;
17432:
17433: Proc SIRegister_NamedPipes(CL: TPSPascalCompiler);
17434: begin
17435:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
17436:  ConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
17437:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s;
17438:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
17439:  WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
17440:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
17441:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
17442:   TypeS('TPipeDirection', ( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient );
17443:   TypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg );
17444:   TypeS('TOverlappedResult', '(ov_Failed, ov_Pending, ov_MoreData, ov_Complete );
17445:   SIRegister_TNamedPipe(CL);
17446:   SIRegister_TServerPipe(CL);
17447:   SIRegister_TClientPipe(CL);
17448:  Func CalculateTimeout( aBasis : DWORD) : DWORD;
17449:  Func HasOverlappedIoCompleted( const ov : OVERLAPPED) :Bool;
17450:  Func GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var
       dwBytes:DWORD;bWait:Bool):TOverlappedResult;
17451:  Func GetStreamAsText( stm : TStream) :Str;
17452:  Proc SetStreamAsText(const aTxt:Str; stm : TStream);
17453: end;
17454:
17455: Proc SIRegister_DPUtils(CL: TPSPascalCompiler);
17456: begin
17457:  // TypeS('pRGBTripleArray', '^TRGBTripleArray // will not work;
17458:   SIRegister_TThumbData(CL);
17459:  'PIC_BMP','LongInt').SetInt( 0);
17460:  'PIC_JPG','LongInt').SetInt( 1);
17461:  'THUMB_WIDTH','LongInt').SetInt( 60);
17462:  'THUMB_HEIGHT','LongInt').SetInt( 60);
17463:  Func IsEqualFile(Filename:str; Size:Int; LastWriteTime : TDateTime):Bool;
17464:  Proc GetFileInfo(Filename:str; var Size : Int; var LastWriteTime : TDateTime);
17465:  Func ReadBitmap( Filehandle : Int; Width, Height : Int) : TBitmap;
17466:  Proc WriteBitmap( Filehandle : Int; bmp : TBitmap);
17467:  Func OpenPicture( fn :Str; var tp : Int) : Int;
17468:  Func ConvertPicture( pi : Int; tp : Int) : TBitmap;
17469:  Func LoadPicture( fn :Str; var w, h : Int) : TBitmap;
17470:  Func TurnBitmap( bmp : TBitmap; ang : Int) : TBitmap;
17471:  Func RotateBitmap( Bitmap : TBitmap; Direction : Int) : TBitmap;
17472:  Func StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect;
17473:  Func ThumbBitmap( Bitmap : TBitmap; Width, Height : Int) : TBitmap;
17474:  Proc ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Int);
17475:  Proc FindFiles( path, mask :Str; items : TStringList);
17476:  Func LetFileName( s :Str) :Str;
17477:  Func LetParentPath( path :Str) :Str;
17478:  Func AddBackSlash( path :Str) :Str;
17479:  Func CutBackSlash( path :Str) :Str;
17480: end;
17481:
17482: Proc SIRegister_CommonTools(CL: TPSPascalCompiler);
17483: begin
17484:  //'BYTES','LongInt').SetInt( 1);
17485:  'KBYTES','LongInt').SetInt( 1024);
17486:  'DBG_ALIVE','LongWord').SetUInt( Int ( $11BABE11 ));
17487:  'DBG_DESTROYING','LongWord').SetUInt( Int ( $44FADE44 ));
17488:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
17489:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D};
17490:   SIRegister_MakeComServerMethodsPublic(CL);
17491:   TypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application );
17492:  Func IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) :Bool;
17493:  Proc TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet :Bool);
17494:  Func TBGetTempFolder :Str;
17495:  Func TBGetTempFile :Str;
17496:  Func TBGetModuleFilename :Str;
17497:  Func FormatModuleVersionInfo( const aFilename :Str) :Str;
17498:  Func GetVersionInfoString( const aFile, aEntry :Str; aLang : WORD) :Str;
17499:  Func TBGetFileSize( aFile :Str; aMultipleOf : Int) : Int;
17500:  Func FormatAttribString( aAttr : Int) :Str;
17501:  Func GetSomeFileInfo( aFile :Str; aWhatInfo : TSomeFileInfo) :Str;
17502:  Func ShellRecycle( aWnd : HWND; aFileOrFolder :Str) :Bool;
17503:  Func IsDebuggerPresent : BOOL;
17504:  Func TBNotImplemented : HRESULT;
```

```
17505: end;
17506:
17507: Proc SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
17508: begin
17509:   //TypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA;
17510:   //TypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA;
17511:   TypeS('TOSVersionInfoEx', 'TOSVersionInfo;
17512:   TypeS('TDrivesProperty', 'array[1..26] of boolean;;
17513:   //TDrivesProperty = array['A'..'Z'] of boolean;
17514:  Func TBSetSystemTime( DateTime : TDateTime; DOW : word) :Bool;
17515:  Func IsElevated :Bool;
17516: Proc CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
17517:   TypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined );
17518:  Func TrimNetResource( UNC :Str) :Str;
17519:  Proc GetFreeDrives( var FreeDrives : TDrivesProperty);
17520:  Proc GetMappedDrives( var MappedDrives : TDrivesProperty);
17521:  Func MapDrive(UNCPath:str;Drive:char;PasswordUsage:TPasswordUsage;Password:str;
       UserUsage:TPasswordUsage;User:str;Comment:str):bool;
17522:  Func UnmapDrive( Drive : char; Force :Bool):Bool;
17523:  Func TBIsWindowsVista :Bool;
17524:  Proc SetVistaFonts( const AForm : TForm);
17525:  Proc SetVistaContentFonts( const AFont : TFont);
17526:  Func GetProductType( var sType :Str) :Bool;
17527:  Func lstrcmp( lpString1, lpString2 : PChar) : Int;
17528:  Func lstrcmpi( lpString1, lpString2 : PChar) : Int;
17529:  Func lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Int) : PChar;
17530:  Func lstrcpy( lpString1, lpString2 : PChar) : PChar;
17531:  Func lstrcat( lpString1, lpString2 : PChar) : PChar;
17532:  Func lstrlen( lpString : PChar) : Int;
17533:  Func
       GetTokenInformation(TokenHandle:THandle;TokenInformationClass:TTokenInformationClass;TokenInformation:___Pointer;Token
       ReturnLength:DWORD):BOOL;
17534:  Func
       SetTokenInformation(TokenHandle:THandle;TokenInformationClass:TTokenInformationClass;TokenInformation:___Pointer;Token
17535: end;
17536:
17537: Proc SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
17538: begin
17539:  'cLineTerminator','Char').SetString( #10);
17540:  'cLineTerminators','String').SetString( #13#10);
17541:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
17542:   SIRegister_TFixedCriticalSection(CL);
17543:   SIRegister_TMultiReadSingleWrite(CL);
17544:  Proc SetDecimalSeparator( c : Char);
17545:  Func GetDecimalSeparator : Char;
17546:  Func GetDecimalSeparator : Char;
17547:  Proc SetDateSeparator( c : Char);
17548:  Func GetDateSeparator : Char;
17549:  Proc setTimeSeparator( c : Char);
17550:  Func getTimeSeparator : Char;
17551:  Proc setShortDateFormat( c :Str);
17552:  Func getShortDateFormat :Str;
17553:  Proc setlongDateFormat( c :Str);
17554:  Func getlongDateFormat :Str;
17555:  Proc setShorttimeFormat( c :Str);
17556:  Func getShorttimeFormat :Str;
17557:  Proc setlongtimeFormat( c :Str);
17558:  Func getlongtimeFormat :Str;
17559:   //getTimeSeparator
17560:  Proc setThousandSeparator(c : Char);
17561:  Func getThousandSeparator : Char;
17562:  Proc setListSeparator( c : Char);
17563:  Func getListSeparator : Char;
17564:  'LOGON32_LOGON_INTERACTIVE','LongInt').SetInt( 2);
17565:  'LOGON32_LOGON_NETWORK','LongInt').SetInt( 3);
17566:  'LOGON32_LOGON_BATCH','LongInt').SetInt( 4);
17567:  'LOGON32_LOGON_SERVICE','LongInt').SetInt( 5);
17568:  'LOGON32_PROVIDER_DEFAULT','LongInt').SetInt( 0);
17569:  'LOGON32_PROVIDER_WINNT35','LongInt').SetInt( 1);
17570:  'LOGON32_PROVIDER_WINNT40','LongInt').SetInt( 2);
17571:  'LOGON32_PROVIDER_WINNT50','LongInt').SetInt( 3);
17572:
17573:  TCollectFileProgressEvent', 'Proc ( const directory:Str; var skipScan:Bool);
17574:  Proc CollectFiles(const directory,fileMask:UnicodeString; list:TStrings; recurseSubdirectories:Boolean;
       onProgress : TCollectFileProgressEvent);
17575:   TypeS('NativeInt', 'Int;
17576:   //TypeS('PNativeInt', '^NativeInt // will not work;
17577:   TypeS('NativeUInt', 'Cardinal;
17578:   //TypeS('PNativeUInt', '^NativeUInt // will not work;
17579:   //TypeS('TBytes', 'array of Byte;
17580:   TypeS('RawByteString', 'UnicodeString;
17581:   //TypeS('PNativeInt', '^NativeInt // will not work;
17582:   //TypeS('PUInt64', '^UInt64 // will not work;
17583:   SIRegister_TPath(CL);
17584:   SIRegister_TFile(CL);
17585:   SIRegister_TdwsThread(CL);
17586:  Func GetSystemMilliseconds : Int64;
17587:  Func UTCDateTime : TDateTime;
17588:  Func UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString;
```

```
17589:  Func UnicodeCompareStr( const S1, S2 : UnicodeString) : Int;
17590:  Func dwsAnsiCompareText( const S1, S2 : UnicodeString) : Int;
17591:  Func dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Int;
17592:  Func UnicodeComparePChars(p1:PChar; n1 : Int; p2 : PChar; n2 : Int): Int;;
17593:  Func UnicodeComparePChars1( p1, p2 : PChar; n : Int) : Int;;
17594:  Func UnicodeLowerCase( const s : UnicodeString) : UnicodeString;
17595:  Func UnicodeUpperCase( const s : UnicodeString) : UnicodeString;
17596:  Func ASCIICompareText( const s1, s2 : UnicodeString) : Int;
17597:  Func ASCIISameText( const s1, s2 : UnicodeString) :Bool;
17598:  Func InterlockedIncrement( var val : Int) : Int;
17599:  Func InterlockedDecrement( var val : Int) : Int;
17600:  Proc FastInterlockedIncrement( var val : Int);
17601:  Proc FastInterlockedDecrement( var val : Int);
17602:  Func InterlockedExchangePointer(var target:___Pointer; val:___Pointer):___Pointer;
17603:  Proc SetThreadName( const threadName : Char; threadID :Card);
17604:  Proc dwsOutputDebugString( const msg : UnicodeString);
17605: Proc WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeStr;const logRawData:Str)
17606: Func TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
17607:  Proc VarCopy( out dest : Variant; const src : Variant);
17608:  Func VarToUnicodeStr( const v : Variant) : UnicodeString;
17609:  Func LoadTextFromBuffer( const buf : TBytes) : UnicodeString;
17610:  Func LoadTextFromStream( aStream : TStream) : UnicodeString;
17611:  Func LoadTextFromFile( const fileName : UnicodeString) : UnicodeString;
17612:  Proc SaveTextToUTF8File( const fileName, text : UnicodeString);
17613:  Func OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle;
17614:  Func OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle;
17615:  Proc CloseFileHandle( hFile : THandle);
17616:  Func FileCopy(const existing, new : UnicodeString; failIfExists :Bool):Boolean;
17617:  Func FileMove( const existing, new : UnicodeString) :Bool;
17618:  Func dwsFileDelete( const fileName :Str) :Bool;
17619:  Func FileRename( const oldName, newName :Str) :Bool;
17620:  Func dwsFileSize( const name :Str) : Int64;
17621:  Func dwsFileDateTime( const name :Str) : TDateTime;
17622:  Func DirectSet8087CW( newValue : Word) : Word;
17623:  Func DirectSetMXCSR( newValue : Word) : Word;
17624:  Func TtoObject( const T: byte) : TObject;
17625:  Func TtoPointer( const T: byte) : ___Pointer;
17626:  Proc GetMemForT(var T: byte; Size : Int);
17627:  Func FindDelimiter( const Delimiters, S :Str; StartIdx : Int) : Int;
17628: end;
17629:
17630: Proc SIRegister_AdSocket(CL: TPSPascalCompiler);
17631: begin
17632:  'IPStrSize','LongInt').SetInt( 15);
17633:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17634:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
17635:  'ADWSBASE','LongInt').SetInt( 9000);
17636:   TypeS('TCMAPDSocketMessage', 'record Msg :Card; Socket : TSocket; '
17637:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end;
17638:   SIRegister_EApdSocketException(CL);
17639:   TWsMode', '( wsClient, wsServer );
17640:   TWsNotifyEvent', 'Proc ( Sender : TObject; Socket : TSocket;
17641:   TWsSocketErrorEvent', 'Proc ( Sender : TObject; Socket : TSocket;ErrCode : Int);
17642:   SIRegister_TApdSocket(CL);
17643: end;
17644:
17645: Proc SIRegister_AdPort(CL: TPSPascalCompiler);
17646: begin
17647:   SIRegister_TApdCustomComPort(CL);
17648:   SIRegister_TApdComPort(CL);
17649:  Func ComName( const ComNumber : Word) : ShortString;
17650:  Func SearchComPort( const C : TComponent) : TApdCustomComPort;
17651: end;
17652:
17653: Proc SIRegister_PathFunc(CL: TPSPascalCompiler);
17654: begin
17655:  Func inAddBackslash( const S :Str) :Str;
17656:  Func PathChangeExt( const Filename, Extension :Str) :Str;
17657:  Func PathCharCompare( const S1, S2 : PChar) :Bool;
17658:  Func PathCharIsSlash( const C : Char) :Bool;
17659:  Func PathCharIsTrailByte( const S :Str; const Index : Int) :Bool;
17660:  Func PathCharLength( const S :Str; const Index : Int) : Int;
17661:  Func inPathCombine( const Dir, Filename :Str) :Str;
17662:  Func PathCompare( const S1, S2 :Str) : Int;
17663:  Func PathDrivePartLength( const Filename :Str) : Int;
17664:  Func PathDrivePartLengthEx(const Filename:str;const IncludeSignificantSlash:Bool):Int;
17665:  Func inPathExpand( const Filename :Str) :Str;
17666:  Func PathExtensionPos( const Filename :Str) : Int;
17667:  Func PathExtractDir( const Filename :Str) :Str;
17668:  Func PathExtractDrive( const Filename :Str) :Str;
17669:  Func PathExtractExt( const Filename :Str) :Str;
17670:  Func PathExtractName( const Filename :Str) :Str;
17671:  Func PathExtractPath( const Filename :Str) :Str;
17672:  Func PathIsRooted( const Filename :Str) :Bool;
17673:  Func PathLastChar( const S :Str) : PChar;
17674:  Func PathLastDelimiter( const Delimiters, S :Str) : Int;
17675:  Func PathLowercase( const S :Str) :Str;
17676:  Func PathNormalizeSlashes( const S :Str) :Str;
17677:  Func PathPathPartLength(const Filename:str;const IncludeSlashsAfterPath:Bool):Int;
```

```
17678:  Func PathPos( Ch : Char; const S :Str) : Int;
17679:  Func PathStartsWith( const S, AStartsWith :Str) :Bool;
17680:  Func PathStrNextChar( const S : PChar) : PChar;
17681:  Func PathStrPrevChar( const Start, Current : PChar) : PChar;
17682:  Func PathStrScan( const S : PChar; const C : Char) : PChar;
17683:  Func inRemoveBackslash( const S :Str) :Str;
17684:  Func RemoveBackslashUnlessRoot( const S :Str) :Str;
17685:  Proc PathFuncRunTests( const AlsoTestJapaneseDBCS :Bool);
17686: end;
17687:
17688: Proc SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17689: begin
17690:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion;
17691:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer;
17692:  NEWREGSTR_PATH_SPECIAL_FOLDERS,String').SetStr(NEWREGSTR_PATH_EXPLORER +'\Shell Folders;
17693:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall;
17694:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName;
17695:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString;
17696:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17697:   //TypeS('PLeadByteSet', '^TLeadByteSet // will not work;
17698:   TypeS('TLeadByteSet', 'set of Char;
17699:   SIRegister_TOneShotTimer(CL);
17700:   TypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit );
17701:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17702:  Func NewFileExists( const Name :Str) :Bool;
17703:  Func inDirExists( const Name :Str) :Bool;
17704:  Func FileOrDirExists( const Name :Str) :Bool;
17705:  Func IsDirectoryAndNotReparsePoint( const Name :Str) :Bool;
17706:  Func GetIniString(const Section,Key:str; Default:str;const Filename:str):str;
17707:  Func GetIniInt(const Section,Key:Str;const Default,Min,Max:Longint;const Filenam:Str):Longint;
17708:  Func GetIniBool(const Section,Key:str;const Default:Bool;const Filename:str):Bool;
17709:  Func IniKeyExists( const Section, Key, Filename :Str) :Bool;
17710:  Func IsIniSectionEmpty( const Section, Filename :Str) :Bool;
17711:  Func SetIniString( const Section, Key, Value, Filename :Str) :Bool;
17712:  Func SetIniInt(const Section,Key:str;const Value:Longint;const Filename:str):Bool;
17713:  Func SetIniBool(const Section,Key:str;const Value:Bool;const Filename:Str):Bool;
17714:  Proc DeleteIniEntry( const Section, Key, Filename :Str);
17715:  Proc DeleteIniSection( const Section, Filename :Str);
17716:  Func GetEnv( const EnvVar :Str) :Str;
17717:  Func GetCmdTail :Str;
17718:  Func GetCmdTailEx( StartIndex : Int) :Str;
17719:  Func NewParamCount : Int;
17720:  Func NewParamStr( Index : Int) :Str;
17721:  Func AddQuotes( const S :Str) :Str;
17722:  Func RemoveQuotes( const S :Str) :Str;
17723:  Func inGetShortName( const LongName :Str) :Str;
17724:  Func inGetWinDir :Str;
17725:  Func inGetSystemDir :Str;
17726:  Func GetSysWow64Dir :Str;
17727:  Func GetSysNativeDir( const IsWin64 :Bool) :Str;
17728:  Func inGetTempDir :Str;
17729:  Func StringChange( var S :Str; const FromStr, ToStr :Str) : Int;
17730:  Func StringChangeEx(var S:str;const FromStr,ToStr:str;const SupportDBCS:Bool):Int;
17731:  Func AdjustLength( var S :Str; const Res :Card) :Bool;
17732:  Func UsingWinNT :Bool;
17733:  Func ConvertConstPercentStr( var S :Str) :Bool;
17734:  Func ConvertPercentStr( var S :Str) :Bool;
17735:  Func ConstPos( const Ch : Char; const S :Str) : Int;
17736:  Func SkipPastConst( const S :Str; const Start : Int) : Int;
17737:  Func RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr :Str) :Bool;
17738:  Func RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr:str):Bool;
17739:  Func RegValueExists( H : HKEY; Name : PChar) :Bool;
17740:  Func RegCreateKeyExView(const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;Reserved:DWORD;lpClass:PChar;
        dwOptions:DWORD;samDesired:REGSAM;lpSecurityAttributes:TObject;var
        phkResult:HKEY;lpdwDisposition:DWORD):Longint;
17741:  Func RegOpenKeyExView(const
        RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17742:  Func RegDeleteKeyView( const RegView:TRegView;const Key:HKEY;const Name:PChar): Longint;
17743:  Func RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17744:  Func RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyNam:PChar):Longint;
17745:  Func GetShellFolderPath( const FolderID : Int) :Str;
17746:  Func IsAdminLoggedOn :Bool;
17747:  Func IsPowerUserLoggedOn :Bool;
17748:  Func IsMultiByteString( const S : Ansistr) :Bool;
17749:  Func FontExists( const FaceName :Str) :Bool;
17750:  //Proc FreeAndNil( var Obj);
17751:  Func SafeLoadLibrary( const Filename :Str; ErrorMode : UINT) : HMODULE;
17752:  Func GetUILanguage : LANGID;
17753:  Func RemoveAccelChar( const S :Str) :Str;
17754:  Func GetTextWidth(const DC : HDC; S :Str; const Prefix:Boolean):Int;
17755:  Func AddPeriod( const S :Str) :Str;
17756:  Func GetExceptMessage :Str;
17757:  Func GetPreferredUIFont :Str;
17758:  Func IsWildcard( const Pattern :Str) :Bool;
17759:  Func WildcardMatch( const Text, Pattern : PChar) :Bool;
17760:  Func IntMax( const A, B : Int) : Int;
17761:  Func Win32ErrorString( ErrorCode : Int) :Str;
17762:  Proc GetLeadBytes( var ALeadBytes : TLeadByteSet);
17763:  Func inCompareMem( P1, P2 : TObject; Length : Int) :Bool;
```

```
17764:  Func DeleteDirTree( const Dir :Str) :Bool;
17765:  Func SetNTFSCompression(const FileOrDir:Str; Compress :Bool) :Bool;
17766:  Proc AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT);
17767:  // TypeS('TSysCharSet', 'set of AnsiChar;
17768:  Func inCharInSet( C : Char; const CharSet : TSysCharSet) :Bool;
17769:  Func ShutdownBlockReasonCreate( Wnd : HWND; const Reason :Str) :Bool;
17770:  Func ShutdownBlockReasonDestroy( Wnd : HWND) :Bool;
17771:  Func TryStrToBoolean( const S :Str; var BoolResult :Bool) :Bool;
17772:  Proc WaitMessageWithTimeout( const Milliseconds : DWORD);
17773:  Func MoveFileReplace(const ExistingFileName, NewFileName :Str) :Bool;
17774:  Proc TryEnableAutoCompleteFileSystem( Wnd : HWND);
17775: end;
17776:
17777: Proc SIRegister_CmnFunc(CL: TPSPascalCompiler);
17778: begin
17779:   SIRegister_TWindowDisabler(CL);
17780:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError );
17781:   TMsgBoxCallbackFunc,procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17782:  Proc UpdateHorizontalExtent( const ListBox : TCustomListBox);
17783:  Func MinimizePathName(const Filename:str; const Font:TFont;MaxLen:Int):str;
17784:  Func AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Int;
17785:  Func MsgBoxP(const Text,Caption:PChar;const Typ: TMsgBoxType;const Buttons:Card):Int;
17786:  Func inMsgBox(const Text,Caption:str;const Typ:TMsgBoxType;const Buttons:Card):Int;
17787:  Func MsgBoxFmt(const Text:str;const Args:array of const;const Caption:str;const Typ:TMsgBoxType;const
        Buttons:Card):Int;
17788:  Proc ReactivateTopWindow;
17789:  Proc SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar);
17790:  Proc SetMessageBoxRightToLeft( const ARightToLeft :Bool);
17791:  Proc SetMessageBoxCallbakFunc(const AFunc:TMsgBoxCallbackFunc;const AParam:LongInt);
17792: end;
17793:
17794: Proc SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17795: begin
17796:   SIRegister_TImageGrabber(CL);
17797:   SIRegister_TCaptureDrivers(CL);
17798:   SIRegister_TCaptureDriver(CL);
17799: end;
17800:
17801: Proc SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17802: begin
17803:  Func GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:str;const
        Entries:TGrantPermissionEntry; const EntryCount:Int):Boolean;
17804:  Func GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:str;const Entries:
        TGrantPermissionEntry; const EntryCount:Int):Boolean;
17805: end;
17806:
17807: Proc SIRegister_RedirFunc(CL: TPSPascalCompiler);
17808: begin
17809:   TypeS(TPreviousFsRedirectionState,record DidDisable:Boolean;OldValue: ___Pointer;end;
17810:  Func AreFsRedirectionFunctionsAvailable :Bool;
17811:  Func DisableFsRedirectionIf(const Disable:Bool;var PreviousState:TPreviousFsRedirectionState):Boo;
17812:  Proc RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState);
17813:  Func CreateDirectoryRedir(const DisableFsRedir:Boolean; const Filename:Str) : BOOL;
17814:  Func 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;
17815:  Func CopyFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:str;const
        FailIfExists:BOOL):BOOL;
17816:  Func DeleteFileRedir(const DisableFsRedir:Boolean; const Filename:Str): BOOL;
17817:  Func DirExistsRedir(const DisableFsRedir:Boolean; const Filename:Str):Bool;
17818:  Func FileOrDirExistsRedir( const DisableFsRedir:Bool; const Filename:str):Bool;
17819:  Func FindFirstFileRedir(const DisableFsRedir:Boolean;const Filename:str;var
        FindData:TWin32FindData):THandle;
17820:  Func GetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:str): DWORD;
17821:  Func GetShortNameRedir(const DisableFsRedir:Boolean; const Filename:str):str;
17822:  Func GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:str;var
        VersionNumbers:TFileVersionNumbers): Bool;
17823: Func IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Bool;const Name:str):Bool;
17824: Func MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilenme,NewFilename:str):BOOL;
17825:  Func MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilen,NewFilename:str;const
        Flags:DWORD):BOOL;
17826:  Func NewFileExistsRedir(const DisableFsRedir:Boolean;const Filename:str):Bool;
17827:  Func RemoveDirectoryRedir(const DisableFsRedir:Boolean;const Filename :Str) : BOOL;
17828:  Func SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:str;const Attrib:DWORD):BOOL;
17829: Func SetNTFSCompressionRedir(const DisableFsRedir:Bool;const FileOrDir:Str;Compress:Bool:Bool;
17830:   SIRegister_TFileRedir(CL);
17831:   SIRegister_TTextFileReaderRedir(CL);
17832:   SIRegister_TTextFileWriterRedir(CL);
17833: end;
17834:
17835: Proc SIRegister_Int64Em(CL: TPSPascalCompiler);
17836: begin
17837:   //TypeS('LongWord', 'Cardinal;
17838:   TypeS('Int64', 'record Lo : LongWord; Hi : LongWord; end;
17839:  Func Compare64( const N1, N2 : Int64) : Int;
17840:  Proc Dec64( var X : Int64; N : LongWord);
17841:  Proc Dec6464( var X : Int64; const N : Int64);
```

```
17842:  Func Div64( var X : Int64; const Divisor : LongWord) : LongWord;
17843:  Func Inc64( var X : Int64; N : LongWord) :Bool;
17844:  Func Inc6464( var X : Int64; const N : Int64) :Bool;
17845:  Func Int64ToStr( X : Int64) :Str;
17846:  Func Mod64( const X : Int64; const Divisor : LongWord) : LongWord;
17847:  Func Mul64( var X : Int64; N : LongWord) :Bool;
17848:  Proc Multiply32x32to64( N1, N2 : LongWord; var X : Int64);
17849:  Proc Shr64( var X : Int64; Count : LongWord);
17850:  Func StrToInt64( const S :Str; var X : Int64) :Bool;
17851: end;
17852:
17853: Proc SIRegister_InstFunc(CL: TPSPascalCompiler);
17854: begin
17855:   //TypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work;
17856:   SIRegister_TSimpleStringList(CL);
17857:   TypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle);
17858:   TypeS('TDetermineDefaultLanguageResult','(ddNoMatch,ddMatch,ddMatchLangParameter );
17859:   TypeS('TMD5Digest', 'array[0..15] of Byte;;
17860:   TypeS('TSHA1Digest', 'array[0..19] of Byte;;
17861:  // TMD5Digest = array[0..15] of Byte; TSHA1Digest = array[0..19] of Byte;
17862:  Func CheckForMutexes( Mutexes :Str) :Bool;
17863:  Func CreateTempDir :Str;
17864:  Func DecrementSharedCount(const RegView:TRegView;const Filename:str):Bool;
17865:  Proc DelayDeleteFile( const DisableFsRedir :Bool; const Filename :Str; const MaxTries, FirstRetryDelayMS,
       SubsequentRetryDelayMS : Int);
17866:  //Func DelTree( const DisableFsRedir :Bool; const Path :Str; const IsDir, DeleteFiles, DeleteSubdirsAlso,
       BreakOnError :Bool; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc : TDeleteFileProc; const
       Param : Pointer) :Bool;
17867:  //Func DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
       TSetupLanguageDetectionMethod;const LangParameter:str;var ResultIndex:Int):
       TDetermineDefaultLanguageResult;
17868:  //Proc EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17869:  Func GenerateNonRandomUniqueFilename(Path:Str; var Filename:Str):Bool;
17870:  Func GenerateUniqueName(const DisableFsRedir:Bool;Path:Str;const Extension:str):str;
17871:  Func GetComputerNameString :Str;
17872: Func GetFileDateTime(const DisableFsRedir:Bool;const Filename:Str;var DateTime:TFileTime):Bool;
17873:  Func GetMD5OfFile(const DisableFsRedir:Boolean; const Filename :Str): TMD5Digest;
17874:  Func GetMD5OfAnsistr( const S : Ansistr) : TMD5Digest;
17875: // Func GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest;
17876:  Func GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:str):TSHA1Digest;
17877:  Func GetSHA1OfAnsistr( const S : Ansistr) : TSHA1Digest;
17878: // Func GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest;
17879:  Func GetRegRootKeyName( const RootKey : HKEY) :Str;
17880:  Func GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:str;var FreeBytes,TotBytes:Int64):Bool;
17881:  Func GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:str;var FreeBytes,
       TotalBytes:Int64):Bool;
17882:  Func GetUserNameString :Str;
17883:  Proc IncrementSharedCount(const RegView:TRegView;const Filename:str;const AlreadyExisted:Bool;
17884:  //Func InstExec( const DisableFsRedir :Bool; const Filename, Params :Str; WorkingDir :Str; const
       Wait:TExecWait;const ShowCmd:Int;const ProcessMessagesProc:TProcedure;var ResultCode:Int) :Bool;
17885: // Func InstShellExec( const Verb, Filename, Params :Str; WorkingDir :Str; const Wait:TExecWait;const
       ShowCmd:Int;const ProcessMessagesProc:TProcedure;var ResultCode:Int):Bool;
17886:  Proc InternalError( const Id :Str);
17887:  Proc InternalErrorFmt( const S :Str; const Args : array of const);
17888:  Func IsDirEmpty( const DisableFsRedir :Bool; const Dir :Str) :Bool;
17889:  Func IsProtectedSystemFile(const DisableFsRedir:Boolean;const Filename:str):Bool;
17890:  Func MakePendingFileRenameOperationsChecksum : TMD5Digest;
17891:  Func ModifyPifFile( const Filename :Str; const CloseOnExit :Bool) :Bool;
17892:  Proc RaiseFunctionFailedError( const FunctionName :Str);
17893:  Proc RaiseOleError( const FunctionName :Str; const ResultCode : HRESULT);
17894:  Proc RefreshEnvironment;
17895:  Func ReplaceSystemDirWithSysWow64( const Path :Str) :Str;
17896:  Func ReplaceSystemDirWithSysNative( Path :Str; const IsWin64 :Bool) :Str;
17897:  Proc UnregisterFont( const FontName, FontFilename :Str);
17898:  Func RestartComputer :Bool;
17899:  Proc RestartReplace( const DisableFsRedir :Bool; TempFile, DestFile :Str);
17900:  Proc SplitNewParamStr( const Index : Int; var AName, AValue :Str);
17901:  Proc Win32ErrorMsg( const FunctionName :Str);
17902:  Proc Win32ErrorMsgEx( const FunctionName :Str; const ErrorCode : DWORD);
17903:  Func inForceDirectories(const DisableFsRedir:Boolean;Dir:str):Bool;
17904:  //from Func2
17905:  //Func inCreateShellLink(const Filename,Description,ShortcutTo,Parameters, WorkingDir:Str;
       IconFilename:Str; const IconIndex, ShowCmd : Int; const HotKey : Word; FolderShortcut :Bool; '
17906:   //+'const AppUserModelID:Str;const ExcludeFromShowInNewInstall,PreventPinning:Bool):Str;
17907:  Proc RegisterTypeLibrary( const Filename :Str);
17908:  //Proc UnregisterTypeLibrary( const Filename :Str);
17909:  //Func UnpinShellLink( const Filename :Str) :Bool;}
17910:  Func getVersionInfoEx3: TOSVersionInfoEx;;
17911:  Func GetVersionEx3(out verinfo: TOSVersionInfoEx):Bool;;
17912:  Proc InitOle;;
17913:  Func ExpandConst( const S :Str) :Str;
17914:  Func ExpandConstEx( const S :Str; const CustomConsts : array of String) :Str;
17915:  Func ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndivdConst:Bool):Str;
17916:  Func ExpandConstIfPrefixed( const S :Str) :Str;
17917:  Proc LogWindowsVersion;
17918:  Func EvalCheck( const Expression :Str) :Bool;
17919: end;
17920:
17921: Proc SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
```

```
17922: begin
17923: ClassN(CL.FindClass('TOBJECT'),'TResourceDetails;
17924: //TypeS('TResourceDetailsClass', 'class of TResourceDetails;
17925:  SIRegister_TResourceModule(CL);
17926:  SIRegister_TResourceDetails(CL);
17927:  SIRegister_TAnsiResourceDetails(CL);
17928:  SIRegister_TUnicodeResourceDetails(CL);
17929:  Proc RegisterResourceDetails( resourceClass : TResourceDetailsClass);
17930:  Proc UnRegisterResourceDetails( resourceClass : TResourceDetailsClass);
17931:  Func ResourceWideCharToStr( var wstr : PWideChar; codePage : Int) :Str;
17932:  Proc ResourceStrToWideChar( const s :Str; var p : PWideChar; codePage : Int);
17933:  Func ResourceNameToInt( const s :Str) : Int;
17934:  Func CompareDetails( p1, p2 : TObject(Pointer)) : Int;
17935: end;
17936:
17937:
17938: Proc SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17939: begin
17940:  //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17941:   with ClassN(CL.FindClass('TObject'),TSimpleComPort') do begin
17942:     RegisterMethod(Constructor Create;
17943:     RegisterMethod(Proc Free;
17944:     RegisterMethod(Proc Open( PortNumber : Int; const Parameters:str);
17945:     RegisterMethod(Proc WriteString( const S :Str);
17946:     RegisterMethod(Proc ReadString( var S :Str);
17947:   end;
17948:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17949:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1;
17950:       SimpleComPort.WriteString(AsciiChar);
17951: end;
17952:
17953: Proc SIRegister_Console(CL: TPSPascalCompiler);
17954: begin
17955:  ConstantN('White','LongInt').SetInt( 15); }
17956: // ConstantN('Blink','LongInt').SetInt( 128);
17957:  ('conBW40','LongInt').SetInt( 0);
17958:  ('conCO40','LongInt').SetInt( 1);
17959:  ('conBW80','LongInt').SetInt( 2);
17960:  ('conCO80','LongInt').SetInt( 3);
17961:  ('conMono','LongInt').SetInt( 7);
17962:  ConstantN('conFont8x8','LongInt').SetInt( 256);
17963:  //ConstantN('C40','').SetString( CO40);
17964:  //ConstantN('C80','').SetString( CO80);
17965:  Func conReadKey : Char;
17966:  Func conKeyPressed :Bool;
17967:  Proc conGotoXY( X, Y : Smallint);
17968:  Func conWhereX : Int;
17969:  Func conWhereY : Int;
17970:  Proc conTextColor( Color : Byte);;
17971:  Func conTextColor1 : Byte;;
17972:  Proc conTextBackground( Color : Byte);;
17973:  Func conTextBackground1 : Byte;;
17974:  Proc conTextMode( Mode : Word);
17975:  Proc conLowVideo;
17976:  Proc conHighVideo;
17977:  Proc conNormVideo;
17978:  Proc conClrScr;
17979:  Proc conClrEol;
17980:  Proc conInsLine;
17981:  Proc conDelLine;
17982:  Proc conWindow( Left, Top, Right, Bottom : Int);
17983:  Func conScreenWidth : Smallint;
17984:  Func conScreenHeight : Smallint;
17985:  Func conBufferWidth : Smallint;
17986:  Func conBufferHeight : Smallint;
17987:  Proc InitScreenMode;;
17988: end;
17989:
17990: (*----------------------------------------------------------------------------*)
17991: Proc SIRegister_testutils(CL: TPSPascalCompiler);
17992: begin
17993:   SIRegister_TNoRefCountObject(CL);
17994:  Proc FreeObjects( List : TFPList);
17995:  Proc GetMethodList( AObject : TObject; AList : TStrings);;
17996:  Proc GetMethodList1( AClass : TClass; AList : TStrings);;
17997: end;
17998:
17999: Proc SIRegister_ToolsUnit(CL: TPSPascalCompiler);
18000: begin
18001:  'MaxDataSet','LongInt').SetInt( 35);
18002:   //TypeS('TDBConnectorClass', 'class of TDBConnector;
18003:   SIRegister_TDBConnector(CL);
18004:   SIRegister_TDBBasicsTestSetup(CL);
18005:   SIRegister_TTestDataLink(CL);
18006:  'testValuesCount',LongInt).SetInt( 25);
18007:  Proc InitialiseDBConnector;
18008:  Proc FreeDBConnector;
18009:  Func DateTimeToTimeString( d : tdatetime) :Str;
18010:  Func TimeStringToDateTime( d :Str) : TDateTime;
```

```
18011: end;
18012:
18013: Proc SIRegister_fpcunit(CL: TPSPascalCompiler);
18014: begin
18015:   SIRegister_EAssertionFailedError(CL);
18016:   TypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing );
18017:   TypeS('TRunMethod', 'Procedure;
18018:   ClassN(CL.FindClass('TOBJECT'),'TTestResult;
18019:   SIRegister_TTest(CL);
18020:   SIRegister_TAssert(CL);
18021:   SIRegister_TTestFailure(CL);
18022:   SIRegister_ITestListener(CL);
18023:   SIRegister_TTestCase(CL);
18024:   //TypeS('TTestCaseClass', 'class of TTestCase;
18025:   SIRegister_TTestSuite(CL);
18026:   SIRegister_TTestResult(CL);
18027:  Func ComparisonMsg( const aExpected:Str; const aActual :Str) :Str;
18028: end;
18029:
18030: Proc SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
18031: begin
18032:   TOBJECT'),'ETCPBuffer;
18033:   TTCPBuffer', 'record Ptr : TObject; Size : Int; Max : Intege'
18034:    +r; Head : Int; Used : Int; end;
18035:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
18036:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
18037:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
18038:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
18039: Proc TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSiz:Int;const TCPBufSize:Int;
18040:  Proc TCPBufferFinalise( var TCPBuf : TTCPBuffer);
18041:  Proc TCPBufferPack( var TCPBuf : TTCPBuffer);
18042:  Proc TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Int);
18043:  Proc TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Int);
18044:  Proc TCPBufferShrink( var TCPBuf : TTCPBuffer);
18045:  Func TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Int) : Pointer;
18046:  Proc TCPBufferAddBuf(var TCPBuf: TTCPBuffer; const Buf:str; const Size : Int);
18047:  Func TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Int;
18048:  Func TCPBufferPeek(var TCPBuf:TTCPBuffer;var Buf:Str; const Size:Int): Int;
18049:  Func TCPBufferRemove(var TCPBuf:TTCPBuffer;var Buf:str;const Size:Int): Int;
18050:  Proc TCPBufferClear( var TCPBuf : TTCPBuffer);
18051:  Func TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Int) : Int;
18052:  Func TCPBufferUsed( const TCPBuf : TTCPBuffer) : Int;
18053:  Func TCPBufferEmpty( const TCPBuf : TTCPBuffer) :Bool;
18054:  Func TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Int;
18055:  Func TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer;
18056:  Proc TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Int);
18057: end;
18058:
18059: Proc SIRegister_Glut(CL: TPSPascalCompiler);
18060: begin
18061:   //TypeS('PInt', '^Int // will not work;
18062:   //TypeS('PPChar', '^PChar // will not work;
18063:  ConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
18064:  ConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
18065:  ConstantN('GLUT_RGB','LongInt').SetInt( 0);
18066:  ConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
18067:  ConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
18068:  Proc LoadGlut( const dll :Str);
18069:  Proc FreeGlut;
18070: end;
18071:
18072: Proc SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
18073: begin
18074:   TypeS('TLEDColor',( ledcGreen, ledcRed, ledcGray );
18075:  Func GetLEDBitmapHandle( Color : TLEDColor; State:Boolean): THandle;
18076: end;
18077:
18078: Proc SIRegister_SwitchLed(CL: TPSPascalCompiler);
18079: begin
18080:   TLedColor, ( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black );
18081:   TTLedState, ( LedOn, LedOff, LedDisabled );
18082:   SIRegister_TSwitchLed(CL);
18083:  //Proc Register;
18084: end;
18085:
18086: Proc SIRegister_FileClass(CL: TPSPascalCompiler);
18087: begin
18088:   TypeS(TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
18089:    +'xisting, fdOpenAlways, fdTruncateExisting );
18090:   TypeS('TIFileAccess', '( faRead, faWrite, faReadWrite );
18091:   TypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite );
18092:   SIRegister_TCustomFile(CL);
18093:   SIRegister_TIFile(CL);
18094:   SIRegister_TMemoryFile(CL);
18095:   SIRegister_TTextFileReader(CL);
18096:   SIRegister_TTextFileWriter(CL);
18097:   SIRegister_TFileMapping(CL);
18098:   SIRegister_EFileError(CL);
18099: end;
```

```
18100:
18101: Proc SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
18102: begin
18103:   TypeS(TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
18104:    +, ffaDirectory, ffaArchive, ffaAnyFile );
18105:   TypeS('TAttributeSet', 'set of TFileAttributes;
18106:   SIRegister_TFileSearch(CL);
18107: end;
18108:
18109: Proc SIRegister_uColorFunctions(CL: TPSPascalCompiler);
18110: begin
18111:  TRGBType', 'record RedHex :Str; GreenHex :Str; BlueHex :'
18112:    + string; Red : Int; Green : Int; Blue : Int; end;
18113:  Func FadeColor( aColor : Longint; aFade : Int) : Tcolor;
18114:  Func CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:Int):Tcolor;
18115: end;
18116:
18117: Proc SIRegister_uSettings(CL: TPSPascalCompiler);
18118: begin
18119:  Proc SaveOscSettings;
18120:  Proc GetOscSettings;
18121: end;
18122:
18123: Proc SIRegister_cyDebug(CL: TPSPascalCompiler);
18124: begin
18125:   TProcProcessEvent', 'Proc ( Sender : TObject; Index : Int);
18126:   RecProcess', 'record Name : ShortString; DurationMs :Card; '
18127:    FirstDurationMs :Card; LastDurationMs :Card; MinDurationMs : Int'
18128:    64; MaxDurationMs :Card; MarksCount : Int; ArrayMarks : array of '
18129:    Cardinal; EnterCount : Int; ExitCount : Int; end;
18130:   SIRegister_TcyDebug(CL);
18131: end;
18132:
18133: (*----------------------------------------------------------------------------*)
18134: Proc SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
18135: begin
18136:   TCopyFileResult,(cfCreate, cfOverwrite,cfNoNeed, cfForceDirectoryError, cfCopyError );
18137:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent );
18138:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir );
18139:   SIRegister_TDestinationOptions(CL);
18140:   TProcCustomCopyFileEvent,Procedure(Sender:TObject;var CopyFileResult:TCopyFileResult);
18141: TProcOnCopyFileProgressEvent,Procedure(Sender:TObject;FileBytes,TransferedBytes:int64;PercentDone:Int64);
18142:   TProcOnCustomSetFileDestination','Proc (Sender:TObject;var FileName:str);
18143:   SIRegister_TcyCopyFiles(CL);
18144: Func cyCopyFile(FromFile,ToFile:Str; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
       ResetAttr:boolean): TCopyFileResult;
18145:  Func cyCopyFileEx(FromFile,ToFile:str;FileExists:TCopyFileExists;FileNotExists
       TCopyFileNotExists;ResetAttr:bool;aProgressBar:TProgressBar):TCopyFileResult;
18146:  Func cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters :Str; ArchiveFiles,
       ReadOnlyFiles, HiddenFiles,
       SystemFiles:TcyFileAttributeMode;FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;SubFolders,
       ResetAttributes:Boolean):Int;
18147: end;
18148:
18149: Proc SIRegister_cySearchFiles(CL: TPSPascalCompiler);
18150: begin
18151:   TypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth );
18152:   SIRegister_TcyFileAttributes(CL);
18153:   SIRegister_TSearchRecInstance(CL);
18154:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude );
18155:   TOptions', 'set of TOption;
18156:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting );
18157:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
       Accept:bool;
18158:   TProcOnValidateDirectoryEvent,Procedure(Sender:TObject;Directry:str;var Accept:bool);
18159:   SIRegister_TcyCustomSearchFiles(CL);
18160:   SIRegister_TcySearchFiles(CL);
18161:  Func FileNameRespondToMask( aFileName :Str; aMask :Str) :Bool;
18162:  Func IscyFolder( aSRec : TSearchrec) :Bool;
18163: end;
18164:
18165: Proc SIRegister_jcontrolutils(CL: TPSPascalCompiler);
18166: begin
18167:  Func jCountChar( const s :Str; ch : char) : Int;
18168:  Proc jSplit( const Delimiter : char; Input :Str; Strings : TStrings);
18169:  Func jNormalizeDate(const Value:str;theValue: TDateTime;const theFormat:str):str;
18170:  Func jNormalizeTime(const Value:str;theValue:TTime;const theFormat:str) :Str;
18171:  Func jNormalizeDateTime(const Val:str;theValue:TDateTime;const theFormat:str):str;
18172:  Func jNormalizeDateSeparator( const s :Str) :Str;
18173:  Func jIsValidDateString( const Value :Str) :Bool;
18174:  Func jIsValidTimeString( const Value :Str) :Bool;
18175:  Func jIsValidDateTimeString( const Value :Str) :Bool;
18176: end;
18177:
18178: Proc SIRegister_kcMapViewer(CL: TPSPascalCompiler);
18179: begin
18180:  ClassN(CL.FindClass('TOBJECT'),'TMapViewer;
18181:   TypeS('TMapSource', ( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
18182:    +gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
```

```
18183:    +, msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
18184:    +EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
18185:    +hooSatellite, msYahooHybrid,msOviNormal,msOviSatellite, msOviHybrid, msOviPhysical );
18186:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end;
18187:   TRealArea', 'record top: Extended; left:Extended; bottom: Extended; right: Extended; end;
18188:   TIntPoint', 'record X : Int64; Y : Int64; end;
18189:   TkcRealPoint', 'record X : Extended; Y : Extended; end;
18190:   TOnBeforeDownloadEvent','Procedure(Url:Str; str:TStream; var CanHandle:Bool);
18191:   TOnAfterDownloadEvent', 'Procedure( Url :Str; str : TStream);
18192:   SIRegister_TCustomDownloadEngine(CL);
18193:   SIRegister_TCustomGeolocationEngine(CL);
18194:   SIRegister_TMapViewer(CL);
18195: end;
18196:
18197: Proc SIRegister_cparserutils(CL: TPSPascalCompiler);
18198: begin
18199:  (*Func isFunc( name : TNamePart) :Bool;*)
18200:  Func isUnnamedFunc( name : TNamepart) :Bool;
18201:  Func isPtrToFunc( name : TNamePart) :Bool;
18202:  Func isFuncRetFuncPtr( name : TNamePart) :Bool;
18203:  Func isPtrToFuncRetFuncPtr( name : TNamePart) :Bool;
18204:  Func GetFuncParam( name : TNamePart) : TNamePart;
18205:  Func isArray( name : TNamePart) :Bool;
18206:  Func GetArrayPart( name : TNamePart) : TNamePart;
18207:  Func GetIdFromPart( name : TNamePart) : Ansistr;
18208:  Func GetIdPart( name : TNamePart) : TNamePart;
18209:  Func isNamePartPtrToFunc( part : TNamePart) :Bool;
18210:  Func isAnyBlock( part : TNamePart) :Bool;*)
18211:   TypeS('TLineInfo', record linestart : Int; lineend : Int; end;
18212:   SIRegister_TLineBreaker(CL);
18213:   TypeS('TNameKind', 'Int;
18214:   ClassN(CL.FindClass('TOBJECT'),'TNamePart;
18215:   //TypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end;
18216:  Func SphericalMod( X : Extended) : Extended;
18217:  Func cSign( Value : Extended) : Extended;
18218:  Func LimitFloat( const eValue, eMin, eMax : Extended) : Extended;
18219:  Func AngleToRadians( iAngle : Extended) : Extended;
18220:  Func RadiansToAngle( eRad : Extended) : Extended;
18221:  Func Cross180( iLong : Double) :Bool;
18222:  Func Mod180( Value : Int) : Int;
18223:  Func Mod180Float( Value : Extended) : Extended;
18224:  Func MulDivFloat( a, b, d : Extended) : Extended;
18225:  Func LongDiff( iLong1, iLong2 : Double) : Double;
18226:  Proc Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap);
18227:  Func Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap;
18228:  Func FixFilePath( const Inpath, CheckPath :Str) :Str;
18229:  Func UnFixFilePath( const Inpath, CheckPath :Str) :Str;
18230:  Proc FillStringList( sl : TStringList; const aText :Str);
18231: end;
18232:
18233: Proc SIRegister_LedNumber(CL: TPSPascalCompiler);
18234: begin
18235:   TLedSegmentSize', 'Int;
18236:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised );
18237:   SIRegister_TCustomLEDNumber(CL);
18238:   SIRegister_TLEDNumber(CL);
18239: end;
18240:
18241: Proc SIRegister_StStrL(CL: TPSPascalCompiler);
18242: begin
18243:   TypeS('LStrRec',record AllocSize:Longint;RefCount : Longint; Length : Longint; end;
18244:    TypeS(AnsiChar', 'Char;
18245:     TypeS('BTable', array[0..255] of Byte;       //!!!
18246:   Func HexBL( B : Byte) : Ansistr;
18247:  Func HexWL( W : Word) : Ansistr;
18248:  Func HexLL( L : LongInt) : Ansistr;
18249:  Func HexPtrL( P : ___Pointer) : Ansistr;
18250:  Func BinaryBL( B : Byte) : Ansistr;
18251:  Func BinaryWL( W : Word) : Ansistr;
18252:  Func BinaryLL( L : LongInt) : Ansistr;
18253:  Func OctalBL( B : Byte) : Ansistr;
18254:  Func OctalWL( W : Word) : Ansistr;
18255:  Func OctalLL( L : LongInt) : Ansistr;
18256:  Func Str2Int16L( const S : Ansistr; var I : SmallInt) :Bool;
18257:  Func Str2WordL( const S : Ansistr; var I : Word) :Bool;
18258:  Func Str2LongL( const S : Ansistr; var I : LongInt) :Bool;
18259:  Func Str2RealL( const S : Ansistr; var R : Double) :Bool;
18260:  Func Str2RealL( const S : Ansistr; var R : Real) :Bool;
18261:  Func Str2ExtL( const S : Ansistr; var R : Extended) :Bool;
18262:  Func Long2StrL( L : LongInt) : Ansistr;
18263:  Func Real2StrL( R : Double; Width : Byte; Places : ShortInt) : Ansistr;
18264:  Func Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : Ansistr;
18265:  Func ValPrepL( const S : Ansistr) : Ansistr;
18266:  Func CharStrL( C : Char; Len :Card) : Ansistr;
18267:  Func PadChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18268:  Func PadLL( const S : Ansistr; Len :Card) : Ansistr;
18269:  Func LeftPadChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18270:  Func LeftPadL( const S : Ansistr; Len :Card) : Ansistr;
18271:  Func TrimLeadL( const S : Ansistr) : Ansistr;
```

```
18272:  Func TrimTrailL( const S : Ansistr) : Ansistr;
18273:  Func TrimL( const S : Ansistr) : Ansistr;
18274:  Func TrimSpacesL( const S : Ansistr) : Ansistr;
18275:  Func CenterChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18276:  Func CenterL( const S : Ansistr; Len :Card) : Ansistr;
18277:  Func EntabL( const S : Ansistr; TabSize : Byte) : Ansistr;
18278:  Func DetabL( const S : Ansistr; TabSize : Byte) : Ansistr;
18279:  Func ScrambleL( const S, Key : Ansistr) : Ansistr;
18280:  Func SubstituteL( const S, FromStr, ToStr : Ansistr) : Ansistr;
18281:  Func FilterL( const S, Filters : Ansistr) : Ansistr;
18282:  Func CharExistsL( const S : Ansistr; C : AnsiChar) :Bool;
18283:  Func CharCountL( const S : Ansistr; C : AnsiChar) :Card;
18284:  Func WordCountL( const S, WordDelims : Ansistr) :Card;
18285:  Func WordPositionL(N:Card;const S,WordDelims:Ansistr;var Pos:Card):Bool;
18286:  Func ExtractWordL( N :Card; const S, WordDelims : Ansistr) : Ansistr;
18287:  Func AsciiCountL( const S, WordDelims : Ansistr; Quote : AnsiChar) :Card;
18288:  Func AsciiPositionL(N:Card;const S,WordDelims:AnsiStr;Quote:AnsiChr;var Pos:Card):Bool;
18289:  Func ExtractAsciiL(N:Card;const S,WordDelims:Ansistr;Quote:AnsiChar):Ansistr;
18290:  Proc WordWrapL(const InSt:Ansistr;var OutSt,Overlap:Ansistr;Margin:Card;PadToMargin:Bool;
18291:  Proc WordWrap(const InSt:AnsiStr;var OutSt,Overlap:AnsiStr;Margin:Card;PadToMargin:Bool;
18292:  Func CompStringL( const S1, S2 : Ansistr) : Int;
18293:  Func CompUCStringL( const S1, S2 : Ansistr) : Int;
18294:  Func SoundexL( const S : Ansistr) : Ansistr;
18295:  Func MakeLetterSetL( const S : Ansistr) : Longint;
18296:  Proc BMMakeTableL( const MatchString : Ansistr; var BT : BTable);
18297: Func BMSearchL(var Buffer,BufLength:Card;var BT:BTable;const MatchString:Ansistr;var Pos:Card):Bool;
18298:  Func BMSearchUCL( var Buffer, BufLength :Card; var BT: BTable; const MatchString: Ansistr; var Pos:Card)
       :Bool;
18299:  Func DefaultExtensionL( const Name, Ext : Ansistr) : Ansistr;
18300:  Func ForceExtensionL( const Name, Ext : Ansistr) : Ansistr;
18301:  Func JustFilenameL( const PathName : Ansistr) : Ansistr;
18302:  Func JustNameL( const PathName : Ansistr) : Ansistr;
18303:  Func JustExtensionL( const Name : Ansistr) : Ansistr;
18304:  Func JustPathnameL( const PathName : Ansistr) : Ansistr;
18305:  Func AddBackSlashL( const DirName : Ansistr) : Ansistr;
18306:  Func CleanPathNameL( const PathName : Ansistr) : Ansistr;
18307:  Func HasExtensionL( const Name : Ansistr; var DotPos :Card) :Bool;
18308:  Func CommaizeL( L : LongInt) : Ansistr;
18309:  Func CommaizeChL( L : Longint; Ch : AnsiChar) : Ansistr;
18310: Func FloatFormL(const Mask:Ansistr;R:TstFloat;const LtCurr,RtCurr:Ansistr;Sep,DecPt:Char):Ansistr;
18311: Func LongIntFormL(const Mask:Ansistr;L:LongInt;const LtCurr,RtCurr:AnsiStr;Sep:Char):Ansistr);
18312:  Func StrChPosL( const P : Ansistr; C : AnsiChar; var Pos :Card) :Bool;
18313:  Func StrStPosL( const P, S : Ansistr; var Pos :Card) :Bool;
18314:  Func StrStCopyL( const S : Ansistr; Pos, Count :Card) : Ansistr;
18315:  Func StrChInsertL( const S : Ansistr; C : AnsiChar; Pos :Card) : Ansistr;
18316:  Func StrStInsertL( const S1, S2 : Ansistr; Pos :Card) : Ansistr;
18317:  Func StrChDeleteL( const S : Ansistr; Pos :Card) : Ansistr;
18318:  Func StrStDeleteL( const S : Ansistr; Pos, Count :Card) : Ansistr;
18319:  Func ContainsOnlyL( const S, Chars : Ansistr; var BadPos :Card) :Bool;
18320:  Func ContainsOtherThanL(const S,Chars: Ansistr; var BadPos :Card):Bool;
18321:  Func CopyLeftL( const S : Ansistr; Len :Card) : Ansistr;
18322:  Func CopyMidL( const S : Ansistr; First, Len :Card) : Ansistr;
18323:  Func CopyRightL( const S : Ansistr; First :Card) : Ansistr;
18324:  Func CopyRightAbsL( const S : Ansistr; NumChars :Card) : Ansistr;
18325:  Func CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Card;var SubString:AString):Bool;
18326:  Func CopyFromToWordL(const S,WordDelims,Word1,Word2:Ansistr;N1,N2:Card;var SubString:Ansistr):Bool;
18327:  Func CopyWithinL( const S, Delimiter : Ansistr; Strip :Bool) : Ansistr;
18328:  Func DeleteFromNthWordL(const S, WordDelims:Ansistr;const AWord: Ansistr; N:Card; var SubString:
       Ansistr): Bool;
18329:  Func DeleteFromToWordL(const S,WordDelims,Word1,Word2:Ansistr;N1,N2:Card;var SubString:Ansistr):Bool;
18330:  Func DeleteWithinL( const S, Delimiter : Ansistr) : Ansistr;
18331: Func ExtractTokensL(const S,Delims:AnsiStr;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Card;
18332:  Func IsChAlphaL( C : AnsiChar) :Bool;
18333:  Func IsChNumericL( C : AnsiChar; const Numbers : Ansistr) :Bool;
18334:  Func IsChAlphaNumericL( C : AnsiChar; const Numbers : Ansistr) :Bool;
18335:  Func IsStrAlphaL( const S : Ansistr) :Bool;
18336:  Func IsStrNumericL( const S, Numbers : Ansistr) :Bool;
18337:  Func IsStrAlphaNumericL( const S, Numbers : Ansistr) :Bool;
18338:  Func KeepCharsL( const S, Chars : Ansistr) : Ansistr;
18339:  Func LastWordL(const S,WordDelims, AWord:Ansistr; var Position:Card):Bool;
18340:  Func LastWordAbsL( const S, WordDelims:Ansistr;var Position:Card):Bool;
18341:  Func LastStringL( const S, AString : Ansistr; var Position :Card):Bool;
18342:  Func LeftTrimCharsL( const S, Chars : Ansistr) : Ansistr;
18343: Func ReplaceWordL(const S,WordDelims,OldWord,NewWord:Ansistr;N:Card;var Replacements:Card):Ansistr;
18344: Func ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:Ansistr;var Replacements:Card):Ansistr;
18345: Func ReplaceStringL(const S,OldString,NewString:Ansistr;N:Card;var Replacements:Card):Ansistr;
18346: Func ReplaceStringAllL(const S,OldString,NewString:Ansistr; var Replacements:Card):Ansistr;
18347: Func RepeatStringL(const RepeatString:Ansistr;var Repetitions:Card;MaxLen:Card):Ansistr;
18348:  Func RightTrimCharsL( const S, Chars : Ansistr) : Ansistr;
18349:  Func StrWithinL(const S,SearchStr:str;Start:Card;var Position:Card): bool;
18350:  Func TrimCharsL( const S, Chars : Ansistr) : Ansistr;
18351:  Func WordPosL(const S,WordDelims,AWord:Ansistr;N:Card;var Position:Card):Bool;
18352:  Func WordPos(const S,WordDelims,AWord:Ansistr;N:Card;var Position:Card):Bool;
18353:  end;
18354:
18355: Proc SIRegister_pwnative_out(CL: TPSPascalCompiler);
18356: begin
18357:  ConstantN('STDIN','LongInt').SetInt( 0);
18358:  ('STDOUT','LongInt').SetInt( 1);
```

```
18359:  ('STDERR','LongInt').SetInt( 2);
18360:  Proc NativeWrite( s : astr);;
18361:  Proc NativeWrite1( PString : PChar);;
18362:  Proc NativeWrite2( Buffer : PChar; NumChars :Card);;
18363:  Proc NativeWriteLn( s : astr);;
18364:  Proc NativeWriteLn1;;
18365: end;
18366:
18367: Proc SIRegister_synwrap1(CL: TPSPascalCompiler);
18368: begin
18369:   TypeS(TSynwInfo', 'record Err : byte; UrlHtml : Ansistr; ErrResponse
18370:    : Int; UltimateURL : Ansistr; Headers : Ansistr; end;
18371:   TypeS('TUrlInfo', 'record Err : byte; UltimateURL :Str; end;
18372:  Func GetHttpFile(const Url,UserAgent,Outfile:str; verbose :Bool):TSynwInfo;
18373:  Func GetHttpFile1( const Url, UserAgent, Outfile :Str) : TSynwInfo;
18374:  Func GetHttpFile2( const Url, Outfile :Str) : TSynwInfo;;
18375:  Func GetHttpFile3( const Url, outfile :Str; verbose :Bool) : TSynwInfo;
18376:  Func GetHtm( const Url :Str) :Str;;
18377:  Func GetHtm1( const Url, UserAgent :Str) :Str;;
18378:  Func GetUrl( const Url :Str; verbose :Bool) : TSynwInfo;;
18379:  Func GetUrl1( const Url, useragent :Str) : TSynwInfo;;
18380:  Func GetUrl2( const Url :Str) : TSynwInfo;;
18381:  Func GetUrl3(const Url:str;const http THTTPSend; verbose :Bool): TUrlInfo;
18382:  Func GetUrl4( const Url :Str; const http : THTTPSend) : TUrlInfo;;
18383:  Proc StrToStream( s :Str; strm : TMemoryStream);
18384:  Func StrLoadStream( strm : TStream) :Str;
18385: end;
18386:
18387: Proc SIRegister_HTMLUtil(CL: TPSPascalCompiler);
18388: begin
18389:  Func GetVal( const tag, attribname_ci :Str) :Str;
18390:  Func GetTagName( const Tag :Str) :Str;
18391:  Func GetUpTagName( const tag :Str) :Str;
18392:  Func GetNameValPair( const tag, attribname_ci :Str) :Str;
18393:  Func GetValFromNameVal( const namevalpair :Str) :Str;
18394:  Func GetNameValPair_cs( const tag, attribname :Str) :Str;
18395:  Func GetVal_JAMES( const tag, attribname_ci :Str) :Str;
18396:  Func GetNameValPair_JAMES( const tag, attribname_ci :Str) :Str;
18397:  Func CopyBuffer( StartIndex : PChar; Len : Int) :Str;
18398:  Func Ucase( s :Str) :Str;
18399: end;
18400:
18401: Proc SIRegister_pwmain(CL: TPSPascalCompiler);
18402: begin
18403:  ConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT;
18404:  EXPIRED_COOKIE','String').SetString( 'Mon, 01 Jan 2001 12:00:00 GMT;
18405:  'SECURE_OFF','LongInt').SetInt( 0);
18406:  'SECURE_ON','LongInt').SetInt( 2);
18407:  'SECURE_FILTER','LongInt').SetInt( 3);
18408:   THandle or DWord!
18409:  //  astr = Ansistr;
18410:   TypeS('pastr', 'Ansistr;
18411:   TypeS('TFilterFunc', 'function(const s: pastr): pastr;;
18412:   uses pwinit at begin
18413:   //type  TFilterFunc = function(const s: astr): astr;
18414:  Demo: ..\maxbox3\examples2\519_powtils.txt
18415:  //ConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false);
18416:  //ConstantN('CASE_IGNORE','Boolean')BoolToStr( false);
18417:  Proc pwInit;
18418:  Proc OffReadln;
18419:  Func Lcase( const s : pastr) : pastr;
18420:  Func Ucase( const s : pastr) : pastr;
18421:  Func CountPostVars : longword;
18422:  Func GetPostVar( const name : pastr) : pastr;;
18423:  Func GetPostVar1( const name : pastr; filter : TFilterFunc) : pastr;;
18424:  Func GetPostVar_S( const name : pastr; Security : Int) : pastr;
18425:  Func GetPostVar_SF( const name : pastr; Security : Int) : pastr;
18426:  Func GetPostVarAsFloat( const name : pastr) : double;
18427:  Func GetPostVarAsInt( const name : pastr) : longint;
18428:  Func GetPostVar_SafeHTML( const name : pastr) : pastr;
18429:  Func FetchPostVarName( idx : longword) : pastr;
18430:  Func FetchPostVarVal( idx : longword) : pastr;;
18431:  Func FetchPostVarVal1( idx : longword; filter : TFilterFunc) : pastr;;
18432:  Func FetchPostVarName_S( idx : longword; Security : Int) : pastr;
18433:  Func FetchPostVarVal_S( idx : longword; Security : Int) : pastr;
18434:  Func IsPostVar( const name : pastr) :Bool;
18435:  Func CountAny : longword;
18436:  Func GetAny( const name : pastr) : pastr;;
18437:  Func GetAny1( const name : pastr; filter : TFilterFunc) : pastr;;
18438:  Func GetAny_S( const name : pastr; Security : Int) : pastr;
18439:  Func GetAnyAsFloat( const name : pastr) : double;
18440:  Func GetAnyAsInt( const name : pastr) : longint;
18441:  Func IsAny( const name : pastr) : byte;
18442:  Func CountCookies : longword;
18443:  Func FetchCookieName( idx : longword) : pastr;
18444:  Func FetchCookieVal( idx : longword) : pastr;;
18445:  Func FetchCookieVal1( idx : longword; filter : TFilterFunc) : pastr;;
18446:  Func GetCookie( const name : pastr) : pastr;;
18447:  Func GetCookie1( const name : pastr; filter : TFilterFunc) : pastr;;
```

```
18448:  Func GetCookieAsFloat( const name : pastr) : double;
18449:  Func GetCookieAsInt( const name : pastr) : longint;
18450:  Func IsCookie( const name : pastr) :Bool;
18451:  Func SetCookie( const name, value : pastr) :Bool;
18452:  Func SetCookieAsFloat( const name : pastr; value : double) :Bool;
18453:  Func SetCookieAsInt( const name : pastr; value : longint) :Bool;
18454:  Func SetCookieEx( const name, value, path, domain, expiry : pastr) :Bool;
18455:  Func SetCookieAsFloatEx(const name:pastr;value:double;const path,domai,expiry:pastr):bool;
18456:  Func SetCookieAsIntEx(const name:pastr;value:longint;const path,domain,expiry:pastr):bool;
18457:  Func UnsetCookie( const name : pastr) :Bool;
18458:  Func UnsetCookieEx( const name, path, domain : pastr) :Bool;
18459:  Func FilterHtml( const input : pastr) : pastr;
18460:  Func FilterHtml_S( const input : pastr; security : Int) : pastr;
18461:  Func TrimBadChars( const input : pastr) : pastr;
18462:  Func TrimBadFile( const input : pastr) : pastr;
18463:  Func TrimBadDir( const input : pastr) : pastr;
18464:  Func TrimBad_S( const input : pastr; security : Int) : pastr;
18465:  Func CountHeaders : longword;
18466:  Func FetchHeaderName( idx : longword) : pastr;
18467:  Func FetchHeaderVal( idx : longword) : pastr;
18468:  Func GetHeader( const name : pastr) : pastr;
18469:  Func IsHeader( const name : pastr) :Bool;
18470:  Func SetHeader( const name, value : pastr) :Bool;
18471:  Func UnsetHeader( const name : pastr) :Bool;
18472:  Func PutHeader( const header : pastr) :Bool;
18473:  Proc Out1( const s : pastr);
18474:  Proc OutLn( const s : pastr);
18475:  Proc OutA( args : array of const);
18476:  Proc OutF( const s : pastr);
18477:  Proc OutLnF( const s : pastr);
18478:  Proc OutFF( const s : pastr);
18479:  Proc OutF_FI( const s : pastr; HTMLFilter :Bool);
18480:  Proc OutLnFF( const s : pastr);
18481:  Proc OutLnF_FI( const s : pastr; HTMLFilter :Bool);
18482:  Func FileOut( const fname : pastr) : word;
18483:  Func ResourceOut( const fname : pastr) : word;
18484:  Proc BufferOut( const buff, len : LongWord);
18485:  Func TemplateOut( const fname : pastr; HtmlFilter :Bool) : word;;
18486:  Func TemplateOut1( const fname : pastr) : word;;
18487:  Func TemplateOut2( const fname : pastr; filter : TFilterFunc) : word;;
18488:  Func TemplateOut3( const fname : pastr; HtmlFilter :Bool) : word;;
18489:  Func TemplateRaw( const fname : pastr) : word;
18490:  Func Fmt( const s : pastr) : pastr;;
18491:  Func Fmt1( const s : pastr; filter : TFilterFunc) : pastr;;
18492:  Func FmtFilter( const s : pastr) : pastr;
18493: Func Fmt_SF(const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecur,TrimSecurity:int):pastr;
18494:  Func Fmt_SF1(const s:pastr;HTMLFilter:bool;FilterSecurity,TrimSecurity:Int):pastr;;
18495:  Func CountRtiVars : longword;
18496:  Func FetchRtiName( idx : longword) : pastr;
18497:  Func FetchRtiVal( idx : longword) : pastr;
18498:  Func GetRti( const name : pastr) : pastr;
18499:  Func GetRtiAsFloat( const name : pastr) : double;
18500:  Func GetRtiAsInt( const name : pastr) : longint;
18501:  Func IsRti( const name : pastr) :Bool;
18502:  Proc SetRTI( const name, value : pastr);
18503:  Func FetchUpfileName( idx : longword) : pastr;
18504:  Func GetUpfileName( const name : pastr) : pastr;
18505:  Func GetUpfileSize( const name : pastr) : longint;
18506:  Func GetUpfileType( const name : pastr) : pastr;
18507:  Func CountUpfiles : longword;
18508:  Func IsUpfile( const name : pastr) :Bool;
18509:  Func SaveUpfile( const name, fname : pastr) :Bool;
18510:  Func CountVars : longword;
18511:  Func FetchVarName( idx : longword) : pastr;
18512:  Func FetchVarVal( idx : longword) : pastr;;
18513:  Func FetchVarVal1( idx : longword; filter : TFilterFunc) : pastr;;
18514:  Func GetVar( const name : pastr) : pastr;;
18515:  Func GetVar1( const name : pastr; filter : TFilterFunc) : pastr;;
18516:  Func GetVar_S( const name : pastr; security : Int) : pastr;
18517:  Func GetVarAsFloat( const name : pastr) : double;
18518:  Func GetVarAsInt( const name : pastr) : longint;
18519:  Proc SetVar( const name, value : pastr);
18520:  Proc SetVarAsFloat( const name : pastr; value : double);
18521:  Proc SetVarAsInt( const name : pastr; value : longint);
18522:  Func IsVar( const name : pastr) : byte;
18523:  Proc UnsetVar( const name : pastr);
18524:  Func LineEndToBR( const s : pastr) : pastr;
18525:  Func RandomStr( len : longint) : pastr;
18526:  Func XorCrypt( const s : pastr; key : byte) : pastr;
18527:  Func CountCfgVars : longword;
18528:  Func FetchCfgVarName( idx : longword) : pastr;
18529:  Func FetchCfgVarVal( idx : longword) : pastr;
18530:  Func IsCfgVar( const name : pastr) :Bool;
18531:  Func SetCfgVar( const name, value : pastr) :Bool;
18532:  Func GetCfgVar( const name : pastr) : pastr;
18533:  Proc ThrowErr( const s : pastr);
18534:  Proc ThrowWarn( const s : pastr);
18535:  Proc ErrWithHeader( const s : pastr);
18536: TypeS(TWebVar', 'record name : pastr; value : pastr; end;
```

```
18537: TypeS(TWebVars', 'array of TWebVar;
18538:  Func iUpdateWebVar(var webv:TWebVars; const name,value:pastr; upcased:boolean):boolean;
18539:  Func iAddWebCfgVar( const name, value : pastr) :Bool;
18540:  Proc iAddWebVar( var webv : TWebVars; const name, value : pastr);
18541:  Proc iSetRTI( const name, value : pastr);
18542:  Func iCustomSessUnitSet :Bool;
18543:  Func iCustomCfgUnitSet :Bool;
18544: end;
18545:
18546: Proc SIRegister_W32VersionInfo(CL: TPSPascalCompiler);
18547: begin
18548:   SIRegister_TProjectVersionInfo(CL);
18549:  Func MSLanguageToHex( const s :Str) :Str;
18550:  Func MSHexToLanguage( const s :Str) :Str;
18551:  Func MSCharacterSetToHex( const s :Str) :Str;
18552:  Func MSHexToCharacterSet( const s :Str) :Str;
18553:  Func MSLanguages : TStringList;
18554:  Func MSHexLanguages : TStringList;
18555:  Func MSCharacterSets : TStringList;
18556:  Func MSHexCharacterSets : TStringList;
18557: end;
18558:
18559: Proc SIRegister_IpUtils(CL: TPSPascalCompiler);
18560: begin
18561: TIpHandle', 'Cardinal;
18562: TIpMD5StateArray', 'array[0..3] of DWORD;;
18563: TypeS('TIpMD5CountArray', 'array[0..1] of DWORD;;
18564: TIpMD5ByteBuf', 'array[0..63] of Byte;;
18565: TIpMD5LongBuf', 'array[0..15] of DWORD;;
18566: TIpMD5Digest', 'array[0..15] of Byte;;
18567: TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther );
18568: TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5'
18569:  +'CountArray; ByteBuf : TIpMD5ByteBuf; end;
18570: ClassN(CL.FindClass('TOBJECT'),'EIpBaseException;
18571: ClassN(CL.FindClass('TOBJECT'),'EIpAccessException;
18572: ClassN(CL.FindClass('TOBJECT'),'EIpHtmlException;
18573: SIRegister_TIpBaseAccess(CL);
18574: SIRegister_TIpBasePersistent(CL);
18575: //TIpComponentClass', 'class of TIpBaseComponent;
18576: SIRegister_TIpBaseComponent(CL);
18577: SIRegister_TIpBaseWinControl(CL);
18578:  Func InClassA( Addr : LongInt) :Bool;
18579:  Func InClassB( Addr : LongInt) :Bool;
18580:  Func InClassC( Addr : LongInt) :Bool;
18581:  Func InClassD( Addr : LongInt) :Bool;
18582:  Func InMulticast( Addr : LongInt) :Bool;
18583:  Func IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar) : DWORD;
18584:  Func IpCompStruct( const S1, S2, Size :Card) : Int;
18585:  Func IpMaxInt( A, B : Int) : Int;
18586:  Func IpMinInt( A, B : Int) : Int;
18587:  Proc IpSafeFree( var Obj: TObject);
18588:  Func IpShortVersion :Str;
18589:  Func IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD) : DWORD;
18590:  Func IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD) : DWORD;
18591:  Func IpInternetSumOfFile( const FileName :Str) : DWORD;
18592:  Func MD5SumOfFile( const FileName :Str) :Str;
18593:  Func MD5SumOfStream( Stream : TStream) :Str;
18594:  Func MD5SumOfStreamDigest( Stream : TStream) : TIpMD5Digest;
18595:  Func MD5SumOfString( const S :Str) :Str;
18596:  Func MD5SumOfStringDigest( const S :Str) : TIpMD5Digest;
18597:  Func SafeYield : LongInt;
18598:  Func AllTrimSpaces( Strng :Str) :Str;
18599:  Func IpCharPos( C : AnsiChar; const S :Str) : Int;
18600:  Func CharPosIdx( C : AnsiChar; const S :Str; Idx : Int) : Int;
18601:  Func NthCharPos( C : AnsiChar; const S :Str; Nth : Int) : Int;
18602:  Func RCharPos( C : AnsiChar; const S :Str) : Int;
18603:  Func RCharPosIdx( C : AnsiChar; const S :Str; Idx : Int) : Int;
18604:  Func RNthCharPos( C : AnsiChar; const S :Str; Nth : Int) : Int;
18605:  Func IpRPos( const Substr :Str; const S :Str) : Int;
18606:  Func IpPosIdx( const SubStr, S :Str; Idx : Int) : Int;
18607:   ACharSet', 'set of AnsiChar;
18608:   TIpAddrRec', 'record Scheme:str; UserName :Str; Password string; Authority :Str;
18609:    Port:str;Path:str; Fragment:Str; Query :Str; QueryDelim : AnsiChar; end;
18610:  Proc Initialize( var AddrRec : TIpAddrRec);
18611:  Proc Finalize( var AddrRec : TIpAddrRec);
18612:  Func ExtractEntityName( const NamePath :Str) :Str;
18613:  Func ExtractEntityPath( const NamePath :Str) :Str;
18614:  Func IpParseURL( const URL :Str; var Rslt : TIpAddrRec) :Bool;
18615:  Func BuildURL( const OldURL, NewURL :Str) :Str;
18616:  Func PutEscapes( const S :Str; EscapeSet : ACharSet) :Str;
18617:  Func RemoveEscapes( const S :Str; EscapeSet : ACharSet) :Str;
18618:  Proc SplitParams( const Parms :Str; Dest : TStrings);
18619:  Func NetToDOSPath( const PathStr :Str) :Str;
18620:  Func DOSToNetPath( const PathStr :Str) :Str;
18621:  Proc SplitHttpResponse( const S :Str; var V, MsgID, Msg :Str);
18622:  Proc FieldFix( Fields : TStrings);
18623:  Func AppendSlash( APath :Str) :Str;
18624:  Func RemoveSlash( APath :Str) :Str;
18625:  Func GetParentPath( const Path :Str) :Str;
```

```
18626:  Func GetLocalContent( const TheFileName :Str) :Str;
18627:  Func IPDirExists( Dir :Str) :Bool;
18628:  Func GetTemporaryFile( const Path :Str) :Str;
18629:  Func GetTemporaryPath :Str;
18630:  Func AppendBackSlash( APath :Str) :Str;
18631:  Func IpRemoveBackSlash( APath :Str) :Str;
18632:  Func INetDateStrToDateTime( const DateStr :Str) : TDateTime;
18633:  Func DateTimeToINetDateTimeStr( DateTime : TDateTime) :Str;
18634:  Func IpTimeZoneBias : Int;
18635:  Proc SplitCookieFields( const Data :Str; Fields : TStrings);
18636: end;
18637:
18638: Proc SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18639: begin
18640:   TypeS('TPOStyle', '( postStandard, postPropName, postFull );
18641:  Proc Lrt2Po( const LRTFile :Str; POStyle : TPOStyle);
18642:  Proc CombinePoFiles( SL : TStrings; const FName :Str);
18643: end;
18644:
18645: Proc SIRegister_GPS(CL: TPSPascalCompiler);
18646: begin
18647:  ConstantN('MAX_SATS','LongInt').SetInt( 12);
18648:  GPSMSG_START','String').SetString( '$;
18649:  GPSMSG_STOP','String').SetString( '*;
18650:  SEC_BETWEEN_SEG','LongInt').SetInt( 5);
18651:   TypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor'
18652:    +'tint; Azimut : Smallint; SignLevel : Smallint; end;
18653:   TypeS('TSatellites', 'array[1..12] of TSatellite;
18654:  //TSatellites = array[1..MAX_SATS] of TSatellite;
18655:   TGPSSatEvent,Procedure(Sender: TObject; NbSat, NbSatUse: Shortint; Sats: TSatellites);
18656:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Heigh'
18657:    tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid :Bool; '
18658:    NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end;
18659:   TypeS('TGPSDatasEvent', Proc ( Sender : TObject; GPSDatas : TGPSDatas);
18660:   TypeS('TMsgGP',( msgGP,msgGPGGA,msgGPGLL,msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA );
18661:   TypeS('TSpeedUnit',( suKilometre, suMile, suNauticalMile );
18662:   SIRegister_TGPSLink(CL);
18663:   SIRegister_TCustomGPS(CL);
18664:   SIRegister_TGPS(CL);
18665:   SIRegister_TGPStoGPX(CL);
18666:   SIRegister_TGPSSpeed(CL);
18667:   SIRegister_TGPSSatellitesPosition(CL);
18668:   SIRegister_TGPSSatellitesReception(CL);
18669:   SIRegister_TGPSCompass(CL);
18670:  //Proc Register( );
18671:  Func IndexMsgGP( StrMsgGP :Str) : TMsgGP;
18672:  Func StrCoordToAngle( Point : Char; Angle :Str) : Double;
18673:  Func StrTimeToTime( const Time :Str) : TDateTime;
18674:  Func StrToInt( const Str :Str) : Int;
18675:  Func StrToReal( const Str :Str) : Extended;
18676:  Func GPSRotatePoint( Angle : Double; Ct, Pt : TPoint) : TPoint;
18677:  Proc LoadRessource( RessourceName :Str; ImageList : TImageList);
18678: end;
18679:
18680: Proc SIRegister_NMEA(CL: TPSPascalCompiler);
18681: begin
18682:  NMEADataArray', 'array of string;
18683:  Proc TrimNMEA( var S :Str);
18684:  Proc ExpandNMEA( var S :Str);
18685:  Func ParseNMEA( S :Str) : NMEADataArray;
18686:  Func ChkValidNMEA( S :Str) :Bool;
18687:  Func IdNMEA( S :Str) :Str;
18688:  Func ChkSumNMEA( const S :Str) :Str;
18689:  Func PosInDeg( const PosStr :Str) : Double;
18690:  Func DateTimeNMEA( const StrD, StrT :Str) : TDateTime;
18691:  Func SysClockSet( const StrD, StrT :Str) :Bool;
18692:  Func Ticks2Secs(Ticks : LongInt) : LongInt;;
18693:  Func Secs2Ticks(Secs : LongInt) : LongInt;;
18694:  Func MSecs2Ticks(MSecs : LongInt) : LongInt;;
18695: end;
18696:
18697: Proc SIRegister_SortUtils(CL: TPSPascalCompiler);
18698: begin
18699:   TypeS('SortType1', 'Byte;
18700:   TypeS('SortType2', 'Double;
18701:   TypeS('SortType3', 'DWord;
18702:   //TypeS('PDWordArray', '^DWordArray // will not work;
18703:   TypeS('TDataRecord4', 'record Value : Int; Data : Int; end;
18704:  Function('Proc QuickSort( var List : array of SortType1; Min, Max : Int);
18705:  Proc QuickSortDWord( var List : array of SortType3; Min, Max : Int);
18706:  Proc QuickSortDataRecord4( var List : array of TDataRecord4; Count : Int);
18707:  Proc HeapSort( var List : array of SortType1; Count : DWord; FirstNeeded : DWord);
18708:  Func QuickSelect(var List: array of SortType1; Min, Max, Wanted : Int) : SortType1;
18709:  Func QuickSelectDouble( var List : array of SortType2; Min, Max, Wanted : Int) : SortType2;
18710:  Func QuickSelectDWord( var List : array of SortType3; Min,Max,Wanted: Int):SortType3;
18711: end;
18712:
18713: Proc SIRegister_BitmapConversion(CL: TPSPascalCompiler);
18714: begin
```

```
18715:  // TMatrix3x3 = array[1..3,1..3] of Double;
18716:  // TMatrix4x4 = array[1..4,1..4] of Double;
18717:  TypeS('TMatrix3x31', 'array[1..3] of Double;
18718:  TypeS('TMatrix3x3', 'array[1..3] of TMatrix3x31;
18719:  TypeS('TMatrix4x41', 'array[1..4] of Double;
18720:  TypeS('TMatrix4x4', 'array[1..4] of TMatrix4x41;
18721:  Proc ColorTransform( A, B, C : Byte; out X, Y, Z : Byte; const T : TMatrix4x4);
18722:  Proc ColorTransform1( A, B, C : Byte; out X, Y, Z : Float; const T : TMatrix4x4);
18723:  Proc ColorTransform2(const A,B,C:Float; out X, Y, Z : Byte; const T: TMatrix4x4);
18724:  Proc ColorTransformHSI2RGB( H, S, I : Byte; out R, G, B : Byte);
18725:  Proc ColorTransformRGB2HSI( R, G, B : Byte; out H, S, I : Byte);
18726:  Proc ColorTransformRGB2Lab( R, G, B : Byte; out L, a_, b_ : Byte);
18727:  Proc ColorTransformLab2RGB( L, a_, b_ : Byte; out R, G, B : Byte);
18728:  Proc ColorTransformRGB2LOCO( R, G, B : Byte; out S0, S1, S2 : Byte);
18729:  Proc ColorTransformLOCO2RGB( S0, S1, S2 : Byte; out R, G, B : Byte);
18730:  Proc ConvertColorSpace(Image:TLinarBitmap;const T:TMatrix4x4;NewImage:TLinarBitmap);
18731: //Proc ConvertColorSpace1(Image:TLinarBitmap;ColorTransform:TColorTransformProc;NewImage:TLinarBitmap);
18732:  Proc ConvertToGrayscale( const Image, GrayImage : TLinarBitmap);;
18733:  Proc ConvertToGrayscale1( const Image : TLinarBitmap);;
18734: end;
18735:
18736: Proc SIRegister_ZDbcUtils(CL: TPSPascalCompiler);
18737: begin
18738:  { TZSQLType = (zsqlstUnknown, zsqlstBoolean, zsqlstByte, zsqlstShort, zsqlstInt, zsqlstLong,
18739:     zsqlstFloat,zsqlstDouble,zsqlstBigDecimal, zsqlstString, zsqlstUnicodeString, zsqlstBytes,
18740:     zsqlstDate, zsqlstTime, zsqlstTimestamp, zsqlstDataSet, zsqlstGUID,
18741:     stAsciiStream, stUnicodeStream, stBinaryStream);}
18742:  Func ResolveConnectionProtocol(Url:str; SupportedProtocols:TStringDynArray):Str;
18743:  //Proc ResolveDatabaseUrl( const Url:Str; Info:TStrings; var HostName:str; var Port:Int;var
       Database:str;var UserName:str; var Password :Str; ResultInfo : TStrings);
18744:  Func CheckConvertion( InitialType : TZSQLType; ResultType : TZSQLType) :Bool;
18745:  Func DefineColumnTypeName( ColumnType : TZSQLType) :Str;
18746:  Proc RaiseSQLException( E : Exception);
18747:  //Proc CopyColumnsInfo( FromList : TObjectList; ToList : TObjectList);
18748:  Func ToLikeString( const Value :Str) :Str;
18749:  Func GetSQLHexWideString( Value : PChar; Len : Int; ODBC :Bool) : WideString;
18750:  Func GetSQLHexAnsistr( Value : PChar; Len : Int; ODBC :Bool) : RawByteString;
18751:  Func GetSQLHexString( Value : PChar; Len : Int; ODBC :Bool) :Str;
18752:  Func WideStringStream( const AString : WideString) : TStream;
18753:  Func ConvertAdoToTypeName(FieldType: SmallInt):Str;
18754:  Func GetTableName(const AField: TField):Str;;
18755:  Func GetFieldName(const AField: TField):Str;;
18756: end;
18757:
18758: Proc SIRegister_JclTD32(CL: TPSPascalCompiler);
18759: TypeS('TSymbolInfo', 'record Size : Word; SymbolType : Word; end;
18760: //TypeS('PSymbolInfos', '^TSymbolInfos // will not work;
18761: SIRegister_TJclModuleInfo(CL);
18762: SIRegister_TJclLineInfo(CL);
18763: SIRegister_TJclSourceModuleInfo(CL);
18764: SIRegister_TJclSymbolInfo(CL);
18765: SIRegister_TJclProcSymbolInfo(CL);
18766: ClassN(CL.FindClass('TOBJECT'),'TJclLocalProcSymbolInfo;
18767: ClassN(CL.FindClass('TOBJECT'),'TJclGlobalProcSymbolInfo;
18768: SIRegister_TJclTD32InfoParser(CL);
18769: SIRegister_TJclTD32InfoScanner(CL);
18770: SIRegister_TJclPeBorTD32Image(CL);
18771: end;
18772:
18773: Proc SIRegister_JvIni(CL: TPSPascalCompiler);
18774: begin
18775:   TypeS('TReadObjectEvent', 'Func ( Sender : TObject; const Section, '
18776:    +'Item, Value :Str) : TObject;
18777:   TWriteObjectEvent,Procedure(Sender:TObject;const Section, Item:Str; Obj: TObject);
18778:  Func StringToFontStyles( const Styles :Str) : TFontStyles;
18779:  Func FontStylesToString( Styles : TFontStyles) :Str;
18780:  Func FontToString( Font : TFont) :Str;
18781:  Proc StringToFont( const Str :Str; Font : TFont);
18782:  Func RectToStr( Rect : TRect) :Str;
18783:  Func StrToRect( const Str :Str; const Def : TRect) : TRect;
18784:  Func JPointToStr( P : TPoint) :Str;
18785:  Func JStrToPoint( const Str :Str; const Def : TPoint) : TPoint;
18786:  Func DefProfileName :Str;
18787:  Func DefLocalProfileName :Str;
18788:  ConstantN('idnListItem','String').SetString( 'Item;
18789: end;
18790:
18791: Proc SIRegister_JvHtControls(CL: TPSPascalCompiler);
18792: begin
18793:  Proc ItemHtDrawEx(Canvas : TCanvas;Rect  TRect; const State:TOwnerDrawState; const Text :Str; const
       HideSelColor:Boolean; var PlainItem:Str; var Width: Int; CalcWidth:Boolean);
18794: Func ItemHtDraw(Canvas: TCanvas;Rect:TRect;const State TOwnerDrawState;const Text:str;const
       HideSelColor:Booln):str;
18795: Func ItemHtWidth(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;const
       HideSelColor:Bool):Int;
18796:  Func ItemHtPlain( const Text :Str) :Str;
18797:  Proc ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:str);
18798:  Func IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:str;MouseX,MouseY:Int;var HyperLink:str):Bool;
18799: end;
```

```
18800:
18801: Proc SIRegister_NeuralNetwork(CL: TPSPascalCompiler);
18802: begin
18803:   ClassN(CL.FindClass('TOBJECT'),'TNeuron;
18804:   TypeS(TSynapse', 'record W : Real; Connection : TNeuron; end;
18805:   TypeS(TAcson', 'record Alfa : Real; Beta : Real; Gama : Real; end;
18806:   SIRegister_TNeuron(CL);
18807:   SIRegister_TNeuronLayer(CL);
18808:   SIRegister_TNeuralNet(CL);
18809: end;
18810:
18811: Proc SIRegister_StExpr(CL: TPSPascalCompiler);
18812: begin
18813: //TypeS('PStFloat', '^TStFloat // will not work;
18814: TStMethod0Param', 'Func  : TStFloat;
18815: TStMethod1Param', 'Func ( Value1 : TStFloat) : TStFloat;
18816: TStMethod2Param', 'Func ( Value1, Value2 : TStFloat) : TStFloat;
18817: TStMethod3Param', 'Func ( Value1, Value2, Value3 : TStFloat) : TStFloat;
18818: TStGetIdentValueEvent', 'Proc ( Sender : TObject; const Ide'
18819:  +ntifier : Ansistr; var Value : TStFloat);
18820: TStToken', '( ssStart, ssInIdent, ssInNum, ssInSign, ssInExp, ss'
18821:  +Eol,ssNum,ssIdent,ssLPar,ssRPar,ssComma,ssPlus,ssMinus,ssTimes,ssDiv,ssEqual,ssPower);
18822: SIRegister_TStExpression(CL);
18823: TStExprErrorEvent', 'Proc ( Sender : TObject; ErrorNumber :'
18824:  + LongInt; const ErrorStr : Ansistr);
18825: SIRegister_TStExpressionEdit(CL);
18826:  Func AnalyzeExpr( const Expr : Ansistr) : Double;
18827:  Proc TpVal( const S : Ansistr; var V : Extended; var Code : Int);
18828: end;
18829:
18830: Proc SIRegister_GR32_Containers(CL: TPSPascalCompiler);
18831: begin
18832:  ConstantN('BUCKET_MASK','LongWord').SetUInt( $FF);
18833:  ConstantN('BUCKET_COUNT','LongInt').SetInt( BUCKET_MASK + 1);
18834:  Proc SmartAssign( Src, Dst : TPersistent; TypeKinds : TTypeKinds);
18835:  Proc Advance( var Node : TLinkedNode; Steps : Int);
18836: end;
18837:
18838: Proc SIRegister_StSaturn(CL: TPSPascalCompiler);
18839: begin
18840:  TStJupSatPos', 'record X: double; Y: Double; end;
18841:  TStJupSats record Io:TStJupSatPos;Europa:TStJupSatPos;Ganymede:TStJupSatPos;Callisto:TStJupSatPos;end;
18842:  Func ComputeSaturn( JD : Double) : TStEclipticalCord;
18843:  Func ComputePluto( JD : Double) : TStEclipticalCord;
18844:  Func ComputeVenus( JD : Double) : TStEclipticalCord;
18845:  Func ComputeMars( JD : Double) : TStEclipticalCord;
18846:  Func ComputeMercury( JD : Double) : TStEclipticalCord;
18847:  Func ComputeJupiter( JD : Double) : TStEclipticalCord;
18848:  Func ComputeUranus( JD : Double) : TStEclipticalCord;
18849:  Func ComputeNeptune( JD : Double) : TStEclipticalCord;
18850:  Func GetJupSats(JD : TDateTime; HighPrecision, Shadows :Bool) : TStJupSats;;
18851: end;
18852:
18853: Proc SIRegister_JclParseUses(CL: TPSPascalCompiler);
18854: begin
18855:   ClassN(CL.FindClass('TOBJECT'),'EUsesListError;
18856:  Func CreateGoal( Text : PChar) : TCustomGoal;
18857: end;
18858:
18859: Proc SIRegister_JvFinalize(CL: TPSPascalCompiler);
18860: begin
18861: //type
18862: //TFinalizeProc = procedure;
18863:   TypeS('TFinalizeProc', 'procedure;
18864:  Proc AddFinalizeProc( const UnitName :Str; FinalizeProc : TFinalizeProc);
18865:  Func AddFinalizeObject( const UnitName :Str; Instance : TObject) : TObject;
18866:  Func AddFinalizeObjectNil(const UnitName:str; var Reference: TObject):TObject;
18867:  Func AddFinalizeFreeAndNil(const UnitName:str; var Reference: TObject) : TObject;
18868:  Func AddFinalizeMemory( const UnitName :Str; Ptr : ___Pointer) : ___Pointer;
18869:  Func AddFinalizeMemoryNil(const UnitName:Str;var Ptr: ___Pointer) : ___Pointer;
18870:  Proc FinalizeUnit( const UnitName :Str);
18871: end;
18872:
18873: Proc SIRegister_BigIni(CL: TPSPascalCompiler);
18874: begin
18875:  ConstantN('IniTextBufferSize','LongWord').SetUInt( $7000);
18876:  cIniCount','String').SetString( 'Count;
18877:  TEraseSectionCallback', 'Func ( const sectionName :Str; '
18878:   const sl1, sl2 : TStringList) :Bool;
18879:   SIRegister_TCommaSeparatedInfo(CL);
18880:   SIRegister_TSectionList(CL);
18881:   SIRegister_TBigIniFile(CL);
18882:   SIRegister_TBiggerIniFile(CL);
18883:   SIRegister_TAppIniFile(CL);
18884:   SIRegister_TLibIniFile(CL);
18885:  Func ModuleName( getLibraryName :Bool) :Str;
18886: end;
18887:
18888: Proc SIRegister_ShellCtrls(CL: TPSPascalCompiler);
```

```
18889: begin
18890:   TypeS(TRoot', 'string;
18891:   TypeS(TRootFolder', '( rfDesktop, rfMyComputer, rfNetwork, rfRecycleBi'
18892:    +n, rfAppData, rfCommonDesktopDirectory, rfCommonPrograms, rfCommonStartMen'
18893:    +u, rfCommonStartup, rfControlPanel, rfDesktopDirectory, rfFavorites, rfFon'
18894:    +ts, rfInternet, rfPersonal, rfPrinters, rfPrintHood, rfPrograms, rfRecent,'
18895:    + rfSendTo, rfStartMenu, rfStartup, rfTemplates );
18896:   TShellFolderCapability', '( fcCanCopy, fcCanDelete, fcCanLink, f'
18897:    +'cCanMove, fcCanRename, fcDropTarget, fcHasPropSheet );
18898:   TShellFolderCapabilities', 'set of TShellFolderCapability;
18899:   TShellFolderProperty', '( fpCut, fpIsLink, fpReadOnly, fpShared,'
18900:    + fpFileSystem, fpFileSystemAncestor, fpRemovable, fpValidate );
18901:   TShellFolderProperties', 'set of TShellFolderProperty;
18902:   TShellObjectType', '( otFolders, otNonFolders, otHidden );
18903:   TShellObjectTypes', 'set of TShellObjectType;
18904:   ClassN(CL.FindClass('TOBJECT'),'EInvalidPath;
18905:   SIRegister_IShellCommandVerb(CL);
18906:   SIRegister_TShellFolder(CL);
18907:   TNotifyFilter', '( nfFileNameChange, nfDirNameChange, nfAttribut'
18908:    +eChange, nfSizeChange, nfWriteChange, nfSecurityChange );
18909:   TNotifyFilters', 'set of TNotifyFilter;
18910:   SIRegister_TShellChangeThread(CL);
18911:   SIRegister_TCustomShellChangeNotifier(CL);
18912:   SIRegister_TShellChangeNotifier(CL);
18913:   ClassN(CL.FindClass('TOBJECT'),'TCustomShellComboBox;
18914:   ClassN(CL.FindClass('TOBJECT'),'TCustomShellListView;
18915:   TAddFolderEvent', 'Proc ( Sender : TObject; AFolder: TShel'
18916:    +lFolder; var CanAdd :Bool);
18917:   TGetImageIndexEvent', 'Proc ( Sender : TObject; Index : Int'
18918:    +'eger; var ImageIndex : Int);
18919:   SIRegister_TCustomShellTreeView(CL);
18920:   SIRegister_TShellTreeView(CL);
18921:   SIRegister_TCustomShellComboBox(CL);
18922:   SIRegister_TShellComboBox(CL);
18923:   SIRegister_TCustomShellListView(CL);
18924:   SIRegister_TShellListView(CL);
18925:  Proc InvokeContextMenu(Owner : TWinControl; AFolder : TShellFolder; X, Y : Int);
18926: end;
18927:
18928: Proc SIRegister_fmath(CL: TPSPascalCompiler);
18929: begin
18930:  TypeS('Float', 'Double; !
18931:  'FN_OK','LongInt').SetInt( 0);
18932:  ConstantN('FN_DOMAIN','LongInt').SetInt( - 1);
18933:  'FN_SING','LongInt').SetInt( - 2);
18934:  'FN_OVERFLOW','LongInt').SetInt( - 3);
18935:  'FN_UNDERFLOW','LongInt').SetInt( - 4);
18936:  'FN_TLOSS','LongInt').SetInt( - 5);
18937:  'FN_PLOSS','LongInt').SetInt( - 6);
18938:  //ConstantN('NFACT','LongInt').SetInt( 33);
18939:  Func MathError : Int;
18940:  Func FMin( X, Y : Float) : Float;
18941:  Func FMax( X, Y : Float) : Float;
18942:  Func IMin( X, Y : Int) : Int;
18943:  Func IMax( X, Y : Int) : Int;
18944:  Func FSgn( X : Float) : Int;
18945:  Func Sgn0( X : Float) : Int;
18946:  Func DSgn( A, B : Float) : Float;
18947:  Proc FSwap( var X, Y : Float);
18948:  Proc ISwap( var X, Y : Int);
18949:  Func fExpo( X : Float) : Float;
18950:  Func fExp2( X : Float) : Float;
18951:  Func fExp10( X : Float) : Float;
18952:  Func fLog( X : Float) : Float;
18953:  Func fLog2( X : Float) : Float;
18954:  Func fLog10( X : Float) : Float;
18955:  Func fLogA( X, A : Float) : Float;
18956:  Func fIntPower( X : Float; N : Int) : Float;
18957:  Func fPower( X, Y : Float) : Float;
18958:  Func Pythag( X, Y : Float) : Float;
18959:  Func FixAngle( Theta : Float) : Float;
18960:  Func fTan( X : Float) : Float;
18961:  Func fArcSin( X : Float) : Float;
18962:  Func fArcCos( X : Float) : Float;
18963:  Func fArcTan2( Y, X : Float) : Float;
18964:  Proc fSinCos( X : Float; var SinX, CosX : Float);
18965:  Func fSinh( X : Float) : Float;
18966:  Func fCosh( X : Float) : Float;
18967:  Func fTanh( X : Float) : Float;
18968:  Func fArcSinh( X : Float) : Float;
18969:  Func fArcCosh( X : Float) : Float;
18970:  Func fArcTanh( X : Float) : Float;
18971:  Proc fSinhCosh( X : Float; var SinhX, CoshX : Float);
18972:  Func fFact( N : Int) : Float;
18973:  Func fBinomial( N, K : Int) : Float;
18974:  Func fGamma( X : Float) : Float;
18975:  Func fSgnGamma( X : Float) : Int;
18976:  Func LnGamma( X : Float) : Float;
18977:  Func fIGamma( A, X : Float) : Float;
```

```
18978:  Func fJGamma( A, X : Float) : Float;
18979:  Func fBeta( X, Y : Float) : Float;
18980:  Func fIBeta( A, B, X : Float) : Float;
18981:  Func fErf( X : Float) : Float;
18982:  Func fErfc( X : Float) : Float;
18983:  Func fPBinom(N: Int; P: Float; K : Int) : Float;
18984:  Func FBinom(N: Int; P: Float; K : Int) : Float;
18985:  Func PPoisson( Mu : Float; K : Int) : Float;
18986:  Func FPoisson( Mu : Float; K : Int) : Float;
18987:  Func fDNorm( X : Float) : Float;
18988:  Func FNorm( X : Float) : Float;
18989:  Func PNorm( X : Float) : Float;
18990:  Func InvNorm( P : Float) : Float;
18991:  Func fDStudent( Nu : Int; X : Float) : Float;
18992:  Func FStudent( Nu : Int; X : Float) : Float;
18993:  Func PStudent( Nu : Int; X : Float) : Float;
18994:  Func fDKhi2( Nu : Int; X : Float) : Float;
18995:  Func FKhi2( Nu : Int; X : Float) : Float;
18996:  Func PKhi2( Nu : Int; X : Float) : Float;
18997:  Func fDSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
18998:  Func FSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
18999:  Func PSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
19000:  Func fDExpo( A, X : Float) : Float;
19001:  Func FExpo( A, X : Float) : Float;
19002:  Func fDBeta( A, B, X : Float) : Float;
19003:  Func FBeta( A, B, X : Float) : Float;
19004:  Func fDGamma( A, B, X : Float) : Float;
19005:  Func FGamma( A, B, X : Float) : Float;
19006:  Proc RMarIn( Seed1, Seed2 : Int);
19007:  Func IRanMar : LongInt;
19008:  Func RanMar : Float;
19009:  Func RanGaussStd : Float;
19010:  Func RanGauss( Mu, Sigma : Float) : Float;
19011: end;
19012:
19013: Proc SIRegister_fcomp(CL: TPSPascalCompiler);
19014: begin
19015:  TypeS('ComplexForm', '( Rec, Pol );
19016:  TypeS(TComplex', 'record Form:ComplexForm; X Float;Y: Float; R:Float; Theta:Float; end;
19017:  Proc CSet( var Z : TComplex; A, B : Float; F : ComplexForm);
19018:  Proc CConvert( var Z : TComplex; F : ComplexForm);
19019:  Proc CSwap( var X, Y : TComplex);
19020:  Func CReal( Z : TComplex) : Float;
19021:  Func CImag( Z : TComplex) : Float;
19022:  Func CAbs( Z : TComplex) : Float;
19023:  Func CArg( Z : TComplex) : Float;
19024:  Func CSgn( Z : TComplex) : Int;
19025:  Proc CNeg( A : TComplex; var Z : TComplex);
19026:  Proc CConj( A : TComplex; var Z : TComplex);
19027:  Proc CAdd( A, B : TComplex; var Z : TComplex);
19028:  Proc CSub( A, B : TComplex; var Z : TComplex);
19029:  Proc CDiv( A, B : TComplex; var Z : TComplex);
19030:  Proc CMult( A, B : TComplex; var Z : TComplex);
19031:  Proc CLn( A : TComplex; var Z : TComplex);
19032:  Proc CExp( A : TComplex; var Z : TComplex);
19033:  Proc CPower( A, B : TComplex; var Z : TComplex);
19034:  Proc CIntPower( A : TComplex; N : Int; var Z : TComplex);
19035:  Proc CRealPower( A : TComplex; X : Float; var Z : TComplex);
19036:  Proc CSqrt( A : TComplex; var Z : TComplex);
19037:  Proc CRoot( A : TComplex; K, N : Int; var Z : TComplex);
19038:  Proc CCSin( A : TComplex; var Z : TComplex);
19039:  Proc CCos( A : TComplex; var Z : TComplex);
19040:  Proc CTan( A : TComplex; var Z : TComplex);
19041:  Proc CArcSin( A : TComplex; var Z : TComplex);
19042:  Proc CArcCos( A : TComplex; var Z : TComplex);
19043:  Proc CArcTan( A : TComplex; var Z : TComplex);
19044:  Proc CSinh( A : TComplex; var Z : TComplex);
19045:  Proc CCosh( A : TComplex; var Z : TComplex);
19046:  Proc CTanh( A : TComplex; var Z : TComplex);
19047:  Proc CArcSinh(A: TComplex; var Z : TComplex);
19048:  Proc CArcCosh(A: TComplex; var Z : TComplex);
19049:  Proc CArcTanh(A: TComplex; var Z : TComplex);
19050:  Proc CLnGamma(A: TComplex; var Z : TComplex);
19051: end;
19052:
19053:  XSDd:= DateTimeToXSDateTime(now,true)
19054:      xsdd.nativetoxs;
19055:      writeln(datetimetostr(xsdd.asUTCdatetime));
19056:      xsdd.free;
19057:
19058: Proc SIRegister_XSBuiltIns(CL: TPSPascalCompiler);
19059: begin
19060:  ConstantN('SHexMarker','String').SetString( '$;
19061:  Func DateTimeToXMLTime( Value : TDateTime; ApplyLocalBias :Bool) : WideString;
19062:  Func XMLTimeToDateTime(const XMLDateTime : WideString; AsUTCTime:Boolean): TDateTime;
19063:  Func DateTimeToXSDateTime(const Value:TDateTime;ApplyLocalBias:Bool):TXSDateTime;
19064:  Func GetDataFromFile( AFileName :Str) :Str;
19065:  Func SoapFloatToStr( Value : double) :Str;
19066:  Func SoapStrToFloat( Value :Str) : double;
```

```
19067:  Proc InitXSTypes;
19068: end;
19069:
19070: Proc SIRegister_CompFileIo(CL: TPSPascalCompiler);
19071: begin
19072:  SIRegister_TComponentStream(CL);
19073:  TypeS('TComponentStream', 'TMemoryStream;
19074:  TComponentFileFormat', '( cffText, cffBinary );
19075:  TComponentArray', 'array of TComponent;
19076:  TResourceNaming,rnClassNameTag,rnClassName,rnClassTag,rnNameTag,rnClass,rnName, rnTag );
19077:  Func VOID_COMP : ___Pointer;
19078:  Func VOID_FORM : ___Pointer;
19079:  Func CreateIoForm( AForm : TCustomForm; ClassType : TFormClass) : TCustomForm;
19080: Func GetComponentTree(Component:TComponent;pComponents:TComponentArray;bAddSelf:Bool):LongInt;
19081:  Func pPosEx( SearchStr: PChar; Str : PChar; var Pos : LongInt) : PChar;
19082:  Func pGetTextBetween(pBuff:PChar;bSearchCode:str;eSearchCode:str;Container:TStrings): LongInt;
19083:  Func pComponentToString( Component : TComponent) :Str;
19084:  //Func StringToComponent(Value:str; ComponentClass:TComponentClass): TComponent;
19085:  Func StringToObjectBinaryStream(Value:str;BinStream:TMemoryStream;ResName:str):Bool;
19086: Func ObjectBinaryStreamToString(BinStream:TMemoryStream;var sResult:str;bResource:Bool):Bool
19087:  Func ObjectTextToBinaryStream(StrStream,BinStream:TComponentStream;ObjCount:LongInt): LongInt;
19088:  Func
       ObjectBinaryStreamToObjectTextStream(BinStream:TMemoryStream;StrStream:TMemoryStream;bResource:Bool):Bool;
19089:  Func GetResHeaderInfo( Stream : TMemoryStream; sl : TStrings) :Bool;
19090:  Func pGetResourceName(Component:TComponent;NamingMethod: TResourceNaming) :Str;
19091: Func GetFormResourceStream(Form:TCustomForm;ResName:str;var ResourceStream:TMemoryStream):Bool;
19092:  Func WriteComponentsToFile(FormsAndComponents:array of
       TComponent;FileName:str;Format:TComponentFileFormat; StoreComponentNames:Boolean):Boolean;
19093:  Func ReadComponentsFromFile(FormsAndComponents:array of TComponent;FileName:str):Bool;
19094:  //Func ReadComponentsFromFile1(FileName:str;pComponents: array of TPComponent;ComponentClasses:array of
       TComponentClass):Bool;
19095:  Func ReadComponentTreeFromFile(FormOrComponent:TComponent;FileName:Str):Boolean;
19096:  Func
       WriteComponentToFile(Component:TComponent;FileName:str;Format:TComponentFileFormat;StoreComponentName:Bool):Bool;
19097:  Func
       WriteComponentTreeToFile(FormOrComponent:TComponent;FileName:str;Format:TComponentFileFormat;StoreComponentName:Bool):
19098:  Func ReadComponentFromFile4(FormOrComponent:TComponent; FileName :Str):Bool;;
19099:  Func ReadFormFromFile(pInstance:TComponent;FormClass:TFormClass;FileName:str): Bool;
19100:  Func ReadComponentResourceFile5( Instance : TObject; FileName :Str) :Bool;;
19101: Func WriteComponentResourceFile(nstance:TObj;FileName:str;StoreFormAsVisible:Bool):Bool;
19102:  Func ReadComponentsResourceFile(Components:array of TComponent;FileName:str;NamingMethod:TResourceNaming;
       bLoadTotalForm:Boolean):Boolean;
19103:  Func WriteComponentsResourceFile( Components : array of TComponent; FileName:Str; NamingMethod :
       TResourceNaming) :Bool;
19104:  Func ReadComponentResourceHeader(sHeaderInfo:TStrings; pSize : Int; FileName:Str; NamingMethod :
       TResourceNaming) :Bool;
19105:  Func ConvertComponentResourceToTextFile(SourceFileName:str;TargetFileName:str;bStoreResNames:Bool):Bool;
19106:  Func CheckComponentInResourceFile(Component:TComponent;FileName:str;NamingMethod:TResourceNaming):Boolean;
19107:  Func DeleteComponentFromResourceFile(ResourceName:str; FileName:str):Bool;;
19108:  Func DeleteComponentFromResourceFile8(Component:TComponent;FileName
       string;NamingMethod:TResourceNaming):Bool;
19109: //Func CreateFormFromResFile(AOwner:TComponent;NewClassType:TFormClass;FileName:str): Pointer;
19110:  //Func CreateComponentFromResFile( AOwner : TComponent; NewClassType : TComponentClass; FileName :Str) :
       Pointer;
19111: end;
19112:
19113: Proc SIRegister_SMScript(CL: TPSPascalCompiler);
19114: begin
19115:   CL.FindClass('TOBJECT'),'TSMScriptExecutor;
19116:   SIRegister_TSMSEError(CL);
19117:   ClassN(CL.FindClass('TOBJECT'),'TSMSEModule;
19118:   TypeS('TSMSEProcedureType', '( ptProcedure, ptFunc );
19119:   SIRegister_TSMSEProcedure(CL);
19120:   SIRegister_TSMSEProcedures(CL);
19121:   SIRegister_TSMSEModule(CL);
19122:   SIRegister_TSMSEModules(CL);
19123:   TypeS('TSMScriptLanguage', ( slCustom, slVBScript, slJavaScript );
19124:   SIRegister_TSMScriptExecutor(CL);
19125:  //DelphiFunction('Proc Register;
19126: end;
19127:
19128: Proc SIRegister_geometry2(CL: TPSPascalCompiler);
19129: begin
19130:   TPointF2', 'record X : Double; Y : Double; end;
19131:   TPoint3', 'record X : Int; Y : Int; Z : Int; end;
19132:   //TypeS('PPointF3', '^TPointF3 // will not work;
19133:   TPointF3', 'record X : double; Y : double; Z : Double; end;
19134:  Func AreLinesParallel1( p1, p2, p3, p4 : TPoint) :Bool;;
19135:  Func AreLinesParallel1( p1, p2, p3, p4 : TPointF2; e : Double) :Bool;;
19136:  Func AreLinesParallel2( p1, p2, p3, p4, p5, p6 : TPoint3) :Bool;;
19137:  Func AreLinesParallel3( p1, p2, p3, p4, p5, p6 : TPointF3; e : Double) :Bool;;
19138:  Func IntersectLines( p1, p2, p3, p4 : TPoint) : TPoint;;
19139:  Func IntersectLines1( p1, p2, p3, p4 : TPointF2; e : Double) : TPointF2;;
19140:  Func AngleDifference( alpha, beta : Double) : Double;
19141: end;
19142:
19143:  source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
19144:  New Functions /Classes mX4:
19145:  Func DownloadJPGToBitmap(const URL :Str; ABitmap: TBitmap):Bool;);
```

```
19146:  Proc GetImageLinks(AURL:Str; AList: TStrings);;
19147:  Proc SaveByteCode;;
19148:  Proc ResetKeyPressed;;
19149:  Proc SetKeyPressed;;
19150:
19151:  RIRegister_HTTPProd_Routines(Exec);
19152:  Func ContentFromScriptStream(AStream:TStream; AWebModuleContext : TWebModuleContext; AStripParamQuotes :
        Bool; AHandleTag : THandleTagProc; AHandleScriptTag : THandledTagProc; const AScriptEngine :Str;
        ALocateFileService : ILocateFileService) :Str;
19153:  Func ContentFromScriptFile(const AFileName:TFileName; AWebModuleContext:TWebModuleContext;
        AStripParamQuotes :Bool; AHandleTag:THandleTagProc;AHandleScriptTag:THandledTagProc; '+
19154:                  const AScriptEngine:str; ALocateFileService:ILocateFileService) :Str;
19155:  Func FindComponentWebModuleContext( AComponent : TComponent) : TWebModuleContext;
19156:  Func GetTagID( const TagString :Str) : TTag;
19157:  Func ContentFromStream(AStream:TStream;
        AStripParamQuotes:Boolean;AHandleTag:THandleTagProc;AHandledTag:THandledTagProc):str;
19158:  Func ContentFromString(const AValue:str;
        AStripParamQuotes:Bool;AHandleTag:THandleTagProc;AHandledTag:THandledTagProc):str;
19159:  RIRegister_synacrypt_Routines(Exec);
19160:  Func TestDes:Bool;
19161:  {:Call internal test of all 3DES encryptions. Returns @true if all is OK.}
19162:  Func Test3Des:Bool;
19163:  {:Call internal test of all AES encryptions. Returns @true if all is OK.}
19164:  Func TestAes:Bool;
19165:  Proc LogMessage( const Fmt :Str; const Params : array of const);
19166:  Func UnixPathToDosPath2( const Path :Str) :Str;
19167:  Func DosPathToUnixPath2( const Path :Str) :Str;
19168:  Proc InitISAPIApplicationList;
19169:  Proc DoneISAPIAPplicationList;
19170:
19171:  RIRegister_xmldom_Routines(Exec);
19172:  Func IsPrefixed( const AName : DOMString) :Bool;
19173:  Func IsPrefixedW( const AName : DOMStringW) :Bool;
19174:  Func ExtractLocalName( const AName : DOMString) : DOMString;
19175:  Func ExtractLocalNameW( const AName : DOMStringW) : DOMStringW;
19176:  Func ExtractPrefixW( const AName : DOMStringW) : DOMStringW;
19177:  Func MakeNodeNameW( const Prefix, LocalName : DOMStringW) : DOMStringW;
19178:  Func ExtractPrefix( const AName : DOMString) : DOMString;
19179:  Func MakeNodeName( const Prefix, LocalName : DOMString) : DOMString;
19180:  Func SameNamespace(const Node:IDOMNode;const namespaceURI:WideString):Boolean;;
19181:  Func SameNamespace2( const URI1, URI2 : WideString) :Bool;;
19182:  Func NodeMatches(const Node:IDOMNode;const TagName,NamespaceURI:DOMString):Bool;;
19183:  Func GetDOMNodeEx( const Node : IDOMNode) : IDOMNodeEx;
19184:  Proc RegisterDOMVendor( const Vendor : TDOMVendor);
19185:  Proc UnRegisterDOMVendor( const Vendor : TDOMVendor);
19186:  Func GetDOMVendor( VendorDesc :Str) : TDOMVendor;
19187:  Func GetDOM( const VendorDesc :Str) : IDOMImplementation;
19188:  Proc DOMVendorNotSupported( const PropOrMethod, VendorName :Str);
19189:  Func IsValidLocale( Locale : LCID; dwFlags : DWORD) : BOOL;
19190:  Func ConvertDefaultLocale( Locale : LCID) : LCID;
19191:  Func GetThreadLocale : LCID;
19192:  Func SetThreadLocale( Locale : LCID) : BOOL;
19193:  Func GetSystemDefaultLangID : LANGID;
19194:  Func GetUserDefaultLangID : LANGID;
19195:  Func GetSystemDefaultLCID : LCID;
19196:  Func GetUserDefaultLCID : LCID;
19197:  Func AbsInt( const B : Int) : Int;
19198:  Func AbsFloat( const B : double) : extended;
19199:
19200:  Proc ReconcileDeltas(const cdsArray:array of TClientDataset;vDeltaArray:OleVariant);
19201:  Proc CDSApplyUpdates(ADatabase:TDatabase;var vDeltaArray:OleVariant;const
        vProviderArray:OleVariant;Local:Bool);
19202:  Function(@CheckMemory, 'Proc CheckMemory;;
19203:  Function(@GetMemoryInfo, 'Func getMemoryInfo;;
19204:  Function(@GetMemoryInfo, 'Func getMemInf;;
19205:  Proc RaiseLastWin32_2( const Text :Str);;
19206:
19207:  RIRegister_Gameboard_Routines(Exec);
19208:  Func Opponent( Player : TPlayer) : TPlayer;
19209:  Proc InitZobritsNumbers( var ZobristNumbers, Count : Int);
19210:  Proc SaveStringToStream( const Str :Str; Stream : TStream);
19211:  Func LoadStringFromStream( Stream : TStream) :Str;
19212:  Func WaitForSyncObject(SyncObject:THandle;Timeout:Card;BlockInput:Boolean):Card);
19213:  Func ProcessMessage :Bool;
19214:  Proc ProcessMessages( Timeout : DWORD);  //without application!
19215:
19216:  RIRegister_PersistSettings_Routines(Exec);
19217:  Proc SetStorageHandler( AFunc : TStorageHandlerFunction);;
19218:  Proc SetStorageHandler1( AMethod : TStorageHandlerMethod);;
19219:  Func GetStorage : TPersistStorage;
19220:  Proc SaveComponents( Root : TComponent; Storage : TPersistStorage);
19221:  Proc LoadComponents( Root : TComponent; Storage : TPersistStorage);
19222:  Proc AutoSave( Root : TComponent);
19223:  Proc AutoLoad( Root : TComponent);
19224:
19225:  Proc FloatToDecimalE(var Result:TFloatRec;const Value:extended;ValueType:TFloatValue;Precision,
        Decimals:Int);
19226:  Func FloatToTextE(BufferArg:PChar; const Value:extended; ValueType: FloatValue;Format:TFloatFormat;
        Precision, Digits: Int): Int;
```

```
19227:  Proc FloatToDecimalE(var Result:TFloatRec;const Value:extended;ValueType:TFloatValue;Precision,
        Decimals:Int);;
19228:  Func FloatToTextE(BufferArg:PChar;const
        Value:extended;ValueType:TFloatValue;Format:TFloatFormat;Precision,Digits:Int):Int;
19229:  Func GetSystemDefaultLCID : LCID;
19230:  Func GetUserDefaultLCID : LCID;
19231:  Func CreateMutex2(lpMutexAttributes:TObject;bInitialOwner:BOOL;lpName:PChar) : THandle;
19232:  Func CreateSemaphore2(lpSemaphoreAttributes:TObject;lInitialCount,
        lMaximumCount:Longint;lpName:PChar):THandle;
19233:  Func GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD) : BOOL;
19234:  Createmutex2 fhand:= OpenFileHandle(exepath+'maxbox4.exe')
19235:  Func getMatchString(arex, atext:Str):Str;
19236:  Func getLastInput: DWord;
19237:  Proc GetKLList(List: TStrings);;   //Keyboardlist2
19238:  Proc EnableCTRLALTDEL(YesNo :Bool);
19239:  Func LocalIP:Str;
19240:  Func IPAddrToName(IPAddr:Str):Str;
19241:  Func GetIPFromHost(const HostName:Str):Str;
19242:  function getIPFromHostName(hostName: string): string;
19243:  Func FindComputers(Computers : TStringList): DWORD;
19244:  Func GetWin32TypeLibList(var Lines: TStringList):Bool;
19245:  Func RecurseWin32(const R: TRegistry; const ThePath:Str;
19246:                          const TheKey:Str):Str;
19247:  Func RunAsAdmin(hWnd: HWND; filename:Str; Parameters:Str):Bool;
19248:  Ex.:     RunAsAdmin2(hinstance,'cmd.exe',+
19249:                          'Set OPENSSL_CONF='+OpenSSLPathF+'openssl.cfg');
19250:  writeln(getDosOutput('openssl.exe version',OpenSSLPath));
19251:  Proc SaveGraphicToStream(Graphic: TGraphic; Stream: TStream);
19252:  Func LoadGraphicFromStream(Stream: TStream): TGraphic;
19253:  Proc CopyStreamToFile(S: TStream; F: THandle);
19254:  Func BigPow(aone, atwo: Int):Str;
19255:  Proc GetPelsPerMeter(CONST  Bitmap: TBitmap;VAR xPelsPerMeter,yPelsPerMeter: Int);
19256:  Proc RainbowColor(CONST fraction:  Double; VAR R,G,B:  BYTE);
19257:  RIRegister_JclPCRE_Routines(S: TPSExec);
19258:  Proc InitializeLocaleSupport;
19259:  Proc TerminateLocaleSupport;
19260:  Func StrReplaceRegEx(const Subject, Pattern:Ansistr;Args:array of const):Ansistr;
19261:
19262: Proc SIRegister_VariantRtn(CL: TPSPascalCompiler);
19263: begin
19264: TProcReadElementValue','procedure(Value:Variant; IndexValue:Intarray; const HighBoundInd:Int; Var
       Continue:bool);
19265: TProcWriteElementValue','procedure(OldValue:Variant;IndexValue:Intarray;Var NewValue:Variant; Var
       Continue:bool);
19266:  //TProcReadElementValue=procedure(Value:Variant;IndexValue:array of Int; const HighBoundInd:Int;Var
        Continue:bool);
19267:  //TProcWriteElementValue=procedure(OldValue:Variant;IndexValue:array of Int;Var NewValue:Variant;Var
        Continue:bool); //
19268:  Func SafeVarArrayCreate(const Bounds:array of Int;VarType, DimCount: Int): Variant;
19269:  Func VarArrayGet2(constA:Variant;const Indice:array of Int;const HighBound:Int):Variant);
19270:  Proc VarArrayPut2(var A:Variant;const Value:Variant;const Indices:array of Int;const HighBound:Int);
19271:  Func CycleReadArray( vArray : Variant; CallBackProc : TProcReadElementValue) :Bool;
19272:  Func CycleWriteArray(var vArray:Variant;CallBackProc: TProcWriteElementValue):Bool;
19273:  Func CompareVarArray1( vArray1, vArray2 : Variant) :Bool;
19274:  Func EasyCompareVarArray1(vArray1,vArray2:Variant;HighBound:Int):Bool;
19275: end;
19276:
19277: Proc SIRegister_StdFuncs(CL: TPSPascalCompiler);
19278: begin
19279:   ClassN(CL.FindClass('TOBJECT'),'EParserError;
19280:   //TypeS('TCharSet', 'set of Char;
19281:  Func ConvertFromBase(sNum:Str; iBase: Int; cDigits:Str): Int;
19282:  Func ConvertToBase( iNum, iBase : Int; cDigits :Str) :Str;
19283:  Func EnsureSentenceTerminates( Sentence :Str; Terminator : Char) :Str;
19284:  Func FindTokenStartingAt(st:Str;var i:Int;TokenChars:TCharSet;TokenCharsInToken:Bool):Str;
19285:  Func GetDirectoryOfFile( FileName :Str) :Str;
19286:  Func GetDirOfFile( FileName :Str) :Str;
19287:  Func GetTempFile( FilePrefix :Str) :Str;
19288:  Func Icon2Bitmap( Icon : HICON) : HBITMAP;
19289:  Func Maxfib( n1, n2 : Int) : Int;
19290:  Func MaxD( n1, n2 : Double) : Double;
19291:  Func Minfib( n1, n2 : Int) : Int;
19292:  Func MinD( n1, n2 : Double) : Double;
19293:  Func RandomStringfib( iLength : Int) :Str;
19294:  Func RandomIntfib( iLow, iHigh : Int) : Int;
19295:  Func Soundexfib( st :Str) :Str;
19296:  Func StripStringfib( st :Str; CharsToStrip :Str) :Str;
19297:  Func ClosestWeekday( const d : TDateTime) : TDateTime;
19298:  Func Yearfib( d : TDateTime) : Int;
19299:  Func Monthfib( d : TDateTime) : Int;
19300:  Func DayOfYearfib( d : TDateTime) : Int;
19301:  Func DayOfMonth( d : TDateTime) : Int;
19302:  Func VarCoalesce( V1, V2 : Variant) : Variant;
19303:  Func VarEqual( V1, V2 : Variant) :Bool;
19304:  Proc WeekOfYearfib( d : TDateTime; var Year, Week : Int);
19305:  Func Degree10( Degree : Int) : double;
19306:  Func CompToStr( Value : comp) :Str;
19307:  Func StrToComp( const Value :Str) : comp;
19308:  Func CompDiv( A, B : comp) : comp;
```

```
19309:  Func CompMod( A, B : comp) : comp;
19310:  // TypeS('PComp', '^Comp // will not work;
19311: end;
19312:
19313: Proc SIRegister_RegUtils(CL: TPSPascalCompiler);
19314: begin
19315:  Proc DefWriteToRegistry(const OtherKeys,ParamNams:array of string;const Values:array of Variant);
19316:  Proc WriteToRegistry(aRootKey:HKEY;const OtherKeys,ParamNames:array of string;const Values:array of
       Variant);
19317:  Func ReadFromRegistry(aRootKey:HKEY;const OtherKeys,ParamNames: array of string): Variant;
19318:  Func DefReadFromRegistry( const OtherKeys, ParamNames : array of string) : Variant;
19319:  Func AllSubKey( aRootKey : HKEY; const ForPath : array of string) : Variant;
19320:  Func DefAllSubKey( const ForPath : array of string) : Variant;
19321:  Func SaveRegKey( const FileName :Str; const ForKey : array of string) :Bool;
19322:  Func LoadRegKey( const FileName :Str; const ForKey : array of string) :Bool;
19323:  Func AltSaveRegKey(const FileName:str;const ForKey: array of string) :Bool;
19324:  Func AltLoadRegKey(const FileName:str;const ForKey: array of string) :Bool;
19325:  Func GetKeyForParValue( const aRootKey, ParName, ParValue :Str) :Str;
19326: end;
19327:
19328: Func GetEulerPhi( n : int64) : int64;
19329:  Func isprime(f: int64):Bool;;
19330:  Func  DispositionFrom(const SQLText:str):TPoint;
19331:  Proc AllTables(const SQLText:str;FTables:Tstrings);
19332:  Func  TableByAlias(const SQLText,Alias:str):str;
19333:  Func  FullFieldName(const SQLText,FieldName:str):str;
19334:  Func  AddToWhereClause(const SQLText,NewClause:str):str;
19335:  Func  GetWhereClause(SQLText:str;N:Int;var StartPos,EndPos:Int ):str;
19336:  Func  WhereCount(SQLText:str):Int;
19337:  Func  GetOrderInfo(SQLText:str):variant;
19338:  Func  OrderStringTxt(SQLText:str; var StartPos,EndPos:Int ):str;
19339:  Func  PrepareConstraint(Src:Tstrings):str;
19340:  Proc DeleteEmptyStr(Src:Tstrings);
19341:  Func  NormalizeSQLText(const SQL:Str;MacroChar:Char):Str;
19342:  Func  CountSelect(const SrcSQL:str):str;
19343:  Func  GetModifyTable(const SQLText:str;AlreadyNormal:boolean):str;
19344: Func GetCharFromVKey(vkey: Word):Str;
19345: Func Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile:Str):Bool;
19346: Func IsObjectActive(ClassName:Str):Bool;
19347: Func GetActiveObject(ClassID:TGUID; anil:TObject; aUnknown:IUnknown):HRESULT;;
19348: Func RegisterOCX(FileName:Str):Bool;
19349: Func UnRegisterOCX(FileName:Str):Bool;
19350: Func RegisterServer2(const aDllFileName:Str; aRegister:Bool):Bool;
19351: Proc mIRCDDE(Service, Topic, Cmd:Str);
19352:             //mIRCDDE('mIRC', 'COMMAND', '/say Hallo von SwissDelphiCenter.ch;
19353: Func OpenIE(aURL:Str):Bool;
19354: Func XRTLIsInMainThread:Bool;
19355: Func IsInMainThread:Bool;
19356:
19357: Proc SIRegister_SpectraLibrary(CL: TPSPascalCompiler);
19358: begin
19359:  TypeS('Nanometers', 'Double;
19360:  ConstantN('WavelengthMinimum','LongInt').SetInt( 380);
19361:  ConstantN('WavelengthMaximum','LongInt').SetInt( 780);
19362:  Proc WavelengthToRGB( const Wavelength : Nanometers; var R, G, B : BYTE);
19363: end;
19364: Proc SIRegister_DrawFigures(CL: TPSPascalCompiler);
19365: begin
19366:  Proc DrawCube( const PantoGraph : TPantoGraph; const color : TColor);
19367:  Proc DrawSphere(const PantoGraph:TPantoGraph;const LatitudeColor,LongitudeColor:TColor; const
       LatitudeCircles,LongitudeSemicircles,PointsInCircle:WORD);
19368:  Proc DrawSurface( const PantoGraph : TPantoGraph);
19369:  Proc DrawFootballField(const PantoGraph:TPantoGraph;const ColorField,ColorLetters,ColorGoals:TColor);
19370: end;
19371: Proc SIRegister_synadbg(CL: TPSPascalCompiler);
19372: begin
19373:   SIRegister_TSynaDebug(CL);
19374:  Proc AppendToLog( const value : Ansistr);
19375: end;
19376:
19377: Proc SIRegister_XmlRpcCommon(CL: TPSPascalCompiler);
19378: begin
19379:   SIRegister_TRC4(CL);
19380:   TypeS(TRPCDataType', '( rpNone, rpString, rpInt, rpBoolean, rpDoub'
19381:    +le, rpDate, rpBase64, rpStruct, rpArray, rpName, rpError );
19382:  Func GetTempDirRPC :Str;
19383:  Func FileIsExpired( const FileName :Str; Elapsed : Int) :Bool;
19384:  Func EncodeEntities( const Data :Str) :Str;
19385:  Func DecodeEntities( const Data :Str) :Str;
19386:  Func ReplaceRPC(const Data:str;const Find:str;const Replace:str):Str;
19387:  Func InStr( Start : Int; const Data :Str; const Find :Str) : Int;
19388:  Func Mid( const Data :Str; Start : Int) :Str;
19389:  Func DateTimeToISO( ConvertDate : TDateTime) :Str;
19390:  Func IsoToDateTime( const ISOStringDate :Str) : TDateTime;
19391:  Func ParseStringRPC(const SearchString:str; Delimiter:Char; Substrings: TStrings; const
       AllowEmptyStrings:Bool; ClearBeforeParse:Boolean):Int;
19392:  Func ParseStream(
       SearchStream:TStream;Delimiter:Char;ubstrings:TStrings;AllowEmptyStrings:Bool;ClearBeforeParse:Bool): Int;
19393:  Func FixEmptyString( const Value :Str) :Str;
```

```
19394:  Func URLEncodeRPC( const Value :Str) :Str;
19395:  Func StreamToStringRPC( Stream : TStream) :Str;
19396:  Proc StringToStream( const Text :Str; Stream : TStream);
19397:  Func StreamToVariant( Stream : TStream) : OleVariant;
19398:  Proc VariantToStream( V : OleVariant; Stream : TStream);
19399:  Func Hash128AsHex( const Hash128Value : T4x4LongWordRecord) :Str;
19400: ConstantN(ValidURLChars','String').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$-_@.&+-!''*"(),;/#?:;
19401: end;
19402:
19403: Proc SIRegister_synafpc(CL: TPSPascalCompiler);
19404: begin
19405:  Func LoadLibraryfpc( ModuleName : PChar) : TLibHandle;
19406:  Func FreeLibraryfpc( Module : TLibHandle) : LongBool;
19407:  Func GetProcAddressfpc( Module : TLibHandle; Proc : PChar) : Pointer;
19408:  Func GetModuleFileNamefpc(Module:TLibHandle;Buffer:PChar; BufLen:Int):Int;
19409:   TypeS('TLibHandle', 'Int;
19410:   TypeS('TLibHandle2', 'HModule;
19411:   TypeS(LongWordfpc', DWord;
19412: Proc Sleepfpc( milliseconds :Card);
19413: end;
19414:
19415: Proc SIRegister_Spring_Utilsmx(CL: TPSPascalCompiler);
19416: begin
19417:   TypeS('TOSPlatformType', '( ptUnknown, ptWin3x, ptWin9x, ptWinNT );
19418:   SIRegister_TOperatingSystem(CL);
19419:   SIRegister_TEnvironmentClass(CL);
19420:   TypeS('Environment', 'TEnvironment;
19421:  Func ApplicationPath :Str;
19422:  Func ApplicationVersion : TVersion;
19423:  Func ApplicationVersionString :Str;
19424:  Func GetLastErrorMessage :Str;
19425:  function getLastSysErrorMessage: string;
19426:  Func CreateCallback( obj : TObject; methodAddress : Pointer) : TCallbackFunc;
19427:  Func ConvertFileTimeToDateTime(const fileTime:TFileTime;useLocalTimeZone:Bool):TDateTime;;
19428:  Func ConvertDateTimeToFileTime(const datetime:TDateTime;useLocalTimeZone:Bool):TFileTime;
19429:  {Proc Synchronize( threadProc : TThreadProcedure);
19430:  Proc Queue( threadProc : TThreadProcedure); }
19431:  Func TryGetPropInfo(instanc:TObject;const propertyName:str;out propInfo:PPropInfo):Bool;
19432:  Func IsCtrlPressed :Bool;
19433:  Func IsShiftPressed :Bool;
19434:  Func IsAltPressed :Bool;
19435:  Proc CheckFileExists( const fileName :Str);
19436:  Proc CheckDirectoryExists( const directory :Str);
19437:  ConstantN('COneKB','Int64').SetInt64( 1024);
19438:  COneMB',Int64').SetInt64( 1048576);
19439:  COneGB',Int64').SetInt64( 1073741824);
19440:  COneTB',Int64').SetInt64( 1099511627776);
19441:  Func TryConvertStrToDateTime(const s,format:str;out value:TDateTime):Boolean;;
19442:  Func ConvertStrToDateTime(const s, format:Str): TDateTime;;
19443: end;
19444:
19445: Proc SIRegister_rxOle2Auto(CL: TPSPascalCompiler);
19446: begin
19447:  ConstantN('MaxDispArgs','LongInt').SetInt( 64);
19448:  ConstantN('MaxDispArgs','LongInt').SetInt( 32);
19449:   ClassN(CL.FindClass('TOBJECT'),'EPropReadOnly;
19450:   ClassN(CL.FindClass('TOBJECT'),'EPropWriteOnly;
19451:   SIRegister_TOleController(CL);
19452:  Proc InitOLE2;
19453:  Proc DoneOLE;
19454:  Func OleInitialized :Bool;
19455:  Func MakeLangID( PrimaryLangID, SubLangID : Word) : Word;
19456:  Func MakeLCID( LangID : Word) : TLCID;
19457:  Func CreateLCID( PrimaryLangID, SubLangID : Word) : TLCID;
19458:  Func ExtractLangID( LCID : TLCID) : Word;
19459:  Func ExtractSubLangID( LCID : TLCID) : Word;
19460: Func CLSIDFromProgID(pszProgID: widestring; out clsid: TGUID): Longint'); //stdcall;
19461: Func ProgIDFromCLSID(const clsid: TGUID; out pszProgID: widestring): Longint'); //stdcall;
19462: Func StringFromCLSID(const clsid: TGUID; out psz: widestring): longint'); //stdcall;
19463: //{$EXTERNALSYM CLSIDFromString}
19464:  Func CLSIDFromString(psz: widestring; out clsid: TGUID): longint'); //stdcall;
19465:  Func StringFromGUID2(const guid: TGUID;psz: widestring;cbMax:Integer):Integer');//stdcall;
19466:  Func ProgIDExists(const ProgID:WideString):Boolean');
19467:  Proc CoUninitialize');
19468:  Proc OleUninitialize');
19469: end;
19470:
19471: Proc SIRegister_ulogifit(CL: TPSPascalCompiler);
19472: begin
19473: Proc LogiFit(X,Y:TVector;Lb,
       Ub:Int;ConsTerm:Boolean;General:Bool;MaxIter:Int;Tol:Float;B:TVector;V:TMatrix);
19474: Proc WLogiFit(X,Y,S:TVector;Lb,Ub:Int;ConsTerm Bool General:Bool;MaxIter
       Int;Tol:Float;B:TVector;V:TMatrix);
19475: Func LogiFit_Func( X : Float; B : TVector) : Float;
19476: end;
19477:
19478: Proc FormattedTextOut(TargetCanvas: TCanvas; const Rect : TRect; const Text :Str; Selected :Bool; Columns
       : TProposalColumns; Images : TImageList);
19479:  Func FormattedTextWidth(TargetCanvas:TCanvas;const
       Text:str;Columns:TProposalColumns;Images:TImageList):Int;
```

```
19480:  Func PrettyTextToFormattedString(const APrettyText:Str; AlternateBoldStyle:Boolean):str;
19481:
19482:  Proc SIRegister_ulinfit(CL: TPSPascalCompiler);
19483: begin
19484:  Proc LinFit( X, Y : TVector; Lb, Ub : Int; B : TVector; V : TMatrix);
19485:  Proc WLinFit( X, Y, S : TVector; Lb, Ub : Int; B : TVector; V : TMatrix);
19486:  Proc SVDLinFit(X,Y:TVector; Lb, Ub:Int; SVDTol : Float; B: TVector; V : TMatrix);
19487:  Proc WSVDLinFit(X,Y,S: TVector; Lb,Ub:Int;SVDTol: Float; B: TVector; V : TMatrix);
19488:   Proc SVDFit(X:TMatrix;Y:TVector;Lb,Ub Nvar:Int;ConsTerm:Bool; SVDTol: Float;B: TVector; V: TMatrix);
19489:  Proc WSVDFit(X: TMatrix;Y,S:TVector;Lb Ub,Nvar:Int;ConsTerm:Boolean;SVDTol:Float; B: TVector;V : TMatrix);
19490: end;
19491:
19492: Proc SIRegister_MaxUtils(CL: TPSPascalCompiler);
19493: begin
19494:   TypeS('MaxCharSet', 'set of Char;
19495:  Func GetMachineNamemax :Str;
19496:  Func GetModuleNamemax( HModule : THandle) :Str;
19497:  Func TrimChars( const S :Str; Chars : MaxCharSet) :Str;
19498:  Func TickCountToDateTime( Ticks :Card) : TDateTime;
19499:  Proc OutputDebugStringmax( const S :Str);
19500:  Proc OutputDebugFormat( const FmtStr :Str; Args : array of const);
19501:  Func IsAppRunningInDelphi :Bool;
19502: Proc ParseFields(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strigs:TStrings;Decode: Bool;
19503:  Func HTTPDecodemax( const AStr :Str) :Str;
19504:  Func HTTPEncodemax( const AStr :Str) :Str;
19505:  Func FormatDate( const DateString :Str) :Str;
19506:  Func FormatListMasterDate(const DateStr,FormatDefStr:str; Len:Int):str;
19507:  Func InvertCase( const S :Str) :Str;
19508:  Func CommentLinesWithSlashes( const S :Str) :Str;
19509:  Func UncommentLinesWithSlashes( const S :Str) :Str;
19510:  Func StripChars( const S :Str; Strip : CharSet) :Str;
19511:  Func TrimChars( const S :Str; Chars : CharSet) :Str;
19512:  Func TrimLeftChars( const S :Str; Chars : CharSet) :Str;
19513:  Func TrimRightChars( const S :Str; Chars : CharSet) :Str;
19514:  Func ContainsChars( const S :Str; Strip : CharSet) :Bool;
19515:  Func DequotedStrmax( const S :Str; AQuoteChar : Char) :Str;
19516:  Proc LeftPadStr( var S :Str; toLength : Int; withChar : Char);
19517:  Proc RightPadStr( var S :Str; toLength : Int; withChar : Char);
19518:  Func RemoveChars( S :Str; Chars : CharSet) :Str;
19519:  Func FilterChars( S :Str; Chars : CharSet) :Str;
19520:  Func RemoveNonNumericChars( S :Str) :Str;
19521:  Func RemoveNonAlphanumChars( S :Str) :Str;
19522:  Func RemoveNonAlphaChars( S :Str) :Str;
19523:  Func HasAlphaChars( S :Str) :Bool;
19524:  Func ReplaceChars( S :Str; Chars : CharSet; ReplaceWith : Char) :Str;
19525:  Func DomainOfEMail( const EMailAddress :Str) :Str;
19526:  Func IPToHexIP( const IP :Str) :Str;
19527:  Proc CmdLineToStrings( S : Ansistr; const List : TStrings);
19528:  BASE2','String').SetString( '01;
19529:  ConstantN('BASE10','String').SetString(0123456789;
19530:  ConstantN('BASE16','String').SetString(0123456789ABCDEF;
19531:  ConstantN('BASE36','String').SetString(0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ;
19532:  ConstantN('BASE62','String').SetString(0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;
19533:  Func BaseConvert( Number, FromDigits, ToDigits :Str) :Str;
19534: Func ValidXmlName( AName : PChar; ASize : Int) :Bool;;
19535: Func ValidXmlName1( const AName :Str) :Bool;;
19536:  Func EncodeXmlAttrValue( const AStr : Ansistr) : Ansistr;;
19537:  Proc EncodeXmlAttrValue3(ABuff:PChar;ABuffSize:Int;var AStr:AnsiStr;var ALen,AOffset:Int);
19538:  Proc EncodeXmlAttrValue4(const ASource:Str;var ADest:str;var ALen,AOffset:Int);
19539:  Func EncodeXmlString5( const AStr :Str) :Str;;
19540:  Proc EncodeXmlString6(ABuff:PChar;ABuffSize:Int;var AStr:Ansistr;var ALen,AOffset:Int;
19541:  Proc EncodeXmlComment(const ASource:Ansistr;var ADest:Ansistr;var ALen,AOffset:Int);
19542:  Func HasEncoding( const AStr : Ansistr) :Bool;
19543:  Func DecodeXmlAttrValue( const AStr :Str) :Str;
19544:  Proc ReallocateString(var AString:Ansistr; var ALen : Int; AReqLen : Int);
19545:  Proc AttrFillXMLString(AAttr:IAttribute;var aString:Ansistr;var aOffset,aLen : Int);
19546:  Proc FillXMLString(ANode:INode;var AStr:Str;var AOffset,ALen:Int;ASiblig:INode;ALevel:Int)
19547:  Func NodeToXML( ANode : INode) :Str;
19548:  Proc XMLSaveToFile( ANode : INode; const AFileName :Str);
19549:  Func XMLLoadFromFile( const AFileName :Str) : INode;
19550:  Func Hashmax( const ASource : Ansistr) :Card;
19551:  ConstantN('GXMLIndentSpaces','Int').SetInt( 2);
19552:  //ConstantN('GXMLMultiLineAttributes','Boolean')BoolToStr( True);
19553:  GXMLMultiLineAttributeThreshold','Int').SetInt( 7);
19554: end;
19555:
19556: Proc SIRegister_MaxDOMDictionary(CL: TPSPascalCompiler);
19557: begin
19558:   SIRegister_IDictionary(CL);
19559:   SIRegister_TDictionary(CL);
19560:  Func HashFast( const AKey :Str) :Card;
19561:  Func HashCarlos( const AKey :Str) :Card;
19562:  Func BorlandHashOf( const AKey :Str) :Card;
19563:  Func HashSumOfChars( const AKey :Str) :Card;
19564: end;
19565:
19566: Proc SIRegister_MaxDOM(CL: TPSPascalCompiler);
19567: begin
19568:   TypeS('TNodeType', '( ntElement, ntText, ntCDATA, ntComment );
```

```
19569:   Interface(CL.FindInterface('IUNKNOWN'),INode, 'INode;
19570:   Interface(CL.FindInterface('IUNKNOWN'),IAttribute, 'IAttribute;
19571:   Interface(CL.FindInterface('IUNKNOWN'),IAttributeCollection, 'IAttributeCollection;
19572:   Interface(CL.FindInterface('IUNKNOWN'),INodeCollection, 'INodeCollection;
19573:   SIRegister_INode(CL);
19574:   SIRegister_IAttribute(CL);
19575:   SIRegister_IAttributeCollection(CL);
19576:   SIRegister_INodeCollection(CL);
19577:  Func NodeCreate( const ANodeName :Str; ANodeType : TNodeType) : INode;
19578:   SIRegister_TAttribute(CL);
19579:   TypeS(TNodes', 'array of INode;
19580:   TypeS(TAttributes2', 'array of IAttribute;
19581:   TypeS(THashedAttributes', 'record Attributes : TAttributes2; AttrCount '
19582:    +': Int; AttrCapacity : Int; end;
19583:   TypeS(TAttrHashTable', 'array of THashedAttributes;
19584:   SIRegister_TNode(CL);
19585:   //TypeS('TNodeClass', 'class of TNode;
19586:   //TypeS('TAttributeClass', 'class of TAttribute;
19587:  Func PointerToStr( P : ___Pointer) :Str;
19588:  Func StrToPointer( const S :Str) : ___Pointer;
19589:  Func INodeToStr( ANode : INode) :Str;
19590:  Func StrToINode( const S :Str) : INode;
19591:  Func CompareByNodeName( N1, N2 : INode) : Int;
19592:  Func CompareByNameAttr( N1, N2 : INode) : Int;
19593: end;
19594:
19595: Proc SIRegister_cASN1(CL: TPSPascalCompiler);
19596: begin
19597:   ClassN(CL.FindClass('TOBJECT'),'EASN1;
19598:  'ASN1_ID_END_OF_CONTENT','LongWord').SetUInt( $00);
19599:  ConstantN('ASN1_ID_BOOLEAN','LongWord').SetUInt( $01);
19600:  'ASN1_ID_Int','LongWord').SetUInt( $02);
19601:  'ASN1_ID_BIT_STRING','LongWord').SetUInt( $03);
19602:  'ASN1_ID_OCTET_STRING','LongWord').SetUInt( $04);
19603:  'ASN1_ID_NULL','LongWord').SetUInt( $05);
19604:  'ASN1_ID_OBJECT_IDENTIFIER','LongWord').SetUInt( $06);
19605:  'ASN1_ID_OBJECT_DESCRIPTOR','LongWord').SetUInt( $07);
19606:  'ASN1_ID_EXTERNAL','LongWord').SetUInt( $08);
19607:  'ASN1_ID_REAL','LongWord').SetUInt( $09);
19608:  'ASN1_ID_ENUMERATED','LongWord').SetUInt( $0A);
19609:  'ASN1_ID_EMBEDDED_PDV','LongWord').SetUInt( $0B);
19610:  'ASN1_ID_UTF8STRING','LongWord').SetUInt( $0C);
19611:  'ASN1_ID_RELATIVE_OID','LongWord').SetUInt( $0D);
19612:  'ASN1_ID_NUMERICSTRING','LongWord').SetUInt( $12);
19613:  'ASN1_ID_PRINTABLESTRING','LongWord').SetUInt( $13);
19614:  'ASN1_ID_T61STRING','LongWord').SetUInt( $14);
19615:  'ASN1_ID_VIDEOTEXSTRING','LongWord').SetUInt( $15);
19616:  'ASN1_ID_IA5STRING','LongWord').SetUInt( $16);
19617:  'ASN1_ID_UTCTIME','LongWord').SetUInt( $17);
19618:  'ASN1_ID_GENERALIZEDTIME','LongWord').SetUInt( $18);
19619:  'ASN1_ID_GRAPHICSTRING','LongWord').SetUInt( $19);
19620:  'ASN1_ID_VISIBLESTRING','LongWord').SetUInt( $1A);
19621:  'ASN1_ID_GENERALSTRING','LongWord').SetUInt( $1B);
19622:  'ASN1_ID_UNIVERSALSTRING','LongWord').SetUInt( $1C);
19623:  'ASN1_ID_CHARACTERSTRING','LongWord').SetUInt( $1D);
19624:  'ASN1_ID_BMPSTRING','LongWord').SetUInt( $1E);
19625:  'ASN1_ID_SEQUENCE','LongWord').SetUInt( $30);
19626:  'ASN1_ID_SET','LongWord').SetUInt( $31);
19627:  'ASN1_ID_CONSTRUCTED','LongWord').SetUInt( $20);
19628:  'ASN1_ID_APPLICATION','LongWord').SetUInt( $40);
19629:  'ASN1_ID_CONTEXT_SPECIFIC','LongWord').SetUInt( $80);
19630:  'ASN1_ID_PRIVATE','LongWord').SetUInt( $C0);
19631:   TASN1ObjectIdentifier', 'array of Int;
19632:   TASN1ParseProc', 'procedure(const TypeID:Byte; const DataBuf:Str; const DataSize: Int; const ObjectIdx:
       Int; const CallerData: Int);;
19633: {TASN1ParseProc =
19634:    Proc (const TypeID: Byte; const DataBuf; const DataSize: Int;
19635:                const ObjectIdx: Int; const CallerData: Int); }
19636:
19637:  Proc ASN1OIDInit( var A : TASN1ObjectIdentifier; const B : array of Int);
19638:  Func ASN1OIDToStr( const A : TASN1ObjectIdentifier) : Ansistr;
19639:  Func ASN1OIDEqual(const A:TASN1ObjectIdentifier; const B : array of Int) :Bool;
19640:  Func ASN1EncodeLength( const Len : Int) : Ansistr;
19641:  Func ASN1EncodeObj( const TypeID : Byte; const Data : Ansistr) : Ansistr;
19642:  Func ANS1EncodeEndOfContent : Ansistr;
19643:  Func ASN1EncodeNull : Ansistr;
19644:  Func ASN1EncodeBoolean( const A :Bool) : Ansistr;
19645:  Func ASN1EncodeDataInt8( const A : ShortInt) : Ansistr;
19646:  Func ASN1EncodeDataInt16( const A : SmallInt) : Ansistr;
19647:  Func ASN1EncodeDataInt24( const A : LongInt) : Ansistr;
19648:  Func ASN1EncodeDataInt32( const A : LongInt) : Ansistr;
19649:  Func ASN1EncodeDataInt64( const A : Int64) : Ansistr;
19650:  Func ASN1EncodeInt8( const A : ShortInt) : Ansistr;
19651:  Func ASN1EncodeInt16( const A : SmallInt) : Ansistr;
19652:  Func ASN1EncodeInt24( const A : LongInt) : Ansistr;
19653:  Func ASN1EncodeInt32( const A : LongInt) : Ansistr;
19654:  Func ASN1EncodeInt64( const A : Int64) : Ansistr;
19655:  Func ASN1EncodeIntBuf( const A :Str; const Size : Int) : Ansistr;
19656:  Func ASN1EncodeIntBufStr( const A : Ansistr) : Ansistr;
```

```
19657:  Func ASN1EncodeEnumerated( const A : Int64) : Ansistr;
19658:  Func ASN1EncodeBitString(const A: Ansistr; const UnusedBits : Byte): Ansistr;
19659:  Func ASN1EncodeOctetString( const A : Ansistr) : Ansistr;
19660:  Func ASN1EncodeInt32AsOctetString( const A : LongInt) : Ansistr;
19661:  Func ASN1EncodeUTF8String( const A : Ansistr) : Ansistr;
19662:  Func ASN1EncodeIA5String( const A : Ansistr) : Ansistr;
19663:  Func ASN1EncodeVisibleString( const A : Ansistr) : Ansistr;
19664:  Func ASN1EncodeNumericString( const A : Ansistr) : Ansistr;
19665:  Func ASN1EncodePrintableString( const A : Ansistr) : Ansistr;
19666:  Func ASN1EncodeTeletexString( const A : Ansistr) : Ansistr;
19667:  Func ASN1EncodeUniversalString( const A : WideString) : Ansistr;
19668:  Func ASN1EncodeBMPString( const A : WideString) : Ansistr;
19669:  Func ASN1EncodeUTCTime( const A : TDateTime) : Ansistr;
19670:  Func ASN1EncodeGeneralizedTime( const A : TDateTime) : Ansistr;
19671:  Func ASN1EncodeOID( const OID : array of Int) : Ansistr;
19672:  Func ASN1EncodeSequence( const A : Ansistr) : Ansistr;
19673:  Func ASN1EncodeSet( const A : Ansistr) : Ansistr;
19674:  Func ASN1EncodeContextSpecific( const I : Int; const A : Ansistr) : Ansistr;
19675:  Func ASN1DecodeLength(const Buf:Str;const Size:Int;var Len:Int) : Int;
19676:  Func ASN1DecodeObjHeader(const Buf:str;const Size:Int;var TypeID:Byte;var Len:Int;var Data:str):Int;
19677:  Func ASN1TypeIsConstructedType( const TypeID : Byte) :Bool;
19678:  Func ASN1TypeIsContextSpecific( const TypeID : Byte; var Idx : Int) :Bool;
19679:  Func ASN1DecodeDataBoolean(const Buf:str;const Size:Int; var A:Boolean) : Int;
19680:  Func ASN1DecodeDataInt32(const Buf:str;const Size:Int;var A:LongInt): Int;
19681:  Func ASN1DecodeDataInt64(const Buf:str; const Size:Int; var A:Int64): Int;
19682:  Func ASN1DecodeDataIntBuf(const Buf:str; const Size:Int;var A: Ansistr): Int;
19683:  Func ASN1DecodeDataBitString(const Buf:str; const Size:Int;var A:Ansistr;var UnusedBits:Byte): Int;
19684:  Func ASN1DecodeDataRawAnsistr(const Buf:str;const Size:Int;var A:Ansistr):Int;
19685:  Func ASN1DecodeDataOctetString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19686:  Func ASN1DecodeDataIA5String(const Buf:str;const Size:Int;var A:Ansistr): Int;
19687:  Func ASN1DecodeDataVisibleString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19688:  Func ASN1DecodeDataNumericString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19689:  Func ASN1DecodeDataPrintableString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19690:  Func ASN1DecodeDataTeletexString(const Buf:str;const Size:Int; var A:Ansistr):Int;
19691:  Func ASN1DecodeDataUTF8String(const Buf:str; const Size:Int;var A:Ansistr):Int;
19692:  Func ASN1DecodeDataUniversalString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19693:  Func ASN1DecodeDataBMPString(const Buf:Str; const Size:Int; var A: Ansistr):Int;
19694:  Func ASN1DecodeDataOID(const Buf:str;const Size:Int;var A:TASN1ObjectIdentifier):Int;
19695:  Func ASN1DecodeDataUTCTime(const Buf:str; const Size:Int;var A TDateTime): Int;
19696:  Func ASN1DecodeDataGeneralizedTime(const Buf:str;const Size:Int;var A:TDateTime):Int;
19697:  Func ASN1Parse(const Buf:str;const Size:Int;const ParseProc:TASN1ParseProc;const CallerData:Int):Int;
19698: Func ASN1DecodeBoolean(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var A:Bool):Int;
19699: Func ASN1DecodeInt32( const TypeID:Byte;const DataBuf:str;const DataSize: Int; var A : LongInt): Int;
19700: Func ASN1DecodeInt64(const TypeID:Byte;const DatBuf:str;const DataSize:Int;var A:Int64):Int;
19701: Func ASN1DecodeIntBuf(const TypeID:Byte;const DataBuf:Str; const DataSize: Int;var A : Ansistr):Int;
19702:  Func ASN1DecodeBitString(const TypeID:Byte;const DataBuf:str; const DataSize:Int;var A:Ansistr;var
       UnusedBits:Byte):Int;
19703:  Func ASN1DecodeString(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var A:Ansistr):Int;
19704:  Func ASN1DecodeOID(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var
       A:TASN1ObjectIdentifier):Int;
19705:  Func ASN1DecodeTime const TypeID:Byte;const DataBuf:str; const DataSize:Int;var A:TDateTime): Int;
19706:  Proc SelfTestASN1;
19707: end;
19708:
19709: Proc SIRegister_cX509Certificate(CL: TPSPascalCompiler);
19710: begin
19711:   ClassN(CL.FindClass('TOBJECT'),'EX509;
19712:   TypeS(TX509AttributeType', 'TASN1ObjectIdentifier;
19713:   TypeS(TX509AttributeValue', 'Ansistr;
19714:   TypeS(TX509AttributeTypeAndValue', 'record AType : TX509AttributeType;'
19715:    +' Value : TX509AttributeValue; _Decoded :Bool; end;
19716:   //TypeS('PX509AttributeTypeAndValue', '^TX509AttributeTypeAndValue // wil not work;
19717:  Proc InitX509AttributeTypeAndValue(var A : TX509AttributeTypeAndValue; const AType : TX509AttributeType;
       const Value : TX509AttributeValue);
19718:  Proc InitX509AtName( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19719:  Proc InitX509AtSurname( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19720:  Proc InitX509AtGivenName( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19721:  Proc InitX509AtInitials( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19722:  Proc InitX509AtGenerationQuailifier(var A:TX509AttributeTypeAndValue; const B:Ansistr);
19723:  Proc InitX509AtCommonName(var A: TX509AttributeTypeAndValue; const B : Ansistr);
19724:  Proc InitX509AtLocailityName(var A:TX509AttributeTypeAndValue;const B: Ansistr);
19725:  Proc InitX509AtStateOrProvince(var A:TX509AttributeTypeAndValue;const B:Ansistr);
19726:  Proc InitX509AtOriganizationName(var A: TX509AttributeTypeAndValue; const B:Ansistr);
19727:  Proc InitX509AtOriganizationUnitName(var A:TX509AttributeTypeAndValue;const B:Ansistr);
19728:  Proc InitX509AtTitle( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19729:  Proc InitX509AtDnQualifier(var A: TX509AttributeTypeAndValue; const B: Ansistr);
19730:  Proc InitX509AtCountryName(var A :TX509AttributeTypeAndValue; const B : Ansistr);
19731:  Proc InitX509AtEmailAddress(var A: TX509AttributeTypeAndValue; const B: Ansistr);
19732:  Func EncodeX509AttributeTypeAndValue(const A:TX509AttributeTypeAndValue):Ansistr;
19733:  Func DecodeX509AttributeTypeAndValue(const TypeID  Byte;const Buf:str; const Size:Int;var
       A:TX509AttributeTypeAndValue): Int;
19734:   TypeS('TX509RelativeDistinguishedName', 'array of TX509AttributeTypeAndValue;
19735:  // TypeS('PX509RelativeDistinguishedName','^TX509RelativeDistinguishedName //will not work;
19736:  Proc AppendX509RelativeDistinguishedName( var A : TX509RelativeDistinguishedName; const V :
       TX509AttributeTypeAndValue);
19737:  Func EncodeX509RelativeDistinguishedName(const A:TX509RelativeDistinguishedName):AnsiStr;
19738:  Func DecodeX509RelativeDistinguishedName(const TypeID:Byte; const Buf:Str; const Size:Int;var
       A:TX509RelativeDistinguishedName): Int;
19739:  TypeS('TX509RDNSequence', 'array of TX509RelativeDistinguishedName;
```

```
19740:   //TypeS('PX509RDNSequence', '^TX509RDNSequence // will not work;
19741:  TypeS('TX509Name', 'TX509RDNSequence;
19742:  Proc AppendX509RDNSequence(var A:TX509RDNSequence;const B:TX509RelativeDistinguishedName);
19743:  Func EncodeX509RDNSequence( const A : TX509RDNSequence) : Ansistr;
19744: Func DecodeX509RDNSequence(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RDNSequence):Int;
19745:  Func EncodeX509Name( const A : TX509Name) : Ansistr;
19746: Func DecodeX509Name(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Name):Int;
19747:  'CurrentX509Version','string').SetString(' X509v3;
19748: type
19749:  TX509Version = (
19750:      X509v1 = 0,
19751:      X509v2 = 1,
19752:      X509v3 = 2,
19753:      X509vUndefined = $FF // implementation defined value   );
19754:  TX509Version', '(X509v1, X509v2,X509v3, X509vUndefined);
19755:
19756: type
19757:  TX509GeneralNameType = (
19758:  gnOtherName                  = 0,
19759:  gnRFC822Name                 = 1,
19760:  gnDNSName                    = 2,
19761:  gnX400Address                = 3,
19762:  gnDirectoryName              = 4,
19763:  gnEDIPartyName               = 5,
19764:  gnUniformResourceIdentifier = 6,
19765:  gnIPAddress                  = 7,
19766:  gnRegisteredID               = 8);
19767: TypeS(TX509GeneralNameType',(gnOtherName,gnRFC822Name,gnDNSName,gnX400Address,gnDirectoryName,
       gnEDIPartyName,gnUniformResourceIdentifier,gnIPAddress,gnRegisteredID);
19768:
19769:  Proc InitX509Version( var A : TX509Version);
19770:  Func EncodeX509Version( const A : TX509Version) : Ansistr;
19771:  Func DecodeX509Version(const TypeID:Byte;const Buf:str;const Size:Int;var A: TX509Version):Int;
19772:   TypeS(TX509Time', TDateTime;
19773:  Func EncodeX509Time( const A : TX509Time) : Ansistr;
19774:  Func DecodeX509Time(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Time):Int;
19775:   TX509Validity','record NotBefore:TX509Time; NotAfter:TX509Time; _Decoded:Bool; end;
19776:  // TypeS('PX509Validity', '^TX509Validity // will not work;
19777:  Func EncodeX509Validity( const A : TX509Validity) : Ansistr;
19778:  Func DecodeX509Validity(const TypeID:Byte;const Buf:Str; const Size:Int; var A: TX509Validity): Int;
19779:   TypeS('TX509DHValidationParms','record Seed: Ansistr; PgenCounter : Int; end;
19780:   //TypeS('PX509DHValidationParms', '^TX509DHValidationParms // will not work;
19781:   TypeS('TX509DHDomainParameters', record P : Ansistr; G : Ansistr'
19782:    +; Q : Ansistr; J : Ansistr; ValidationParms : TX509DHValidationParms; end;
19783:  // TypeS('PX509DHDomainParameters', '^TX509DHDomainParameters // will not work;
19784:  Func EncodeX509DHValidationParms( const A : TX509DHValidationParms) : Ansistr;
19785:  Func DecodeX509DHValidationParms(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509DHValidationParms): Int;
19786:  Func EncodeX509DHDomainParameters( const A: TX509DHDomainParameters) : Ansistr;
19787:  Func DecodeX509DHDomainParameters(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509DHDomainParameters):Int;
19788:   TypeS('TX509DSSParms', 'record P : Ansistr; Q : Ansistr; G : Ansistr; end;
19789:   //TypeS('PX509DSSParms', '^TX509DSSParms // will not work;
19790:  Func EncodeX509DSSParms( const A : TX509DSSParms) : Ansistr;
19791:  Func DecodeX509DSSParms(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509DSSParms):Int;
19792: TypeS('TX509DSSSigValue', 'record R : Ansistr; S : Ansistr; end;
19793:   //TypeS('PX509DSSSigValue', '^TX509DSSSigValue // will not work;
19794:  Func EncodeX509DSSSigValue( const A : TX509DSSSigValue) : Ansistr;
19795:  Func DecodeX509DSSSigValue(const TypeID:Byte; const Buf:Str; const Size:Int; var A:TX509DSSSigValue):Int;
19796:   TypeS(TX509AlgorithmIdentifier','record Algorithm : TASN1ObjectIdenti'
19797:    +fier; Parameters : Ansistr; _Decoded :Bool; end;
19798: //TypeS('PX509AlgorithmIdentifier', '^TX509AlgorithmIdentifier // will not work;
19799:  Proc InitX509AlgorithmIdentifier(var A:TX509AlgorithmIdentifier; const Algorithm : array of Int; const
       Parameters : Ansistr);
19800:  Proc InitX509AlgorithmIdentifierDSA_SHA1(var A:TX509AlgorithmIdentifier;const Params:AnsiStr;
19801:  Func EncodeX509AlgorithmIdentifier( const A : TX509AlgorithmIdentifier) : Ansistr;
19802:  Func DecodeX509AlgorithmIdentifier(const TypeID:Byte; const Buf:str; const Size:Int; var
       A:TX509AlgorithmIdentifier): Int;
19803:  TypeS(TX509RSAPublicKey','record Modulus:Ansistr;PublicExponent:Ansistr; end;
19804:  //TypeS('PX509RSAPublicKey', '^TX509RSAPublicKey // will not work;
19805:  Func EncodeX509RSAPublicKey( const A : TX509RSAPublicKey) : Ansistr;
19806:  Func DecodeX509RSAPublicKey(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RSAPublicKey): Int;
19807:  Func ParseX509RSAPublicKey(const Buf:str;const Size:Int;var A:TX509RSAPublicKey):Int);
19808:   TypeS('TX509DHPublicKey', 'Ansistr;
19809:  Func EncodeX509DHPublicKey( const A : TX509DHPublicKey) : Ansistr;
19810:   TypeS(TX509DSAPublicKey', 'Ansistr;
19811:  Func EncodeX509DSAPublicKey( const A : TX509DSAPublicKey) : Ansistr;
19812:   TypeS(TX509SubjectPublicKeyInfo', 'record Algorithm : TX509AlgorithmId'
19813:    +entifier; SubjectPublicKey : Ansistr; _Decoded :Bool; end;
19814:  //TypeS('PX509SubjectPublicKeyInfo', '^TX509SubjectPublicKeyInfo // will not work;
19815:  Proc InitX509SubjectPublicKeyInfoDSA(var A:TX509SubjectPublicKeyInfo;const B:TX509DSSParms;const
       PublicKey: Ansistr);
19816:  Func EncodeX509SubjectPublicKeyInfo( const A: TX509SubjectPublicKeyInfo) : Ansistr;
19817:  Func DecodeX509SubjectPublicKeyInfo(const TypeID:Byte;const Buf:str;const Size:Int;var A
       TX509SubjectPublicKeyInfo): Int;
19818:  Func ParseX509SubjectPublicKeyInfo(const Buf:str;const Size:Int;var A:TX509SubjectPublicKeyInfo):Int;
19819:  Func EncodeX509GeneralName(const A:TX509GeneralNameType;const EncodedName:Ansistr):Ansistr;
19820:  Func DecodeX509GeneralName(const TypeID:Byte; const Buf:str; const Size:Int;var
       A:TX509GeneralNameType;var B  Ansistr): Int;
```

```
19821:  TX509GeneralNames','array of record NameType:TX509GeneralNameType; Name : Ansistr; end;
19822:   //TypeS('PX509GeneralNames', '^TX509GeneralNames // will not work;
19823:  Func EncodeX509GeneralNames( const A : TX509GeneralNames) : Ansistr;
19824:  Func DecodeX509GeneralNames(const TypeID:Byte;const Buf:str; const Size:Int; var A:TX509GeneralNames):Int;
19825:   TX509BasicConstraints,record CA:Bool;PathLenConstraint:Ansistr;_DecodedCA:Boolean;end
19826:   //TypeS('PX509BasicConstraints', '^TX509BasicConstraints // will not work;
19827:  Func EncodeX509BasicConstraints( const A : TX509BasicConstraints) : Ansistr;
19828:  Func DecodeX509BasicConstraints(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509BasicConstraints): Int;
19829:  TypeS(TX509AuthorityKeyIdentifier', 'record KeyIdentifier : Ansistr'
19830:  +; AuthorityCertIssuer : TX509GeneralNames; AuthorityCertSerialNumber : Int64; end;
19831:  //TypeS('PX509AuthorityKeyIdentifier','^TX509AuthorityKeyIdentifier //will not work;
19832: Func EncodeX509AuthorityKeyIdentifier(const A:TX509AuthorityKeyIdentifier): Ansistr;
19833:  TypeS(TX509SubjectKeyIdentifier', 'Ansistr;
19834:  TypeS('TX509KeyUsage', 'Ansistr;
19835:  TypeS(TX509Extension', 'record ExtnID : TASN1ObjectIdentifier; Critica'
19836:   +l :Bool; ExtnValue: Ansistr;_DecodedCritical:Bool; _Decoded :Bool; end;
19837:  //TypeS('PX509Extension', '^TX509Extension // will not work;
19838:  Proc InitX509ExtAuthorityKeyIdentifier(var A: TX509Extension; const B TX509AuthorityKeyIdentifier);
19839:  Proc InitX509ExtSubjectKeyIdentifier(var A:TX509Extension; const B:TX509SubjectKeyIdentifier);
19840:  Proc InitX509ExtBascConstraints(var A:TX509Extension;const B:TX509BasicConstraints);
19841:  Func EncodeX509Extension( const A : TX509Extension) : Ansistr;
19842:  Func DecodeX509Extension(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Extension):Int;
19843:  TypeS('TX509Extensions', 'array of TX509Extension;
19844:   //TypeS('PX509Extensions', '^TX509Extensions // will not work;
19845:  Proc AppendX509Extensions( var A : TX509Extensions; const B : TX509Extension);
19846:  Func EncodeX509Extensions( const A : TX509Extensions) : Ansistr;
19847:  Func DecodeX509Extensions( const TypeID: Byte; const Buf:Str;const Size: Int;var A: TX509Extensions):Int;
19848:  Func NormaliseX509IntKeyBuf( var KeyBuf : Ansistr) : Int;
19849:   TypeS(TX509TBSCertificate', 'record Version : TX509Version; SerialNumb'
19850:    +er : Ansistr; Signature : TX509AlgorithmIdentifier; Issuer : TX509Name;'
19851:    + Validity : TX509Validity; Subject : TX509Name; SubjectPublicKeyInfo : TX5'
19852:    +09SubjectPublicKeyInfo; IssuerUniqueID : Ansistr; SubjectUniqueID : Ans'
19853:    +iString; Extensions: TX509Extensions;_DecodedVersion :Bool; _Decoded:Bool; end;
19854:  // TypeS('PX509TBSCertificate', '^TX509TBSCertificate // will not work;
19855:  Func EncodeX509TBSCertificate( const A : TX509TBSCertificate) : Ansistr;
19856:  Func DecodeX509TBSCertificate(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509TBSCertificate):Int;
19857:  TypeS('TX509Certificate', record TBSCertificate : TX509TBSCertificate;'
19858:    + SignatureAlgorithm: TX509AlgorithmIdentifier;SignatureValue:Ansistr; _Decoded:Bool;end;
19859:  // TypeS('PX509Certificate', '^TX509Certificate // will not work;
19860:  TypeS('TX509CertificateArray', 'array of TX509Certificate;
19861:  Proc InitX509Certificate( var A : TX509Certificate);
19862:  Func EncodeX509Certificate( const A : TX509Certificate) : Ansistr;
19863:  Func DecodeX509Certificate(const TypeID:Byte; const Buf:Str;const Size: Int; var A:TX509Certificate):Int;
19864:  Func ParseX509Certificate(const Buf:str;const Size:Int;var A:TX509Certificate):Int;
19865:  Proc ParseX509CertificateStr( const BufStr : Ansistr; var A : TX509Certificate);
19866:  Proc ParseX509CertificatePEM( const BufStr : Ansistr; var A : TX509Certificate);
19867:   TypeS('TX509RSAPrivateKey', record Version : Int; Modulus : AnsiSt'
19868:    +ring; PublicExponent : Ansistr; PrivateExponent : Ansistr; Prime1 : '
19869:    +Ansistr; Prime2 Ansistr;Exponent1:Ansistr;Exponent2:Ansistr;CRTCoefficient:Ansistr; _Decoded:Bool; end;
19870:  // TypeS('PX509RSAPrivateKey', '^TX509RSAPrivateKey // will not work;
19871:  Func EncodeX509RSAPrivateKey( const A : TX509RSAPrivateKey) : Ansistr;
19872:  Func DecodeX509RSAPrivateKey(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RSAPrivateKey):Int;
19873:  Func ParseX509RSAPrivateKey(const Buf:str;const Size:Int;var A:TX509RSAPrivateKey):Int;
19874:  Proc ParseX509RSAPrivateKeyStr(const BufStr: Ansistr;var A: TX509RSAPrivateKey);
19875:  Proc ParseX509RSAPrivateKeyPEM(const BufStr: Ansistr;var A: TX509RSAPrivateKey);
19876:  Proc SelfTestX509;
19877: end;
19878:
19879: Proc SIRegister_FileStreamW(CL: TPSPascalCompiler);
19880: begin
19881:   SIRegister_TFileStreamW(CL);
19882:   {TFormFadeSettings = record
19883:     Form: TForm;
19884:     Step: ShortInt;
19885:     DisableBlendOnFinish:Bool;
19886:     Callback: TNotifyEvent;
19887:     MinAlpha, MaxAlpha: Byte;
19888:   end;}
19889:  TFormFadeSettings','record Form:TForm;Step:ShortInt;DisableBlendOnFinish:Bool;Callback:TNotifyEvent;
       MinAlpha,MaxAlpha:Byte;end;
19890:  ConstantN('fmForcePath','LongWord').SetUInt( $80000000);
19891:  Func LoadUnicodeFromStream( S : TStream; AsIsAnsi :Bool) : WideString;
19892:  Func GetClipboardText: WideString;;
19893:  Proc CopyToClipboard(Str: WideString);;
19894:  Func CurrentWinUser: WideString;;
19895:  Func GetTempPathW: WideString;;
19896:  Func GetTempFileNameW: WideString;;
19897:  Func GetDesktopFolderW: WideString;;
19898:  Func IsWritable(const FileName: WideString):Bool;;
19899:  Func SysErrorMessageW(ErrorCode: Int): WideString;;
19900:  Func FormatExceptionInfo: WideString;;
19901:  Proc ShowExceptionW(Message: WideString);;
19902:  Proc ChangeWindowStyle(const Form: HWND; Style: DWord; AddIt:Bool);;
19903:  Func SetNtfsCompressionW(const FileName: WideString; Level: Word):Bool;;
19904:  Func WriteWS(const Stream: TStream; const Str: WideString): Word;;
19905:  Proc FormFadeIn(Form: TForm; Step: ShortInt);;
19906:  Proc FormFadeOut(Form: TForm; Step: ShortInt);;
```

```
19907:  Proc FormFadeOutAndWait(Form: TForm; Step: ShortInt);;
19908:  Func BrowseForFolderw(const Caption,DefaultPath:WideStr;const OwnerWindow:HWND):WideStr;
19909:  Proc FormFade(const Settings: TFormFadeSettings);;
19910:  Func HashOfString(const Str: WideString): DWord;
19911:  Func ComparePoints(const First, Second: TPoint): ShortInt;
19912: end;
19913:
19914: Proc SIRegister_InetUtils(CL: TPSPascalCompiler);
19915: begin
19916:  TypeS(TNetUtilsSettings', 'record UserAgent :Str; ProxyURL : Strin'
19917:   +g; OpenURLFlags: DWord; TrafficCounter:Dword;UploadedCounter:DWord;ReadBufferSize:DWord;end;
19918:   TRawCharset', 'set of Char;
19919:   TInetHeaders', 'array of String;
19920:   SIRegister_EInet(CL);
19921:  TInetDownloadCallback', 'Func ( Downloaded, TotalSize : DWord) :Bool;
19922: Func InetDownloadTo(const DestFile:WideString;const URL:str;Callback:TInetDownloadCallback):Bool;
19923:  Func InetDownloadTo1(const DestFile:WideString;const URL:Str;const Settings:
       TNetUtilsSettings;Callback:TInetDownloadCallback):Bool;;
19924:  Func InetDownload(const URL:str;Dest:TStream;Callback:TInetDownloadCallback):Bool;
19925:  Func InetDownload3(const URL:Str; Dest:TStream; const Settings:TNetUtilsSettings;Callback :
       TInetDownloadCallback) :Bool;;
19926:  Func InetBufferedReadFrom(Handle:HInternet;Dest:TStream;const
       Settings:TNetUtilsSettings;Callback:TInetDownloadCallback):DWord;;
19927:  Func InetBufferedReadFrom5( Handle : HInternet; const Settings : TNetUtilsSettings; Callback :
       TInetDownloadCallback) :Str;;
19928:  Func IsResponseStatusOK( Handle : HInternet) :Bool;
19929:  TMultipartItem', 'record Headers : TInetHeaders; Data : TStream; end;
19930:  { TMaskMatchInfo = record
19931:    Matched:Bool;
19932:    StrPos: Word;
19933:    MatchLength: Word;
19934:  end;}
19935:  TMaskMatchInfo', 'record Matched:Bool; StrPos: Word; MatchLength: Word; end;
19936:  TMultipartItems', 'array of TMultipartItem;
19937:   TUploadFile','record Name:str; SourceFileName : WideString; Data : TStream; end;
19938:   TUploadFiles', 'array of TUploadFile;
19939:  Func FindBoundaryFor( const Items : TMultipartItems) :Str;
19940:  Func RandomBoundary :Str;
19941: Func GenerateMultipartFormFrom(const Items:TMultipartItems;out ExtraHeadrs:TInetHeaders):Str
19942:  Func InetUploadTo( const ToURL:Str;const Headers:TInetHeaders;const Items: TMultipartItems; const
       Settings:TNetUtilsSettings) :Str;;
19943:  Func InetUploadTo7(const ToURL:Str;const Items:TMultipartItems;const Settings:TNetUtilsSettings):str;
19944:  Func InetUploadTo8(const ToURL:Str; const Items: TMultipartItems) :Str;;
19945:  Func InetUploadStreamsTo(const ToURL:str;const Settings:TNetUtilsSettings;Streams:TUploadFiles):str;
19946:  Func InetUploadStreamsTo10(const ToURL:str;const Streams:TUploadFiles):str;
19947:  Func InetUploadFileTo(const ToURL:str;const Settings:TNetUtilsSettings;const ItemName:str;const
       FilePath:WideString):str;
19948:  Func InetUploadFileTo12(const ToURL:Str;const ItemName:Str;const FilePath:WideStr):str;
19949:  Func InetUploadFilesTo(const ToURL:str;const Settings:TNetUtilsSettings;const Files:array of const):str;
19950:  Func InetUploadFilesTo14(const ToURL:Str;const Files: array of const):str;
19951:  Func AppendQueryTo(const URL:Str; const Arguments: array of const):str;
19952:  Func HasQueryPart( const URL :Str) :Bool;
19953:  Func BuildQueryFrom( const Arguments : array of const) :Str;
19954:  Func BuildURLW(Protocol,Host:Str;Port:Word;Path,Script:Str;const Arguments:array of const):Str
19955:  Func CustomEncode(const Str: WideString; const RawChars:TRawCharset):Str;
19956:  Func EncodeURI( const Str : WideString) :Str;
19957:  Func EncodeURIComponent( const Str : WideString) :Str;
19958:  Proc InetGetLastError( out ErrorCode : DWord; out ErrorMessage :Str);
19959:  Func InetGetLastErrorCode : DWord;
19960:  Func InetGetLastErrorMsg :Str;
19961:  Func AbsoluteURLFrom( URL, BaseURL, BasePath :Str) :Str;
19962:  Proc SplitURL( const URL :Str; out Domain, Path :Str);
19963:  Func DomainOf( const URL :Str) :Str;
19964:  Func PathFromURL( const URL :Str) :Str;
19965:  Func InetHeaders( const NameValues : array of const) : TInetHeaders;
19966:  Func NoInetHeaders : TInetHeaders;
19967:  Func JoinHeaders( const Headers : TInetHeaders) :Str;
19968:  ConstantN('InetHeaderEOLN','String').SetString( #13#10);
19969:  Proc SetDefaultNetUtilsSettings;
19970:  Func TotalDownTrafficThroughNetUtils : DWord;
19971:   TDBDraw', 'record DisplayDC : HDC; MemDC : HDC; MemBitmap : HBIT'
19972:    +'MAP; OldBitmap : HBITMAP; OldFont : HFONT; OldPen : HPEN; end;
19973:   TPieceFormatData', 'record Position : TMaskMatchInfo; Color : TColor; end;
19974:   TFormatData', 'array of TPieceFormatData;
19975:   TDrawFormattedTextSettings', 'record Text : WideString; FormatDa'
19976:    +'ta : TFormatData; Canvas : TCanvas; WrapText :Bool; DestPos : TPoint; '
19977:    +'MaxWidth : Word; CharSpacing : Word; end;
19978:   TWrapTextSettings', 'record DC : HDC; Str : WideString; Delimite'
19979:    +r :WideString; MaxWidth:Word; LeftMargin:Word; CharSpacing:Word;LastChar: TSize; end;
19980:  Func TextSize( const DC : HDC; const Str : WideString) : TSize;
19981:  Func TextWidthW2( const DC : HDC; const Str : WideString) : Int;
19982:  Func TextHeightW2( const DC : HDC; const Str : WideString) : Int;
19983:  Func GetLineHeightOf( const Font : HFONT) : Word;
19984:  Func TextWidthEx(const DC:HDC;const Str:WideString; const CharSpacing:Word): Int;
19985:  Func TextHeightEx(const DC:HDC;const Str:WideString; const CharSpacing:Word): Int;
19986:  Func TextSizeEx(const DC:HDC;const Str:WideString;const CharSpacing:Word):TSize;
19987:  Func TextWithBreaksSize( Settings : TWrapTextSettings) : TSize;;
19988:  Func DoubleBufferedDraw(const DisplaySurface:HDC;const BufferSize:TPoint):TDBDraw;
19989:  Func DoubleBufferedDraw17(const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
```

```
19990:  Func DoubleBufferedDraw2(const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
19991:  Proc DrawFormattedText( const Settings : TDrawFormattedTextSettings);
19992: Func GetLastCharPos(const DC:HDC;const Str:WideString;const MaxWidth:Word;const ChrSpacing:Word):TSize;
19993:  Func WrapNonMonospacedText(const DC:HDC;const Str:WideString;const Delimiter:WideString; const
       MaxWidth:Word;const CharSpacing:Word):WideString;;
19994:  Func WrapNonMonospacedText2( var Settings : TWrapTextSettings) : WideString;;
19995: end;
19996:
19997: Proc SIRegister_StrConv(CL: TPSPascalCompiler);
19998: begin
19999:  TypeS('TCodepage', 'DWord;
20000:  ConstantN('CP_INVALID','LongInt').SetInt( TCodepage ( - 1 ));
20001:  'CP_ASIS','LongInt').SetInt( TCodepage ( - 2 ));
20002:  'CP_ANSI','longint').SetInt(0);
20003:  'CP_OEM','longint').SetInt(1);
20004:  'CP_SHIFTJIS','LongInt').SetInt( 932);
20005:  'CP_LATIN1','LongInt').SetInt( 1250);
20006:  'CP_UNICODE','LongInt').SetInt( 1200);
20007:  'CP_UTF8','LongInt').SetInt( 65001);
20008:  Func MinStrConvBufSize( SrcCodepage : TCodepage; Str :Str) : Int;;
20009:  Func MinStrConvBufSize1( DestCodepage : TCodepage; Wide : WideString) : Int;;
20010:  Func ToWideString(SrcCodepage: TCodepage; Str :Str; BufSize: Int) : WideString;
20011:  Func FromWideString(DestCodepage:TCodepage;Str:WideString;BufSize:Int;Fail:Boolean):str;
20012:  Func CharsetToID( Str :Str) : TCodepage;
20013:  Func IdToCharset( ID : TCodepage; GetDescription :Bool) :Str;
20014:  Func CompareStrW(const S1, S2: WideString; Flags: DWord = 0): Int;
20015:  Func CompareTextW(const S1, S2: WideString): Int;
20016:  Func MaskMatch(const Str, Mask: WideString):Bool;
20017:  { Info can have special values in some cases:
20018:   * Matched = True but MatchLength = 0(and StrPos having random value) - means that Mask consisted of only
       "*" and
20019:     thus no particular substring could specified (since it could match anypart ofthe string). }
20020:  Func MaskMatchInfo(const Str,Mask:WideString;StartingPos:Word = 1):TMaskMatchInfo;
20021:  //Strutils ------------------------
20022:   Func TryStrToIntStrict(const S:Str; out Value:Int; Min:Int):Boolean;
20023:  Func TryStrToFloatStrict(const S:str; out Value:Single;const FormatSettings:TFormatSettings):Bool;
20024:  Func TryStrToFloatStrict1(const S:str;out Value:Double;const FormatSettings: TFormatSettings):Bool;
20025:  Func DetectEolnStyleIn( const Str : WideString) : WideString;
20026:  Func DetectEolnStyleInANSI( Stream : TStream) : WideString;
20027:  Func PascalQuote( const Str : WideString) : WideString;
20028:  Func StrRepeatW( const Str : WideString; Times : Int) : WideString;
20029:  Func EscapeString(const Str: WideString; CharsToEscape:WideString): WideString;
20030:  Func UnescapeString(const Str: WideString; CharsToEscape: WideString):WideString;
20031:  Func BinToHexW( const Buf :Str; Delim :Str) :Str;
20032:  Func HexToBinW( Text :Str) :Str;
20033:  Func SoftHexToBin( Text :Str) :Str;
20034:  Func FormatVersion( Version : Word) : WideString;
20035:  Func FormatDateW( Date : DWord) : WideString;
20036:  Func FormatNumber( Number : DWord) : WideString;
20037:  Func GenericFormat(Number:Single;const Language:TGenericFormatLanguage):WideString;
20038:  Func FormatInterval( Millisecs : DWord) : WideString;
20039:  Func FormatSize( Bytes : DWord) : WideString;
20040:  Func PosLast( const Substr, Str :Str; Start : Word) : Int;
20041:  Func PosLastW( const Substr, Str : WideString; Start : Word) : Int;
20042:  Func IsDelimiterW(const Delimiters,S: WideString; Index: Int):Bool;
20043:  Func RemoveNonWordChars(const Str: WideString;DoNotRemove:WideString):WideString;
20044:  Func IsQuoteChar( const aChr : Char) :Bool;
20045:  Func WrapTextW(const Str:WideStr;const Delimiter:WideStr;const MaxWidth:Word):WideStr;
20046:  Func PadText(const Str:WideStr;const NewLine,PadStr:WideStr;const MaxWidth:Word):WideStr;
20047:  Func PadTextWithVariableLineLength(const Str:WideString;const NewLine,PadStr:WideString;const
       LineLengths:array of Int):WideString;
20048:  Func StrPadW(const Str:WideString; ToLength:Int; PadChar: WideChar) : WideString;
20049:  Func StrPadLeftW(const Str:WideString;ToLength:Int;PadChar: WideChar): WideString;
20050:  //Func StrRepeat( const Str : WideString; Times : Int) : WideString;
20051:  Func StrReverseW( const Str : WideString) : WideString;
20052:  Func CountSubstr( const Substr, Str : WideString) : Int;
20053:  Proc DeleteArrayItem( var A : TWideStringArray; Index : Int);
20054:  Func TrimStringArray( WSArray : TWideStringArray) : TWideStringArray;
20055:  Func TrimWS( Str : WideString; const Chars : WideString) : WideString;
20056:  Func TrimLeftWS( Str : WideString; const Chars : WideString) : WideString;
20057:  //Func TrimRightWS( Str : WideString; const Chars : WideString) : WideString;
20058:  Func ConsistsOfChars( const Str, Chars : WideString) :Bool;
20059:  Func UpperCaseW( const Str : WideString) : WideString;
20060:  Func LowerCaseW( const Str : WideString) : WideString;
20061:  Func UpperCaseFirst( const Str : WideString) : WideString;
20062:  Func LowerCaseFirst( const Str : WideString) : WideString;
20063:  Func StripAccelChars( const Str : WideString) : WideString;
20064: end;
20065:
20066: Proc SIRegister_REXX(CL: TPSPascalCompiler);
20067: begin
20068:   TypeS(TState', '( TrimLeader, StartToken, EndToken );
20069:   TStrIndex', 'LongInt;
20070:   TTokIndex', 'WORD;
20071:   TStrIndexB', 'BYTE;
20072:   TTokIndexB', 'BYTE;
20073:   ClassN(CL.FindClass('TOBJECT'),'EConversionError;
20074:  Func Abbrev(const information, info:Str; const nMatch: TStrIndex):BOOLEAN;
20075:  Func AllSame( const s :Str; const c : CHAR) :Bool;
```

```
20076:  Func Capitalize( const s :Str) :Str;
20077:  Func Center( const s :Str; const sLength : TStrIndex) :Str;
20078:  Func Left( const s :Str; const sLength : TStrIndex) :Str;
20079:  Func Right( const s :Str; const sLength : TStrIndex) :Str;
20080:  Func Copies( const s :Str; const n : TStrIndex) :Str;
20081:  Func CountChar( const s :Str; const c : CHAR) : TStrIndex;
20082:  Func DeleteStringrexx( const substring :Str; const s :Str) :Str;
20083:  Func Overlay( const ovly, target :Str; const n : TStrIndex) :Str;
20084:  Func Plural(const n: LongInt; const singularform, pluralform :Str) :Str;
20085:  Func Reverse( const s :Str) :Str;
20086:  Func Spacerexx( const s :Str; const n : TStrIndex) :Str;
20087:  Func Striprexx( const s :Str; const option :Str) :Str;
20088:  Func TestString( const sLength : TStrIndex) :Str;
20089:  Func Translate( const s, OutTable, InTable :Str) :Str;
20090:  Func XRange( const start, stop : BYTE) :Str;
20091:  Func B2X(const b: BYTE) :Str;
20092:  Func C2D(const s:Str) : DOUBLE;
20093:  Func C2I(const s:Str) : Int;
20094:  Func C2L(const s:Str) : LONGINT;
20095:  Func C2W(const s:Str) : WORD;
20096:  Func C2X(const s:Str) :Str;
20097:  Func I2C(const i: Int) :Str;
20098:  Func I2X(const i: Int) :Str;
20099:  Func L2C(const i: LONGINT) :Str;
20100:  Func L2X(const i: LONGINT) :Str;
20101:  Func D2C(const x: DOUBLE; const d : BYTE):Str;
20102:  Func W2C(const w: WORD) :Str;
20103:  Func W2X(const w: WORD) :Str;
20104:  Func X2W(const s:Str) : WORD;
20105:  Func JulianDate( const DateTime : TDateTime) : LongInt;
20106:  Func TimeDifference( const StartTime, StopTime : TDateTime) : DOUBLE;
20107:  Func Pwr( const x, y : DOUBLE) : DOUBLE;
20108: end;
20109:
20110: Proc SIRegister_StringGridLibrary(CL: TPSPascalCompiler);
20111: begin
20112:  Proc ReadGridFile( var StringGrid : TStringGrid; GridFile :Str);
20113:  Proc WriteGridFile( var StringGrid : TStringGrid; GridFile :Str);
20114:  Proc AddBlankRowToTop( var StringGrid : TStringGrid);
20115:  Proc DeleteSelectedRow( var StringGrid : TStringGrid);
20116:  Func StringGridSearch(const StringGrid:TStringGrid;const column:Int;const target:STR):Int;
20117:  Func XLeft( rect : TRect; canvas : TCanvas; s :Str) : Int;
20118:  Func XCenter( rect : TRect; canvas : TCanvas; s :Str) : Int;
20119:  Func XRight( rect : TRect; canvas : TCanvas; s :Str) : Int;
20120:  Func YCenter( rect : TRect; canvas : TCanvas; s :Str) : Int;
20121: end;
20122:
20123: Proc SIRegister_InetUtils2(CL: TPSPascalCompiler);
20124: begin
20125:  Proc AdjustArray(var DWArray:array of DWord;const Delta:Int;MinValueToAdjust:DWord);
20126:  Func GetDroppedFileNames( const DropID : Int) : TWideStringArray;
20127:  Func AreBytesEqual( const First, Second : array of Byte) :Bool;;
20128:  Func AreBytesEqual1( const First, Second, Length : DWord) :Bool;;
20129:  Func MaskForBytes( const NumberOfBytes : Byte) : DWord;
20130:  Func IntToBinByte( Int : Byte) :Str;;
20131:  Func IntToBinWord( Int : Word) :Str;;
20132:  Func IntToBinDWord( Int : DWord; Digits : Byte; SpaceEach : Byte) :Str;;
20133:  Func WriteWS( const Stream : TStream; const Str : WideString) : Word;
20134:  Proc WriteArray( const Stream : TStream; const WSArray : array of WideString);;
20135:  Proc WriteArray6( const Stream : TStream; const DWArray : array of DWord);;
20136:  Func ReadWS( const Stream : TStream) : WideString;;
20137:  Func ReadWS8( const Stream : TStream; out Len : Word) : WideString;;
20138:  Proc ReadArray( const Stream : TStream; var WSArray : array of WideString);;
20139:  Proc ReadArray10( const Stream : TStream; var DWArray : array of DWord);;
20140:  Func ParamStrW( Index : Int) : WideString;
20141:  Func ParamStrFrom( CmdLine : WideString; Index : Int) : WideString;
20142:  Func ParamStrEx( CmdLine : WideString; Index : Int; out Pos : Int) : WideString;
20143:   Proc FindMask( Mask : WideString; Result : TStringsW);
20144:  Proc FindAll( BasePath, Mask : WideString; Result : TStringsW);
20145:  Proc FindAllRelative( BasePath, Mask : WideString; Result : TStringsW);
20146:  Func IsInvalidPathChar( const Char : WideChar) :Bool;
20147:  Func MakeValidFileNameW(const Str:WideString;const SubstitutionChar:WideChar):WideString;
20148:   Func ExtractFilePathW( FileName : WideString) : WideString;
20149:  Func ExtractFileNameW( Path : WideString) : WideString;
20150:  Func ExpandFileNameW( FileName : WideString) : WideString;;
20151:  Func ExpandFileName12( FileName, BasePath : WideString) : WideString;;
20152:  Func CurrentDirectory : WideString;
20153:  Func ChDirW( const ToPath : WideString) :Bool;
20154:  Func ExtractFileExtW( FileName : WideString) : WideString;
20155:  Func ChangeFileExtW2( FileName, Extension : WideString) : WideString;
20156:  Func IncludeTrailingBackslashW( Path : WideString) : WideString;
20157:  Func ExcludeTrailingBackslashW( Path : WideString) : WideString;
20158:  Func IncludeTrailingPathDelimiterW( Path : WideString) : WideString;
20159:  Func ExcludeTrailingPathDelimiterW( Path : WideString) : WideString;
20160:  Func IncludeLeadingPathDelimiter( Path : WideString) : WideString;
20161:  Func ExcludeLeadingPathDelimiter( Path : WideString) : WideString;
20162:  Func FileInfo( Path : WideString) : TWin32FindData;
20163:  Func IsDirectoryW( Path : WideString) :Bool;
20164:  Func FileAgeW( const FileName : WideString) : Int;
```

```
20165:  Func FileExistsW( Path : WideString) :Bool;
20166:  Func FileSizeW( Path : WideString) : DWord;
20167:  Func FileSize64( Path : WideString) : Int64;
20168:  Func DeleteFileW( Path : WideString) :Bool;
20169:  Func CopyDirectoryW( Source, Destination : WideString) :Bool;
20170:  Func RemoveDirectoryW( Path : WideString) :Bool;
20171:  Func ForceDirectoriesW( Path : WideString) :Bool;
20172:  Func MkDirW( Path : WideString) :Bool;
20173:  Func GetEnvironmentVariableW( Name : WideString) : WideString;
20174:  Func ResolveEnvVars(Path:WideString; Callback:TEnvVarResolver;Unescape:Boolean):WideString;
20175:  Func ResolveEnvVars14( Path : WideString; Unescape :Bool) : WideString;;
20176:  Func ReadRegValue( Root : DWord; const Path, Key : WideString) : WideString;
20177:  Func FadeSettings(Form:TForm;Step:ShortInt;Callback:TNotifyEvent):TFormFadeSettings;
20178:  Func FadeSettings17(Form:TForm;MinAlpha,MaxAlpha:Byte;Step:ShortInt):TFormFadeSettings
20179:  Func GetSpecialFolderID( const Path :Str) : Int;
20180:  Func GetSpecialFolderPath2( FolderID : Int) :Str;
20181: end;
20182:
20183: Proc SIRegister_KFunctions(CL: TPSPascalCompiler);
20184: begin
20185:  SHFolderDll','String').SetString( 'SHFolder.dll;
20186:  ConstantN('KM_BASE','LongInt').SetInt( LM_USER + 1024);
20187:  ConstantN('KM_LATEUPDATE','LongInt').SetInt( KM_BASE + 1);
20188:  crHResize','LongInt).SetInt( TCursor ( 101 ));
20189:  ConstantN('crVResize','LongInt').SetInt( TCursor ( 102 ));
20190:  crDragHandFree','LongInt).SetInt( TCursor ( 103 ));
20191:  crDragHandGrip','LongInt).SetInt( TCursor ( 104 ));
20192:  cCheckBoxFrameSize','LongInt).SetInt( 13);
20193:  cCR','Char).SetString( #13);
20194:  cLF','Char).SetString( #10);
20195:  cTAB','Char).SetString( #9);
20196:  cSPACE','Char).SetString( #32);
20197:  cNULL','Char).SetString( #0);
20198:  {cWordBreaks','LongInt').Value.ts32 := ord(cNULL) or ord(cTAB) or ord(cSPACE);
20199:  cLineBreaks','LongInt').Value.ts32 := ord(cCR) or ord(cLF);
20200:  cEllipsis','String').SetString( '...;
20201:  cEOL','').SetString( cLF);
20202:  cFirstEOL','').SetString( cLF);
20203:  cFirstEOL','').SetString( cCR); }
20204:   TypeS('TLMessage', 'TMessage;
20205:   TypeS('TKkString', 'Widestring;
20206:   TypeS('TKkChar', 'char;
20207:   //TKChar = TUTF8Char;
20208:   TypeS(LONG_PTR2', 'Longint;
20209:   TypeS(TKSysCharSet', 'set of AnsiChar;
20210:   TypeS(TKCurrencyFormat', 'record CurrencyFormat : Byte; CurrencyDecima'
20211:    +ls : Byte; CurrencyString : TKkString; DecimalSep : Char; ThousandSep : Cha'
20212:    +r; UseThousandSep :Bool; end;
20213:  {TypeS('TKAppContext', 'record Application : TApplication; Screen : TScr'
20214:    +'een; GlobalNameSpace : IReadWriteSync; MainThreadID : LongWord; IntConstLi'
20215:    +'st : TThreadList; WidgetSet : TWidgetSet; DragManager : TDragManager; end;  }
20216:  // TypeS('PKAppContext', '^TKAppContext // will not work;
20217:  //TypeS('TKClipboardFormat', 'TClipboardFormat;
20218:   TypeS('TKClipboardFormat', 'Word;
20219:   TypeS('TKCellSpan', 'record ColSpan : Int; RowSpan : Int; end;
20220:  DelphiFunction('Func AdjustDecimalSeparator( const S :Str) :Str;
20221:  Func AnsistrToString( const Text : Ansistr; CodePage :Card) : TKkString;
20222:  //Func BinarySearch(AData:str; ACount:Int;KeyPtr:str;ACompareProc:TBSCompareProc;ASortedDown: Bool):Int;
20223:  Proc CallTrackMouseEvent( Control : TWinControl; var Status :Bool);
20224:  Proc CenterWindowInWindow( CenteredWnd, BoundWnd : HWnd);
20225:  Proc CenterWindowOnScreen( CenteredWnd : HWnd);
20226:  Func CharInSetEx( AChar : AnsiChar; const ASet : TKSysCharSet) :Bool;;
20227:  Func CharInSetEx1( AChar : WideChar; const ASet : TKSysCharSet) :Bool;;
20228:  Func ClipboardLoadStreamAs(const AFormat:str;AStream:TStream;var AText:TKkString):Bool;
20229:  Func ClipboardSaveStreamAs(const AFormat:str;AStream:TStream;const AText:TKkString):Bool;
20230:  Func CompareInts(I1,I2: Int):Int;//Func CompareWideChars(W1,W2:PWideChar;Locale:Card):Int;
20231: // Func CompareChars( S1, S2 : PChar; Locale :Card) : Int;
20232:  Func CompareWideStrings( W1, W2 : WideString; Locale :Card) : Int;
20233: // Func CompareStrings( S1, S2 :Str; Locale :Card) : Int;
20234:  Proc ConvertTabsToSpaces( var AText : TKkString; ASpacesForTab : Int);
20235:  Func CreateMultipleDir( const Dir :Str) :Bool;
20236:  Func DigitToNibble( Digit : AnsiChar; var Nibble : Byte) :Bool;
20237:  Func DivUp( Dividend, Divisor : Int) : Int;
20238:  Func DivDown( Dividend, Divisor : Int) : Int;
20239:  Func EditIsFocused( AMustAllowWrite :Bool) :Bool;
20240:  Func EditFocusedTextCanCopy :Bool;
20241:  Func EditFocusedTextCanCut :Bool;
20242:  Func EditFocusedTextCanDelete :Bool;
20243:  Func EditFocusedTextCanPaste :Bool;
20244:  Func EditFocusedTextCanUndo :Bool;
20245:  Proc EditUndoFocused;
20246:  Proc EditDeleteFocused;
20247:  Proc EditCutFocused;
20248:  Proc EditCopyFocused;
20249:  Proc EditPasteFocused;
20250:  Proc EditSelectAllFocused;
20251:  Proc EnableControls2( AParent : TWinControl; AEnabled, ARecursive :Bool);
20252:  Proc EnsureLastPathSlash( var APath :Str);
20253:  Proc Error2( const Msg :Str);
```

```
20254:  Func FillMessage( Msg :Card; WParam : WPARAM; LParam : LPARAM) : TLMessage;
20255:  Func FormatCurrency( Value : Currency; const AFormat : TKCurrencyFormat) : TKkString;
20256:  //Func GetAppContext( var Ctx : TKAppContext) :Bool;
20257:  Func GetAppVersion(const ALibName:str;var MajorVersion,MinorVersion,BuildNumber,RevisionNumber:Word):Bool;
20258:  Func GetCharCount( const AText : TKkString; AChar : TKkChar) : Int;
20259:  Func GetControlText( Value : TWinControl) : TKkString;
20260:  Func GetFormatSettings2 : TFormatSettings;
20261:  Func GetShiftState : TShiftState;
20262:  Func IntToAscii( Value : Int64; Digits : Int) :Str;
20263:  Func IntToBinStr( Value : Int64; Digits : Byte; const Suffix :Str) :Str;
20264:  Func IntToBCD( Value :Card) :Card;
20265:  Func IntToDecStr( Value : Int64) :Str;
20266:  Func IntToHexStr(Value:Int64;Digits:Byte;const Prefix,Suffix:str;UseLowerCase:Bool):str;
20267:  Func IntToOctStr( Value : Int64) :Str;
20268:  Func IntToRoman2( Value : Int; AUpperCase :Bool) :Str;
20269:  Func IntToLatin( Value : Int; AUpperCase :Bool) :Str;
20270:  Func IntPowerInt( Value : Int64; Exponent : Int) : Int64;
20271:  Func AsciiToInt( S :Str; Digits : Int) : Int64;
20272:  Func BCDToInt( Value :Card) :Card;
20273:  Func BinStrToInt2(S:str; Digits:Byte; Signed :Bool; var Code : Int) : Int64;
20274:  Func DecStrToInt( S :Str; var Code : Int) : Int64;
20275:  Func HexStrToInt(S:Str; Digits: Byte; Signed:Boolean; var Code : Int) : Int64;
20276:  Func OctStrToInt( S :Str; var Code : Int) : Int64;
20277:  Func KFormat14(const Format:str;const Args:array of const;const AFormatSettings:TFormatSettings):str;
20278:  Func KFormat15(const Format:WideString;const Args:array of const;const AFormatSettings:TFormatSettings):
        WideString;
20279:  Func MakeCellSpan( AColumns, ARows : Int) : TKCellSpan;
20280:  Func MinMax16(Value,Min,Max: ShortInt) : ShortInt;;
20281:  Func MinMax17(Value,Min,Max: SmallInt) : SmallInt;;
20282:  Func MinMax18(Value,Min,Max: Int) : Int;;
20283:  Func MinMax19(Value,Min,Max: Int64) : Int64;;
20284:  Func MinMax20(Value,Min,Max: Single) : Single;;
20285:  Func MinMax21(Value,Min,Max: Double) : Double;;
20286:  Func MinMax22(Value,Min,Max: Extended) : Extended;;
20287:  Func NibbleToDigit( Nibble : Byte; UpperCase :Bool) : AnsiChar;
20288:  Proc OpenURLWithShell( const AText : TKkString);
20289:  //Proc QuickSortNR(AData: Pointer;ACount:Int; ACompareProc:TQsCompareProc;AExchangeProc:TQsExchangeProc;
        ASortedDown:Bool);
20290:  //Proc QuickSort(AData:Pointer;ACount:Int;ACompareProc:TQsCompareProc;AExchangeProc:TQsExchangeProc;
        ASortedDown:Bool);
20291:  Proc OffsetPoint23( var APoint : TPoint; AX, AY : Int);;
20292:  Proc OffsetPoint24( var APoint : TPoint; const AOffset : TPoint);;
20293:  Func RectInRect( Bounds, Rect : TRect) :Bool;
20294:  Proc OffsetRect25( var ARect : TRect; AX, AY : Int);;
20295:  Proc OffsetRect26( var ARect : TRect; const AOffset : TPoint);;
20296:  //Func SetAppContext( const Ctx : TKAppContext) :Bool;
20297:  Proc SetControlClipRect( AControl : TWinControl; const ARect : TRect);
20298:  Proc SetControlText( Value : TWinControl; const Text : TKkString);
20299:  Proc StripLastPathSlash( var APath :Str);
20300:  Func StrNextCharIndex( const AText : TKkString; Index : Int) : Int;
20301:  Func StrPreviousCharIndex( const AText : TKkString; Index : Int) : Int;
20302:  Func StringCharBegin( const AText : TKkString; Index : Int) : Int;
20303:  Func StringLength( const AText : TKkString) : Int;
20304:  Func StringCopy( const ASource : TKkString; At, Count : Int) : TKkString;
20305:  Proc StringDelete( var ASource : TKkString; At, Count : Int);
20306:  Proc TrimWhiteSpaces27(const AText:TKkString;var AStart,ALen:Int;const ASet:TKSysCharSet);
20307:  Proc TrimWhiteSpaces28( var AText : TKkString; const ASet : TKSysCharSet);;
20308:  Proc TrimWhiteSpaces29( var AText : Ansistr; const ASet : TKSysCharSet);;
20309:  Func StringToAnsistr( const AText : TKkString; CodePage :Card) : Ansistr;
20310:  Func StringToChar( const AText : TKkString; AIndex : Int) : TKkChar;
20311:  Func GetWindowsFolder2( CSIDL :Card; var APath :Str) :Bool;
20312:  Func RunExecutable( const AFileName :Str; AWaitForIt :Bool) : DWORD;
20313:  Func SystemCodePage : Int;
20314:  Func NativeUTFToUnicode( const AText : TKkString) : WideChar;
20315:  Func UnicodeUpperCase2( const AText : TKkString) : TKkString;
20316:  Func UnicodeLowerCase2( const AText : TKkString) : TKkString;
20317:  Func UnicodeToNativeUTF( const AParam : WideChar) : TKkString;
20318: Func UnicodeStringReplace(const AText,AOldPattern,ANewPattern:TKkString;AFlags:TReplaceFlags):TKkString;
20319:  end;
20320:
20321: Proc SIRegister_KMessageBox(CL: TPSPascalCompiler);
20322: begin
20323: TypeS('TKMsgBoxButton',(mbYes,mbNo,mbOK,mbCancel,mbClose,mbAbort,mbRetry,mbIgnorembAllmbNoToAll,mbYesToAll,
       mbHelp);
20324:   TypeS('TKMsgBoxIcon', ( miNone, miInformation, miQuestion, miWarning, miStop );
20325:  Func CreateMsgBox(const Caption,Text:str;const Buttons:array of TKMsgBoxButton;Icon:TKMsgBoxIcon;Def
        Int):TCustomForm;
20326:  Func CreateMsgBoxEx(const Caption,Text:str;const Btns:array of
        string;Icon:TKMsgBoxIcon;Def:Int):TCustomForm;
20327:  Proc FreeMsgBox( AMsgBox : TCustomForm);
20328:  Func KMsgBox(const Caption,Text:str;const Buttons:array of TKMsgBoxButton;Icon:TKMsgBoxIcon;Def:Int):Int;
20329:  Func KMsgBoxEx(const Caption,Text:str;const Buttons:array of string;Icon:TKMsgBoxIcon;Def:Int):Int;
20330:  Func KInputBox( const Caption, Prompt :Str; var Text :Str) : TModalResult;
20331:  //Func KNumberInputBox(const ACaption,APrompt:str;var AValue:double;AMin,AMax:double;
        AFormats:TKNumberEditAcceptedFormats):TModalResult;
20332:  TKMsgBoxButtons','(mbAbortRetryIgnore,mbOkOnly,mbOkCancel,mbRetryCancel,mbYesNo,mbYesNoCancel);
20333:  Func MsgBox2(const Caption,Txt:str;const Buttons:TKMsgBoxButtons;Icon:TKMsgBoxIcon):Int;
20334:  Func AppMsgBox(const Caption, Text :Str; Flags: Int) : Int;
20335: end;
```

```
20336:
20337: Proc SIRegister_Kronos(CL: TPSPascalCompiler);
20338: begin
20339:  'ChurchDayCount','LongInt').SetInt( 21);
20340:  ConstantN('CommonDayCount','LongInt').SetInt( 4);
20341:  'chAdvent1','LongInt').SetInt( 1);
20342:  'chAdvent2','LongInt').SetInt( 2);
20343:  'chAdvent3','LongInt').SetInt( 3);
20344:  'chAdvent4','LongInt').SetInt( 4);
20345:  'chChristmasEve','LongInt').SetInt( 5);
20346:  'chChristmasDay','LongInt').SetInt( 6);
20347:  'chBoxingDay','LongInt').SetInt( 7);
20348:  'chNewYearEve','LongInt').SetInt( 8);
20349:  'chNewYearDay','LongInt').SetInt( 9);
20350:  'chShroveTuesday','LongInt').SetInt( 10);
20351:  'chAshWednesday','LongInt').SetInt( 11);
20352:  'chPalmSunday','LongInt').SetInt( 12);
20353:  'chMaundyThursday','LongInt').SetInt( 13);
20354:  'chGoodFriday','LongInt').SetInt( 14);
20355:  'chEasterEve','LongInt').SetInt( 15);
20356:  'chEasterSunday','LongInt').SetInt( 16);
20357:  'chEasterMonday','LongInt').SetInt( 17);
20358:  'chWhitEve','LongInt').SetInt( 18);
20359:  'chWhitSunday','LongInt').SetInt( 19);
20360:  'chWhitMonday','LongInt').SetInt( 20);
20361:  'chAscensionDay','LongInt').SetInt( 21);
20362:  'coUNDay','LongInt').SetInt( 22);
20363:  'coWomensDay','LongInt').SetInt( 23);
20364:  'coMayDay','LongInt').SetInt( 24);
20365:  'coLiteracyDay','LongInt').SetInt( 25);
20366:  'UserDayType','LongInt').SetInt( 26);
20367:  //UserDayType = ChurchDayCount + CommonDayCount + 1;
20368:    TFirstLastNumber', 'array[1..2] of word;;
20369:    TDaytypeID', 'array[1..255] of word;;
20370:    TMonthImage2', 'array[0..7] of smallint;;
20371:    TMonthImage', 'array[1..6] of tmonthimage2;;
20372:     //TMonthImage = array[1..6, 0..7] of smallint;
20373:   // TFirstLastNumber = array[1..2] of word;
20374:    // TDaytypeID = array[1..255] of word;
20375:   TDay','record Daynum : Word; MonthDate : word; DOWNum : word; M'
20376:    +'onth : word; Week : word; DayCode : Word; end;
20377:   TypeS('TWeek', 'record WeekNum : word; WhichDays : TFirstLastNumber; end;
20378:   TMonth', 'record Month : word; Daycount : Word; WeekCount : Word'
20379:    +'; WhichWeeks : TFirstLastNumber; WhichDays : TFirstLastNumber; end;
20380:   TYear', 'record WeekCount : word; DayCount : Word; end;
20381:   TKron', 'record ActiveYear : Word; IsInitialized :Bool; end;
20382:   TYearExt', 'record Year : word; NumDays : word; NumWeeks : word;'
20383:    + LeapYear :Bool; YeartypeCount : word; end;
20384:   TDateExt', 'record Year : word; DayOfWeekNumber : word; DayName '
20385:    +:Str; MonthDay : Word; DayNumber : word; DaytypeCount: word; DaytypeI'
20386:    +D : TDaytypeID; MonthNumber : word; WeekNumber : word; Holiday : bool; '
20387:    +ChurchDay : Bool; Flagday :Bool; end;
20388:   TMonthExt', 'record Year : word; MonthNumber : word; MonthName :'
20389:    +' string; FirstDay : word; LastDay : word; NumDays : word; NumWeeks : word;'
20390:    +' FirstWeek : word; LastWeek : word; MonthImage : TMonthImage; end;
20391:   TWeekExt', 'record Year: word; WeekNumber: word; FirstDay : word; LastDay : word; end;
20392:   TForeignKey','record KeyName :Str; KeyValue : Variant; end;
20393:   TDaytypeDef','record AName :Str; ADate : word; ARelDayTyp'
20394:    +e : word; AnOffset : Int; AFirstShowUp : word; ALastShowUp : word; ASh'
20395:    +owUpFrequency : word; AChurchDay :Bool; AHoliday :Bool; AFlagday :'
20396:    + bool; AUserCalc :Bool; ATag : Int; end;
20397:   SIRegister_TDaytype(CL);
20398:   TWeekDay', '( Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday );
20399:   TWeekHolidays', 'set of TWeekDay;
20400:   TOcEVent', '( ocYear, ocMonth, ocMonthnumber, ocWeek, ocWeeknumb'
20401:    +'er, ocMonthDay, ocWeekday, ocDate, ocToday, ocCalcDaytype );
20402:   TCalcDaytypeEvent', 'Proc ( Sender : TObject; Daytype : TDa'
20403:    +'ytype; ADateExt : TDateExt; IsCurrentDate :Bool; var Accept :Bool);
20404:   TLoadDaytypeEvent', 'Proc ( Sender : TObject; const Daytype'
20405:    +'Def : TDaytypeDef; const DescKeys :Str; ClassId : Int; var LoadIt: bool);
20406:   TSaveDaytypeEvent', 'Proc ( Sender : TObject; Daytype : TDa'
20407:    +'ytype; var DescKeys :Str; var ClassID : Int; var SaveIt :Bool);
20408:   SIRegister_TKronos(CL);
20409:   ClassN(CL.FindClass('TOBJECT'),'EKronosError;
20410:  //DelphiFunction('Proc Register;
20411: end;
20412:
20413: Proc SIRegister_MapFiles(CL: TPSPascalCompiler);
20414: begin
20415:   ClassN(CL.FindClass('TOBJECT'),'EMemoryMappedFile;
20416:   ClassN(CL.FindClass('TOBJECT'),'EMMEndOfFile;
20417: TypeS('TFileOptions,(foCreateNew,foCreateAlways,foOpenExisting,foOpenAlways,foTruncateExisting);
20418:   TypeS('TmapFileType', ( ftUnspecified, ftRandomAccess, ftSequentialAccess);
20419:   SIRegister_TMapStream(CL);
20420:   SIRegister_TTextMap(CL);
20421:   SIRegister_TRecordMap(CL);
20422: end;
20423:
20424: Proc SIRegister_KGraphics(CL: TPSPascalCompiler);
```

```
20425: begin
20426:   PNGHeader','String').SetString(' #137 ''PNG'' #13#10#26#10;
20427:   MNGHeader','String').SetString( '#138 ''MNG'' #13#10#26#10;
20428:   TKBrightMode', '( bsAbsolute, bsOfBottom, bsOfTop );
20429:   TKColorRec', 'record R : byte; G : byte; B : byte; A : Byte; Value :Card; end;
20430:   //PKColorRec', '^TKColorRec // will not work;
20431:   TKDynColorRecs', 'array of TKColorRec;
20432:   TKImageHeaderString', 'string;
20433:   //TKPngImage', 'TPortableNetworkGraphic;
20434:  with ClassN(CL.FindClass('TLinarGraphic'),'TPNGGraphic') do
20435:       TKPngImage', 'TPNGGraphic;
20436:       TKKString', 'WideString;
20437:    // TKString = WideString;
20438:     //TKPngImage', 'TPngImage;
20439:   //TKPngImage', 'TPngObject;
20440:   TKTextAttribute', '( taCalcRect, taClip, taEndEllipsis, taFillRe'
20441:    +'ct, taFillText, taIncludePadding, taLineBreak, taPathEllipsis, taWordBreak'
20442:    +', taWrapText, taTrimWhiteSpaces, taStartEllipsis );
20443:   TKTextAttributes', 'set of TKTextAttribute;
20444:   TKHAlign', '( halLeft, halCenter, halRight, halJustify );
20445:   TKStretchMode', '( stmNone, stmZoomOutOnly, stmZoomInOnly, stmZoom );
20446:   TKTextHAlign', 'TKHAlign;
20447:   TKVAlign', '( valTop, valCenter, valBottom );
20448:   TKTextVAlign', 'TKVAlign;
20449:   TKButtonDrawState', '( bsUseThemes, bsDisabled, bsPressed, bsFocused, bsHot );
20450:   TKButtonDrawStates', 'set of TKButtonDrawState;
20451:   SIRegister_TKGraphic(CL);
20452:   SIRegister_TKAlphaBitmap(CL);
20453:   SIRegister_TKMetafile(CL);
20454:   TKTextBoxFunction', '( tbfMeasure, tbfGetIndex, tbfGetRect, tbfDraw );
20455:   SIRegister_TKTextBox(CL);
20456:   SIRegister_TKDragWindow(CL);
20457:   SIRegister_TKHintWindow(CL);
20458:   SIRegister_TKTextHint(CL);
20459:   SIRegister_TKGraphicHint(CL);
20460:  //DelphiFunction('Proc BlendLine( Src, Dest : PKColorRecs; Count : Int);
20461:  Func TKBrightColor( Color : TColor; Percent : Single; Mode : TKBrightMode): TColor;
20462:  Proc CanvasGetScale( ACanvas : TCanvas; out MulX, MulY, DivX, DivY : Int);
20463:  Proc CanvasResetScale( ACanvas : TCanvas);
20464:  Func CanvasScaled( ACanvas : TCanvas) :Bool;
20465:  Proc CanvasSetScale( ACanvas : TCanvas; MulX, MulY, DivX, DivY : Int);
20466:  Proc CanvasSetOffset( ACanvas : TCanvas; OfsX, OfsY : Int);
20467:  Func ColorRecToColor( Color : TKColorRec) : TColor;
20468:  Func ColorToColorRec( Color : TColor) : TKColorRec;
20469:  Func TKColorToGrayScale( Color : TColor) : TColor;
20470:  Func CompareBrushes( ABrush1, ABrush2 : TBrush) :Bool;
20471:  Func CompareFonts( AFont1, AFont2 : TFont) :Bool;
20472:  Proc CopyBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcX, SrcY : Int);
20473:  Func CreateEmptyPoint : TPoint;
20474:  Func CreateEmptyRect : TRect;
20475:  Func CreateEmptyRgn : HRGN;
20476:  Proc DrawAlignedText(Canvas: TCanvas; var ARect : TRect; HAlign : TKHAlign; VAlign : TKVAlign; HPadding,
       VPadding:Int;const AText:TKKString;BackColor:TColor;Attributes:TKTextAttributes);
20477:  Proc TKDrawButtonFrame(ACanvs:TCanvas;const ARect:TRect;AStates:TKButtonDrawStates);
20478:  Proc DrawEdges(Canvas:TCanvas;const R:TRect;HighlightColor,ShadowColor:TColor;Flags:Cardi;
20479:  Proc DrawFilledRectangle(Canvas: TCanvas; const ARect : TRect; BackColor : TColor);
20480:  Proc DrawGradientRect(Canvas:TCanvas; const ARect:TRect;AStartColor,AEndColor TColor; AColorStep:
       Int;AHorizontal:Bool);
20481:  Proc ExcludeShapeFromBaseRect(var BaseRect:TRect; ShapeWidth, ShapeHeight : Int; HAlign : TKHAlign;
       VAlign : TKVAlign; HPadding, VPadding : Int; StretchMode : TKStretchMode; out Bounds, Interior : TRect);
20482:  Func ExtSelectClipRect(DC:HDC; ARect:TRect; Mode:Int; var PrevRgn: HRGN):Bool;
20483:  Func ExtSelectClipRectEx(DC:HDC;ARect:TRect;Mode:Int;CurRgn,PrevRgn : HRGN):Bool;
20484:  Proc FillAroundRect(ACanvas:TCanvas;const Boundary,Interior:TRect;BackColor:TColor);
20485:  Func GetFontHeight( DC : HDC) : Int;
20486:  Func GetFontAscent( DC : HDC) : Int;
20487:  Func GetFontDescent( DC : HDC) : Int;
20488:  Func GDICheck( Value : Int) : Int;
20489:  Func HorizontalShapePosition(AAlignment:TKHAlign;const ABoundary:TRect;const AShapeSize:TPoint):Int;
20490:  Func ImageByType( const Header : TKImageHeaderString) : TGraphic;
20491:  Func IntersectClipRectIndirect( DC : HDC; ARect : TRect) :Bool;
20492:  Func IsBrightColor( Color : TColor) :Bool;
20493:  Proc LoadCustomCursor( Cursor : TCursor; const ResName :Str);
20494:  Proc TKLoadGraphicFromResource(Graphic:TGraphic;const ResName:str;ResType:PChar);
20495:  Func MakeColorRec( R, G, B, A : Byte) : TKColorRec;;
20496:  Func MakeColorRec5( Value : LongWord) : TKColorRec;;
20497:  Func PixelFormatFromBpp( Bpp :Card) : TPixelFormat;
20498:  Func TKRectInRegion( Rgn : HRGN; ARect : TRect) :Bool;
20499:  Func RgnCreateAndGet( DC : HDC) : HRGN;
20500:  Proc RgnSelectAndDelete( DC : HDC; Rgn : HRGN);
20501:  Proc RoundRectangle(ACanvas: TCanvas; const ARect: TRect;AXRadius,AYRadius: Int);
20502:  Proc SafeStretchDraw(ACanvas:TCanvas; ARect:TRect;AGraphic:TGraphic;ABackColor:TColor);
20503:  Proc SelectClipRect( DC : HDC; const ARect : TRect);
20504:  Proc TKStretchBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcRect : TRect);
20505:  Func SwitchRGBToBGR( Value : TColor) : TColor;
20506:  Proc TranslateRectToDevice( DC : HDC; var ARect : TRect);
20507:  Func VerticalShapePosition(AAlignment:TKVAlign;const ABoundary:TRect;const AShapeSize: TPoint):Int;
20508: end;
20509:
20510: Proc SIRegister_umaxPipes(CL: TPSPascalCompiler);
```

```
20511: begin
20512:  ConstantN('cShutDownMsg','String').SetString( 'shutdown pipe ;
20513:  ConstantN(cPipeFormat',String).SetString( '\\%s\pipe\%s;
20514:   SIRegister_TPipeServermax(CL);
20515:   SIRegister_TPipeClientmax(CL);
20516: end;
20517:
20518: Proc SIRegister_KControls(CL: TPSPascalCompiler);
20519: begin
20520: TypeS('TKColorArray', 'array of TColor;
20521: TKPreviewColorIndex', 'Int;
20522: TKPrintOption', '( poCollate, poFitToPage, poMirrorMargins, poPa'
20523:  +'geNumbers, poPaintSelection, poTitle, poUseColor );
20524: TKPrintOptions', 'set of TKPrintOption;
20525: TKPrintRange', '( kprAll, kprSelectedOnly, kprRange );
20526: TKPrintUnits', '( kpuMM, kpuCM, kpuInch, kpuHundredthInch );
20527:  //ConstantN('cBorderStyleDef','').SetString( bsSingle);
20528:  'cContentPaddingBottomDef','LongInt').SetInt( 0);
20529:  'cContentPaddingLeftDef','LongInt').SetInt( 0);
20530:  ConstantN(cOptionsDef','LongInt).Value.ts32:= ord(poFitToPage) or ord(poPageNumbers) or ord(poUseColor);
20531:  //ConstantN('cRangeDef','').SetString( prAll);
20532:  'cScaleDef','LongInt').SetInt( 100);
20533:  'cScaleMin','LongInt').SetInt( 10);
20534:  'cScaleMax','LongInt').SetInt( 500);
20535:  //ConstantN('cUnitsDef','').SetString( puCM);
20536:  //ConstantN('cPaperDef','').SetString( clWhite);
20537:  //ConstantN('cBkGndDef','').SetString( clAppWorkSpace);
20538:  //ConstantN('cBorderDef','').SetString( clBlack);
20539:  //ConstantN('cSelectedBorderDef','').SetString( clNavy);
20540:  ConstantN('ciPaper','LongInt').SetInt( TKPreviewColorIndex ( 0 ));
20541:  'ciBkGnd','LongInt').SetInt( TKPreviewColorIndex ( 1 ));
20542:  'ciBorder','LongInt').SetInt( TKPreviewColorIndex ( 2 ));
20543:  'ciSelectedBorder','LongInt').SetInt( TKPreviewColorIndex ( 3 ));
20544:  // 'ciPreviewColorsMax','').SetString( ciSelectedBorder);
20545:  'cScrollNoAction','LongInt').SetInt( - 1);
20546:  'cScrollDelta','LongInt').SetInt( - 2);
20547:  'cPF_Dragging','LongWord').SetUInt( $00000001);
20548:  'cPF_UpdateRange','LongWord').SetUInt( $00000002);
20549:   TKPreviewScaleMode', '( smScale, smPageWidth, smWholePage );
20550:   TKPreviewChangedEvent', 'Proc ( Sender : TObject);
20551:   TKPrintMeasureInfo', 'record OutlineWidth : Int; OutlineHeig'
20552:    ht : Int; ControlHorzPageCount : Int; ControlVertPageCount : Integ'
20553:    er; ExtraLeftHorzPageCount : Int; ExtraLeftVertPageCount : Int; Ex'
20554:    traRightHorzPageCount : Int; ExtraRightVertPageCount : Int; end;
20555:   TKPrintStatus', '( kepsBegin, kepsNewPage, kepsEnd );
20556:   TKPrintNotifyEvent,Procedure(Sender:TObject;Status: TKPrintStatus;var Abort:Bool);
20557:   TKPrintPaintEvent', 'Proc ( Sender : TObject);
20558:   ClassN(CL.FindClass('TOBJECT'),TKPrintPageSetup;
20559:   ClassN(CL.FindClass('TOBJECT'),TKPrintPreview;
20560:   SIRegister_TKRect(CL);
20561:   ClassN(CL.FindClass('TOBJECT'),'TKObjectList;
20562:   SIRegister_TKObject(CL);
20563:   //TKObjectClass', 'class of TKObject;
20564:   SIRegister_TKObjectList(CL);
20565:   SIRegister_TKCustomControl(CL);
20566:   TKColorScheme', '( kcsNormal, kcsGrayed, kcsBright, kcsGrayScale );
20567:   TKColorIndex', 'Int;
20568:   TKColorData', 'record Index:TKColorIndex;Color:TColor; Default:TColor; Name:str; end;
20569:   TKColorSpec', 'record Def : TColor; Name :Str; end;
20570:   SIRegister_TKCustomColors(CL);
20571:   TKPrintMeasureEvent', 'Proc ( Sender : TObject; var Info : TKPrintMeasureInfo);
20572:   SIRegister_TKPrintPageSetup(CL);
20573:   SIRegister_TKPreviewColors(CL);
20574:   SIRegister_TKPrintPreview(CL);
20575:  Func InchesToValue(Units : TKPrintUnits; Value : Double) : Double;
20576:  Func ValueToInches(Units : TKPrintUnits; Value : Double) : Double;
20577: end;
20578:
20579: Proc SIRegister_IdAntiFreezeBase(CL: TPSPascalCompiler);
20580: begin
20581:  ConstantN(ID_Default_TIdAntiFreezeBase_Active,Boolean).SetInt(1);
20582:  ConstantN('ID_Default_TIdAntiFreezeBase_ApplicationHasPriority','Boolean').SetInt(1);
20583:  ConstantN('ID_Default_TIdAntiFreezeBase_IdleTimeOut','LongInt').SetInt( 250);
20584:  ConstantN('ID_Default_TIdAntiFreezeBase_OnlyWhenIdle','Boolean').SetInt( 250);
20585:  SIRegister_TIdAntiFreezeBase(CL);
20586: end;
20587:
20588: Proc SIRegister_OverbyteIcsConApp(CL: TPSPascalCompiler);
20589: begin
20590:  ConstantN('WM_STARTUP','LongInt').SetInt( WM_USER + 789);
20591:  SIRegister_TKeyboardThread(CL);
20592:  //TypeS('TConApplicationClass', 'class of TConApplication;
20593:  SIRegister_TConApplication(CL);
20594: end;
20595:
20596: Proc SIRegister_KMemo(CL: TPSPascalCompiler);
20597: begin
20598:  'cUndoLimitMin','LongInt').SetInt( 100);
20599:  cUndoLimitMax','LongInt).SetInt( 10000);
```

```
20600:  cUndoLimitDef','LongInt).SetInt( 1000);
20601:  cScrollPaddingMin','LongInt).SetInt( 0);
20602:  cScrollPaddingMax','LongInt).SetInt( 1000);
20603:  cScrollPaddingDef','LongInt).SetInt( 30);
20604:  cScrollSpeedMin','LongInt).SetInt( 50);
20605:  cScrollSpeedMax','LongInt).SetInt( 1000);
20606:  cScrollSpeedDef','LongInt).SetInt( 100);
20607:  //cBkGndDef','').SetString( clWindow);
20608:  //cInactiveCaretBkGndDef','').SetString( clBlack);
20609:  //cInactiveCaretSelBkGndDef','').SetString( clBlack);
20610:  //cInactiveCaretSelTextDef','').SetString( clYellow);
20611:  //cInactiveCaretTextDef','').SetString( clYellow);
20612:  //cSelBkGndDef','').SetString( clGrayText);
20613:  //cSelBkGndFocusedDef','').SetString( clHighlight);
20614:  //cSelTextDef','').SetString( clHighlightText);
20615:  //cSelTextFocusedDef','').SetString( clHighlightText);
20616:  ciBkGnd','LongInt).SetInt( TKColorIndex ( 0 ));
20617:  ciInactiveCaretBkGnd','LongInt).SetInt( TKColorIndex ( 1 ));
20618:  ciInactiveCaretSelBkGnd','LongInt).SetInt( TKColorIndex ( 2 ));
20619:  ciInactiveCaretSelText','LongInt).SetInt( TKColorIndex ( 3 ));
20620:  ciInactiveCaretText','LongInt).SetInt( TKColorIndex ( 4 ));
20621:  ciSelBkGnd','LongInt).SetInt( TKColorIndex ( 5 ));
20622:  ciSelBkGndFocused','LongInt).SetInt( TKColorIndex ( 6 ));
20623:  ciSelText','LongInt).SetInt( TKColorIndex ( 7 ));
20624:  ciSelTextFocused','LongInt).SetInt( TKColorIndex ( 8 ));
20625:  //ciMemoColorsMax','').SetString( ciSelTextFocused);
20626:  cInvalidListID','LongInt).SetInt( - 1);
20627:  cHorzScrollStepDef',LongInt').SetInt( 4);
20628:  cVertScrollStepDef',LongInt').SetInt( 10);
20629:  //cHeight','LongInt').SetInt( 200);
20630:  //cWidth','LongInt').SetInt( 300);
20631:  cNewLineChar','Char).SetString( #$B6);
20632:  cSpaceChar','Char).SetString( #$B7);
20633:  cTabChar','Char).SetString( #$2192);
20634:  cBullet','Char).SetString( #$2022);
20635:  cSquareBullet,Char').SetString( #$25AB);
20636:  cArrowBullet',Char').SetString( #$25BA);
20637:  //cDefaultWordBreaks,String').SetString(' ' or  '/' or '\' or ';' or  ':' or  '?' or  '!';
20638:  cRichText','String').SetString( 'Rich Text Format;
20639:  ClassN(CL.FindClass(TOBJECT'),'TKCustomMemo;
20640:  TypeS('TKMemoLinePosition', '( eolInside, eolEnd );
20641:  TKMemoBlockPosition', '( mbpText, mbpRelative, mbpAbsolute );
20642:  TKMemoState', '( elCaretCreated, elCaretVisible, elCaretUpdate, '
20643:    +elIgnoreNextChar, elModified, elMouseCapture, elOverwrite, elReadOnly );
20644:   TKMemoStates', 'set of TKMemoState;
20645:   TKMemoUpdateReason', '( muContent, muExtent, muSelection, muSelectionScroll );
20646:   TKMemoUpdateReasons', 'set of TKMemoUpdateReason;
20647:   SIRegister_TKMemoSparseItem(CL);
20648:   SIRegister_TKMemoSparseList(CL);
20649:   SIRegister_TKMemoSparseStack(CL);
20650:   SIRegister_TKMemoDictionaryItem(CL);
20651:   SIRegister_TKMemoDictionary(CL);
20652:   TKMemoParaNumbering', '( pnuNone, pnuBullets, pnuArabic, pnuLett'
20653:    +erLo, pnuLetterHi, pnuRomanLo, pnuRomanHi );
20654:   SIRegister_TKMemoNumberingFormatItem(CL);
20655:   SIRegister_TKMemoNumberingFormat(CL);
20656:   ClassN(CL.FindClass('TOBJECT'),'TKMemoListLevels;
20657:   SIRegister_TKMemoListLevel(CL);
20658:   ClassN(CL.FindClass('TOBJECT'),'TKMemoList;
20659:   SIRegister_TKMemoListLevels(CL);
20660:   ClassN(CL.FindClass('TOBJECT'),'TKMemoListTable;
20661:   SIRegister_TKMemoList(CL);
20662:   TKMemoListChangedEvent', 'Proc (AList: TKMemoList; ALevel: TKMemoListLevel);
20663:   SIRegister_TKMemoListTable(CL);
20664:   TKMemoScriptCapitals', '( tcaNone, tcaNormal, tcaSmall );
20665:   TKMemoScriptPosition', '( tpoNormal, tpoSuperscript, tpoSubscript );
20666:   SIRegister_TKMemoTextStyle(CL);
20667:   TKMemoBlockWrapMode', '( wrAround, wrAroundLeft, wrAroundRight, '
20668:    +wrTight, wrTightLeft, wrTightRight, wrTopBottom, wrNone, wrUnknown );
20669:   TKMemoBlockStyleChangedEvent', 'Proc ( Sender : TObject; AR'
20670:    +'easons : TKMemoUpdateReasons);
20671:   SIRegister_TKMemoBlockStyle(CL);
20672:   TKMemoLineSpacingMode', '( lsmFactor, lsmValue );
20673:   SIRegister_TKMemoParaStyle(CL);
20674:   SIRegister_TKMemoLine(CL);
20675:   SIRegister_TKMemoLines(CL);
20676:   SIRegister_TKMemoWord(CL);
20677:   SIRegister_TKMemoWordList(CL);
20678:   ClassN(CL.FindClass('TOBJECT'),'TKMemoBlocks;
20679:   TKMemoMouseAction', '( maMove, maLeftDown, maLeftUp, maRightDown'
20680:    +', maRightUp, maMidDown, maMidUp );
20681:   //TKMemoBlockClass', 'class of TKMemoBlock;
20682:   SIRegister_TKMemoBlock(CL);
20683:   SIRegister_TKMemoSingleton(CL);
20684:   SIRegister_TKMemoTextBlock(CL);
20685:   SIRegister_TKMemoHyperlink(CL);
20686:   SIRegister_TKMemoParagraph(CL);
20687:   SIRegister_TKMemoImageBlock(CL);
20688:   SIRegister_TKMemoContainer(CL);
```

```
20689:   ClassN(CL.FindClass('TOBJECT'),'TKMemoTable;
20690:   ClassN(CL.FindClass('TOBJECT'),'TKMemoTableRow;
20691:   SIRegister_TKMemoTableCell(CL);
20692:   SIRegister_TKMemoTableRow(CL);
20693:   SIRegister_TKMemoTable(CL);
20694:   TKMemoUpdateEvent', 'Proc ( Reasons : TKMemoUpdateReasons);
20695:   SIRegister_TKMemoBlocks(CL);
20696:   SIRegister_TKMemoColors(CL);
20697:   TKMemoChangeKind', '( ckCaretPos, ckDelete, ckInsert );
20698:   TKMemoUndoChangeEvent','Procedure(Sender : TObject; ItemReason: TKMemoChangeKind);
20699:   TKMemoChangeItem', 'record Blocks : TKMemoBlocks; Group : Cardin'
20700:    al; GroupKind : TKMemoChangeKind; Inserted :Bool; ItemKind : TKMemoCha'
20701:    ngeKind; Position : Int; end;
20702:   //PKMemoChangeItem', '^TKMemoChangeItem // will not work;
20703:   SIRegister_TKMemoChangeList(CL);
20704:   TKMemoRTFString', 'string;
20705:   SIRegister_TKCustomMemo(CL);
20706:   SIRegister_TKMemo(CL);
20707:   SIRegister_TKMemoEditAction(CL);
20708:   SIRegister_TKMemoEditCopyAction(CL);
20709:   SIRegister_TKMemoEditCutAction(CL);
20710:   SIRegister_TKMemoEditPasteAction(CL);
20711:   SIRegister_TKMemoEditSelectAllAction(CL);
20712:  Func NewLineChar : TKkString;
20713:  Func SpaceChar : TKkString;
20714:  Func TabChar : TKkString;
20715: end;
20716:
20717: Proc SIRegister_OverbyteIcsTicks64(CL: TPSPascalCompiler);
20718: begin
20719:  ConstantN('ISODateMask',String).SetString( 'yyyy-mm-dd;
20720:  'ISODateTimeMask','String').SetString( 'yyyy-mm-dd"T"hh:nn:ss;
20721:  'ISODateLongTimeMask','String').SetString( 'yyyy-mm-dd"T"hh:nn:ss.zzz;
20722:  'ISOTimeMask','String').SetString( 'hh:nn:ss;
20723:  'LongTimeMask','String').SetString( 'hh:nn:ss:zzz;
20724:  'Ticks64PerDay','int64').SetInt64( 24 * 60 * 60 * 1000);
20725:  'Ticks64PerHour','int64').SetInt64( 60 * 60 * 1000);
20726:  'Ticks64PerMinute','int64').SetInt64( 60 * 1000);
20727:  'Ticks64PerSecond','int64').SetInt64( 1000);
20728:   TypeS('TTicks64Mode', ( TicksNone, TicksAPI64, TicksPerf, TicksAPI32 );
20729:  DelphiFunction(Func IcsGetTickCount64 : int64;
20730:  Proc IcsInitTick64( NewMode : TTicks64Mode);
20731:  Func IcsNowPC : TDateTime;
20732:  Proc IcsAlignNowPC;
20733:  Func IcsLastBootDT : TDateTime;
20734:  Func IcsGetPerfCountsPerSec : int64;
20735:  Func IcsPerfCountCurrent : int64;
20736:  Func IcsPerfCountCurrMilli : int64;
20737:  Func IcsPerfCountToMilli( LI : int64) : int64;
20738:  Func IcsPerfCountGetMilli( startLI : int64) : int64;
20739:  Func IcsPerfCountGetMillStr( startLI : int64) :Str;
20740:  Func IcsPerfCountToSecs( LI : int64) : Int;
20741:  Func IcsPerfCountGetSecs( startLI : int64) : Int;
20742:  Func IcsDiffTicks64( const StartTick, EndTick : int64) : int64;
20743:  Func IcsElapsedTicks64( const StartTick : int64) : int64;
20744:  Func IcsElapsedMsecs64( const StartTick : int64) : int64;
20745:  Func IcsElapsedSecs64( const StartTick : int64) : Int;
20746:  Func IcsElapsedMins64( const StartTick : int64) : Int;
20747:  Func IcsWaitingSecs64( const EndTick : int64) : Int;
20748:  Func IcsGetTrgMSecs64( const MilliSecs : int64) : int64;
20749:  Func IcsGetTrgSecs64( const DurSecs : Int) : int64;
20750:  Func IcsGetTrgMins64( const DurMins : Int) : int64;
20751:  Func IcsTestTrgTick64( const TrgTick : int64) :Bool;
20752:  Func IcsAddTrgMsecs64( const TickCount, MilliSecs : int64) : int64;
20753:  Func IcsAddTrgSecs64( const TickCount : int64; DurSecs : Int) : int64;
20754: end;
20755:
20756: Proc SIRegister_OverbyteIcsSha1(CL: TPSPascalCompiler);
20757: begin
20758:  ConstantN(IcsSHA1Version','LongInt.SetInt( 800);
20759:  CopyRight','String.SetString(  IcsSHA1 (c) 2004-2012 F. Piette V8.00 ;
20760:  shaSuccess','LongInt.SetInt( 0);
20761:  shaNull','LongInt.SetInt( 1);
20762:  shaInputTooLong','LongInt.SetInt( 2);
20763:  shaStateError','LongInt.SetInt( 3);
20764:  SHA1HashSize','LongInt.SetInt( 20);
20765:   TypeS('uint32_t', 'LongWord;
20766:   TypeS('uint8_t', 'Byte;
20767:   TypeS('int_least16_t', 'LongInt;
20768:   TypeS('SHA1DigestString', 'Ansistr;
20769:  Func SHA1Reset( var context : SHA1Context) : Int;
20770: Func SHA1Input(var context:SHA1Context;const message_array:str;message_array:PAnsiChar;length
       Cardinal):Int;
20771:  Func SHA1Result( var context : SHA1Context; var Message_Digest : SHA1Digest) : Int;
20772:  Func SHA1ofStr( const s : Ansistr) : SHA1DigestString;
20773:  Func SHA1ofBuf( const buf, buflen : Int) : SHA1DigestString;
20774:  Func SHA1ofStream( const strm : TStream) : SHA1DigestString;
20775:  Func SHA1toHex( const digest : SHA1DigestString) :Str;
20776:  Func SHA1DigestToLowerHex( const Digest : SHA1Digest) :Str;
```

```
20777:  Func SHA1DigestToLowerHexA( const Digest : SHA1Digest) : RawByteString;
20778:  Func SHA1DigestToLowerHexW( const Digest : SHA1Digest) : UnicodeString;
20779:  Proc HMAC_SHA1(const Data,DataLen:Int;const Key,KeyLen:Int;out Digest:SHA1Digest);
20780:  Func HMAC_SHA1_EX( const Data : Ansistr; const Key : Ansistr) : Ansistr;
20781: end;
20782:
20783: Proc SIRegister_KEditCommon(CL: TPSPascalCompiler);
20784: begin
20785:  cCharMappingSize','LongInt.SetInt( 256);
20786:  TKEditCommandk', '( ecNonek, ecLeftk, ecRightk, ecUpk, ecDownk, ecLinek'
20787:   Start, ecLineEndk, ecPageUpk, ecPageDownk, ecPageLeftk, ecPageRightk, ecPageTopk,
20788:    htk, ecSelUpk, ecSelDownk, ecSelLineStartk, ecSelLineEndk, ecSelPageUpk, ecSelPa'
20789:    geDownk, ecSelPageLeftk, ecSelPageRightk, ecSelPageTopk, ecSelPageBottomk, ecSe'
20790:    lEditorTopk, ecSelEditorBottomk, ecSelGotoXYk, ecScrollUpk, ecScrollDownk, ecSc'
20791:    rollLeftk, ecScrollRightk, ecScrollCenterk, ecUndok, ecRedok, ecCopyk, ecCutk, ec'
20792:    +Pastek, ecInsertChark, ecInsertDigitsk, ecInsertStringk, ecInsertNewLinek, ecDe'
20793:    +leteLastChark, ecDeleteChark, ecDeleteBOLk, ecDeleteEOLk, ecDeleteLinek, ecSele'
20794:    +ctAllk, ecClearAllk, ecClearIndexSelectionk, ecClearSelectionk, ecSearchk, ecRe'
20795:    +'placek, ecInsertModek, ecOverwriteModek, ecToggleModek, ecGotFocusk, ecLostFocusk ); *)
20796:   TKEditDisabledDrawStyle', '( eddBright, eddGrayed, eddNormal );
20797:   TKEditKey', 'record Key : Word; Shift : TShiftState; end;
20798:   TKEditCommandAssignment', 'record Key : TKEditKey; Command : TKEditCommand; end;
20799:   TKEditCommandMap', 'array of TKEditCommandAssignment;
20800:   TKEditDropFilesEvent','Procedure(Sender:TObject;X,Y:Int;Files:TStrings);
20801:   SIRegister_TKEditKeyMapping(CL);
20802:   TKEditCharMapping', 'array of AnsiChar;
20803:   //TypeS('PKEditCharMapping', '^TKEditCharMapping // will not work;
20804:   //TypeS('TKEditOption', '( eoDropFiles, eoGroupUndo, eoUndoAfterSave, eoS'
20805:    //+'howFormatting, eoWantTab );
20806:   //TypeS('TKEditOptions', 'set of TKEditOption;
20807:   TKEditReplaceAction', '( eraCancel, eraYes, eraNo, eraAll );
20808:   TKEditReplaceTextEvent', 'Proc ( Sender : TObject; const Te'
20809:    xtToFind, TextToReplace :Str; var Action : TKEditReplaceAction);
20810:   TKEditSearchError', '( eseOk, eseNoDigitsFind, eseNoDigitsReplace, eseNoMatch );
20811:   TKEditSearchOption', '( esoAll, esoBackwards, esoEntireScope, es'
20812:    oFirstSearch,esoMatchCase,esoPrompt,esoSelectedOnly,esoTreatAsDigits,esoWereDigits );
20813:   TKEditSearchOptions', 'set of TKEditSearchOption;
20814:   TKEditSearchData', 'record ErrorReason : TKEditSearchError; Opti'
20815:    ons:TKEditSearchOptions;SelStart:Int;SelEnd:Int;TextToFind:str;TextToReplace:str; end;
20816:  //TypeS('PKEditSearchData', '^TKEditSearchData // will not work;
20817:  //cEditDisabledDrawStyleDef','.SetString( eddBright);
20818:  Func CreateDefaultKeyMapping : TKEditKeyMapping;
20819:  Func DefaultCharMapping : TKEditCharMapping;
20820:  Func DefaultSearchData : TKEditSearchData;
20821: end;
20822:
20823: unit UtilsMax4; unit uPSI_UtilsMax4;
20824:  Func AllDigitsDifferent(N: Int64):Bool;
20825:  Proc DecimalToFraction(Decimal: Extended; out FractionNumerator: Extended;
20826:            out FractionDenominator: Extended; const AccuracyFactor: Extended);
20827:  Func ColorToHTML(const Color: TColor):Str;
20828:  Func DOSCommand(const CommandLine:Str; const CmdShow:Int;
20829:             const WaitUntilComplete:Bool; const WorkingDir:Str = ':Bool;
20830:  Func GetDosOutput(CommandLine:Str; Work:Str = 'C:\:Str;
20831:  Ex.: writeln(GetDosOutput('java -version','C:\'));
20832:       >>>java version "1.8.0_211"
20833:          Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
20834:          Java HotSpot(TM) Client VM (build 25.211-b12, mixed mode)
20835:  Proc CaptureConsoleOutput(DosApp :Str;AMemo : TMemo);
20836:  Ex.: CaptureConsoleOutput('cmd /C dir *.* ',memo2);
20837:       CaptureConsoleOutput('powershell /C dir *.* ',memo2);
20838:  Func ExecuteCommandDOS(CommandLine:str):str;
20839:  Func DOSCommandRedirect(const CommandLine:Str;
20840:            const OutStream: Classes.TStream):Bool; overload;      //8
20841: Proc SendKeysToWindow(const HWnd: Windows.HWND; const Text:Str);
20842: Func IsRunningOnBattery:Bool;
20843: Func IsHexStr(const S:Str):Bool;
20844: Func IsCharInSet(const Ch: Char; const Chars: TCharSet):Bool;
20845: Func StreamHasWatermark(const Stm:Classes.TStream;const Watermark:array of Byte):Bool;
20846: Func ReadBigEndianWord(Stm: Classes.TStream): Word;
20847: Func DownloadURLToFile(const URL, FileName:Str):Bool;
20848: Func ExtractURIQueryString(const URI:Str):Str;
20849: Func GetBiosVendor:Str;
20850: Func GetIEVersionStr:Str;          //18
20851: Func FloatToFixed(const Value: Extended; const DecimalPlaces: Byte;
20852:                              const SeparateThousands:Bool):Str;
20853: Func IntToFixed(const Value: Int;const SeparateThousands:Bool):Str;
20854: Func Int64ToFixed(const Value: Int64;const SeparateThousands:Bool):Str;
20855: Func IntToNumberText2(const Value: Int):Str;      //22
20856:
20857: Func IsLibraryInstalled2(const LibFileName:Str):Bool;
20858: Func RemainingBatteryPercent: Int;
20859: Proc SetLockKeyState(KeyCode: Int; IsOn:Bool);
20860: Func IsLockKeyOn(const KeyCode: Int):Bool;
20861: Proc PostKeyEx322(const Key: Word; const Shift: Classes.TShiftState;
20862:                              const SpecialKey:Bool = False);
20863: Func TerminateProcessByID(ProcessID:Card):Bool;
20864: Func GetWindowProcessName(const Wnd: Windows.HWND):Str;
20865: Func GetProcessName(const PID: Windows.DWORD):Str;
```

```
20866: Func GetWindowProcessID(const Wnd: Windows.HWND): Windows.DWORD;
20867: Func IsAppResponding(const Wnd: Windows.HWND):Bool;
20868: Func IsTabletOS:Bool;
20869: Func ProgIDInstalled(const PID:Str):Bool;
20870: Func GetProcessorName:Str;
20871: Func GetProcessorIdentifier:Str;                    //36
20872: Proc EmptyKeyQueue;
20873: Proc TrimAppMemorySize;
20874: Func GetEnvironmentBlockSize:Card;
20875: Func GetDefaultPrinterName:Str;                     //40
20876: Proc ListDrives(const List: Classes.TStrings);
20877: Proc MultiSzToStrings(const MultiSz: PChar; const Strings: Classes.TStrings);
20878: Func BrowseURL(const URL:Str):Bool;
20879: Func IsValidURLProtocol(const URL:Str):Bool;
20880: Func ExecAssociatedApp(const FileName:Str):Bool;    //45
20881: Func CheckInternetConnection(AHost: PAnsiChar):Bool;
20882: Func MakeSafeHTMLText(TheText:Str):Str;
20883: Func RemoveURIQueryString(const URI:Str):Str;
20884: Func GetRegistryString(const RootKey: Windows.HKEY;const SubKey,Name:str):str;
20885: Proc RefreshEnvironment2(const Timeout:Card = 5000);  //50
20886: Func IsKeyPressed2(const VirtKeyCode: Int):Bool;
20887: Func SizeOfFile64(const FileName:Str): Int64;
20888: Func IsHugeFile(const FileName:Str):Bool;
20889:  Func SetTransparencyLevel(const HWnd: Windows.HWND;const Level: Byte):Bool;
20890: Func IsEqualResID(const R1, R2: PChar):Bool;          //55
20891: Func GetGenericFileType(const FileNameOrExt:Str):Str;
20892: Func GetFileType2(const FilePath:Str):Str;
20893: Proc ShowShellPropertiesDlg(const APath:Str);
20894: Func EllipsifyText(const AsPath:Bool; const Text:Str;
20895:          const Canvas: Graphics.TCanvas; const MaxWidth: Int ):Str;
20896: Func CloneByteArray(const B: array of Byte): TBytes;       //60
20897: Proc AppendByteArray(var B1: TBytes; const B2: array of Byte);
20898: Func IsUnicodeStream(const Stm: Classes.TStream):Bool;
20899: Func FileHasWatermark(const FileName:Str;
20900:             const Watermark: array of Byte; const Offset: Int = 0):Bool;
20901: Func FileHasWatermarkAnsi(const FileName:Str;
20902:            const Watermark: Ansistr; const Offset: Int = 0):Bool;
20903: Func IsASCIIStream(const Stm: Classes.TStream; Count:Int64 = 0;
20904:                              BufSize: Int = 8*1024):Bool;
20905: Func IsASCIIFile(const FileName:str; BytesToCheck:Int64 = 0;
20906:                             BufSize: Int = 8*1024):Bool;      //66
20907: Func BytesToAnsistr(const Bytes: SysUtils.TBytes; const CodePage: Word):str;
20908: Func UnicodeStreamToWideString(const Stm: Classes.TStream): WideString;
20909: Proc WideStringToUnicodeStream(const Str: WideString;const Stm: Classes.TStream);
20910: Proc GraphicToBitmap(const Src: Graphics.TGraphic;
20911:           const Dest: Graphics.TBitmap; const TransparentColor: Graphics.TColor);  //70
20912: Func URIDecode(S:Str; const IsQueryString:Bool):Str;
20913: Func URIEncode(const S:Str; const InQueryString:Bool):Str;
20914: Func URLDecode(const S:Str):Str;
20915: Func URLEncode(const S:Str; const InQueryString:Bool):Str;
20916: Func AllDigitsSame(N: Int64):Bool;
20917: Func GetDosOutput(CommandLine:Str; Work:Str = 'C:\:Str;
20918: Proc CaptureConsoleOutput(DosApp :Str;AMemo : TMemo);
20919:  Ex.: CaptureConsoleOutput('cmd /C dir *.* ',memo2);
20920:      CaptureConsoleOutput('powershell /C dir *.* ',memo2);
20921: Func ExecuteCommandDOS(CommandLine:str):str;
20922: Func DOSCommandRedirect(const CommandLine:Str;
20923:          const OutStream: Classes.TStream):Bool; overload;     //8
20924: Proc SendKeysToWindow(const HWnd: Windows.HWND; const Text:Str);  //75
20925:  Func FoldWrapText(const Line,BreakStr:str;BreakChars:TSysCharSet;MaxCol:Int):str;;
20926:  Func TextWrap(const Text:Str; const Width, Margin: Int):Str;
20927:  Func ShellExecuteX(Operation,FileName,Parameters,Directory:str;ShowCmd:Int):Card;;
20928:  Func KeyboardStateToShiftState1(const KeyboardState: TKeyboardState): TShiftState;;
20929:  Proc SetCustomFormGlassFrame(const CustomForm:TCustomForm;const GlassFrame:TGlassFrame);
20930:  Func GetCustomFormGlassFrame( const CustomForm : TCustomForm) : TGlassFrame;
20931:  Proc SetApplicationMainFormOnTaskBar(const Application:TApplication;Value:Bool);
20932:  Func GetApplicationMainFormOnTaskBar( const Application : TApplication) :Bool;
20933:  Func CompressWhiteSpace(const S:Str):Str;
20934:  Func IsASCIIDigit(const Ch: Char):Bool;
20935:   Func CompareNumberStr(const S1, S2:Str): Int;
20936:  Proc HexToBuf(HexStr:Str; var Buf:Str);
20937:   Func BufToHex(const Buf:Str; const Size:Card):Str;
20938:  Func DOSExec( CommandLine :Str; Work :Str) :Str;
20939:  Proc HexToStrBuf(HexStr:Str; var Buf:Str);;
20940:  Func StrBufToHex(const Buf:Str; const Size:Card):Str;;
20941:  Func GetCharFromVirtualKey(AKey: Word):Str;
20942:  Func GetParentProcessID(const PID: Windows.DWORD): Windows.DWORD;
20943:  Func FormInstanceCount2(AFormClass: Forms.TFormClass): Int; //overload;
20944:  Func FormInstanceCount(const AFormClassName:Str): Int; //overload;
20945:  Func FindAssociatedApp(const Doc:Str):Str;
20946:  Func CreateShellLink2(const LinkFileName,AssocFileName,Desc,WorkDir,Args,IconFileName:str;const
       IconIdx:Int):Bool;
20947: Func FileFromShellLink(const LinkFileName:Str):Str;
20948: Func IsShellLink(const LinkFileName:Str):Bool;
20949: Func ResourceIDToStr(const ResID: PChar):Str;
20950: Func IsEqualResID(const R1, R2: PChar):Bool;
20951: Func RecycleBinInfo(const Drive: Char; out BinSize, FileCount: Int64):Bool;
20952: Func FormInstanceCount2(AFormClass: TFormClass): Int;;
20953: Func FormInstanceCount(const AFormClassName:Str): Int;;
```

```
20954: Func FindAssociatedApp(const Doc:Str):Str;;
20955: Func CreateShellLink2(const LinkFileName,AssocFileName,Desc,WorkDir,Args
20956:    ,IconFileName:Str; const IconIdx: Int):Bool;
20957: Func FileFromShellLink(const LinkFileName:Str):Str;
20958: Func IsShellLink(const LinkFileName:Str):Bool;
20959: Func ResourceIDToStr(const ResID: PChar):Str;
20960: Func IsEqualResID(const R1, R2: PChar):Bool;
20961: Func RecycleBinInfo(const Drive: Char; out BinSize, FileCount: Int64):Bool;
20962: Func SysImageListHandle(const Path:str; const WantLargeIcons:Boolean): THandle;
20963: Func EmptyRecycleBin:Bool;
20964: Func ExploreFile(const Filename:Str ):Bool;
20965: Func ExploreFolder(const Folder:Str):Bool;
20966: Proc ClearRecentDocs2;
20967: Proc AddToRecentDocs2(const FileName:Str);                    //100
20968: Func StringsToMultiSz(const Strings:TStrings;const MultiSz:PChar;const BufSize:Int):Int;
20969: Proc DrawTextOutline(const Canvas: TCanvas;const X,Y:Int;const Text:str);
20970: Func CloneGraphicAsBitmap(const Src:Graphics.TGraphic; const PixelFmt:Graphics.TPixelFormat;
20971: const TransparentColor: Graphics.TColor): Graphics.TBitmap;
20972: Proc InvertBitmap(const ABitmap: Graphics.TBitmap); //overload;
20973: Proc InvertBitmap2(const SrcBmp, DestBmp: Graphics.TBitmap); //overload;
20974: Func MakeFilenameInExePath( aFilename : TFilename) : TFilename;
20975: Func YearOfDate2( DateTime : TDateTime) : Int;
20976: Func MonthOfDate2( DateTime : TDateTime) : Int;
20977: Func DayOfDate2( DateTime : TDateTime) : Int;
20978: Func HourOfTime2( DateTime : TDateTime) : Int;
20979: Func MinuteOfTime2( DateTime : TDateTime) : Int;
20980: Func SecondOfTime2( DateTime : TDateTime) : Int;
20981: Func IsLeapYear2( DateTime : TDateTime) :Bool;
20982: Func DaysInMonth2( DateTime : TDateTime) : Int;
20983: Func MakeUTCTime( DateTime : TDateTime) : TDateTime;
20984: Func MakeLocalTimeFromUTC( DateTime : TDateTime) : TDateTime;
20985: Func IsStandardTime :Bool;
20986: Func UnixNow : Int64;
20987: Func NowString :Str;
20988: Func MakeClosedTag( aTagName, aTagValue :Str) :Str;
20989: Func MakeOpenTag( aTagName, aTagAttributes :Str) :Str;
20990: Func MakeBold( Str :Str) :Str;
20991: Func MakeItalic( Str :Str) :Str;
20992: Func MakeUnderline( Str :Str) :Str;
20993: Func MakeStrikeout( Str :Str) :Str;
20994: Func MakeCenter( Str :Str) :Str;
20995: Func MakeParagraph( Str :Str) :Str;
20996: Func MakeCode( Str :Str) :Str;
20997: Func MakeOption( aValue, aText :Str) :Str;
20998: Func MakeHTMLFontSize( Str :Str; SizeParam :Str) :Str;
20999: Func AddQuotes2( Str :Str) :Str;
21000: Func AddSingleQuotes( Str :Str) :Str;
21001: Func MakeHTMLParam( Str :Str) :Str;
21002: Func MakeLink( URL, name :Str) :Str;
21003: Func MakeLinkTarget( URL, name, Target :Str) :Str;
21004: Func MakeMailTo( Address, name :Str) :Str;
21005: Func HTMLToDelphiColor( S :Str) : TColor;
21006: Func ColorToHTMLHex( Color : TColor) :Str;
21007: Func GetStringFromRes( ResName :Str) :Str;
21008: Func EscapeText( sText :Str) :Str;
21009: Func EncodeForXML( const aString :Str) :Str;
21010: Func IsStringAlpha( Str :Str) :Bool;
21011: Func IsStringNumber( Str :Str) :Bool;
21012: Func EnsurePrefix( aPrefix, aText :Str) :Str;
21013: Func StringToAcceptableChars( S :Str; AcceptableChars : TCharSet) :Str;
21014: Func StringIsAcceptable( S :Str; AcceptableChars : TCharSet) :Bool;
21015: Func ValidateEMailAddress( aEmail :Str) :Bool;
21016: Func FirstChar( Str :Str) : Char;
21017: Func LastChar( Str :Str) : Char;
21018: Func StringIsEmpty( Str :Str) :Bool;
21019: Func StringIsNotEmpty( Str :Str) :Bool;
21020: Func StringHasSpacesInMiddle( Str :Str) :Bool;
21021: Func StringContains( aString :Str; aSubStr :Str) :Bool;
21022: Func SpacesToUnderscore( S :Str) :Str;
21023: Func SpacesToPluses( Str :Str) :Str;
21024: Func SwapString( Str :Str) :Str;
21025: Func MakeCopyrightNotice( aCopyrightHolder :Str) :Str;
21026: Proc WriteStringToFile( aStr :Str; aFilename : TFilename);
21027: //Func WinExecandWait32( Path : PChar; Visibility : Word) : Int;
21028: Func WinExecAndWait32V2( Filename :Str; Visibility : Int) : DWORD;
21029: Func WindowsExit( RebootParam : Longword) :Bool;
21030: Func GetVersionInfo2 :Str;
21031: Func VersionString( aPrefix :Str; aUseColon :Bool) :Str;
21032: Func CreateTempFileName( aPrefix :Str) :Str;
21033: Func GetWindowsTempDir :Str;
21034: Func GetWindowsDir2 :Str;
21035: Func GetSystemDir2 :Str;
21036: Func GetSpecialFolderLocation( aFolderType : Int) :Str;
21037: Func GetTextFileContents( aFilename : TFilename) :Str;
21038: Func GetBDSDir( aVersion : Int) :Str;
21039: Func CaptionMessageDlg(const aCaption:str;const Msg:str;DlgType:TMsgDlgType;Buttons
       TMsgDlgButtons;HelpCtx:Longint):Int;
21040: Func StreamFileCopy(const SourceFilename,TargetFilename:str;KeepDate:Boolean):Int;
21041: Func MakePercentString( f : Double):Str;
```

```
21042: Proc DumpKey( var aKey : Char);
21043: Func ValidateKey2(var aKey:Char; AcceptableKeys:TCharSet;KillTheKey:Bool) :Bool;
21044: Func MakeInterbaseString( aHostName, aFilename :Str) :Str;
21045: Func ParseToken( const S :Str; var FromPos : Int; Delimiter : Char):Str;
21046: Proc MatchBounds( MovedControl, TemplateControl : TControl);
21047: Proc LockWidth( aControl : TControl);
21048: Proc LockHeight( aControl : TControl);
21049: Proc LockBounds( aControl : TControl);
21050: Func TruncateFilename(aCanvas:TCanvas;aRect:TRect;aFilename:str;aMargin Int):Str;
21051: Func PointInRect2( const R : TRect; const P : TPoint) :Bool;;
21052: Func PointInRect3( const R : TRect; const X, Y : Int) :Bool;;
21053: Proc VariantToStream2( const V : OLEVariant; Stream : TStream);
21054: Proc StreamToVariant2( Stream : TStream; var V : OLEVariant);
21055: Proc AssignDocument( Browser : TWebBrowser; Text :Str);
21056: Proc LoadStreamToWebBrowser( WebBrowser : TWebBrowser; Stream : TStream);
21057: Proc SaveWebBrowserSourceToStream( Document : IDispatch; Stream : TStream);
21058: Proc GetStylesFromBrowser( aBrowser : TWebBrowser; aStyles : TStrings);
21059: TypeS('TnxBits', 'Int; TypeS('TDayHours', 'Int;
21060:  ClassN(CL.FindClass('TOBJECT,'NixUtilsException;
21061: Func REG_CURRENT_VERSION :Str;
21062: Func RegisteredOwner :Str;
21063: Func RegisteredCompany :Str;
21064: Func GetLocalComputerName2 :Str;
21065: Func GetLocalUserName2 :Str;
21066: Func DeleteToRecycleBin(WindowHandle: HWND; Filename:str; Confirm:Bool): Bool;
21067: Func RemoveBackSlash2( const Dir :Str) :Str;
21068: Func EnsureBackSlash( aPath :Str) :Str;
21069: Func EnsureForwardSlash( aPath :Str) :Str;
21070: Func RemoveLeadingSlash( aPath :Str) :Str;
21071: Func SameDirectories( aDir1, aDir2 : TFilename) :Bool;
21072: Func EnsureExtensionHasLeadingPeriod( aExtension :Str) :Str;
21073: Func RemoveExtension( aFilename :Str) :Str;
21074: Func GetModuleFileNameStr :Str;
21075: Func ModulePath :Str;
21076: Func IniFileName :Str;
21077: Func MakeFilenameInExePath( aFilename : TFilename) : TFilename;
21078: Func IsBitSet( Bits : Int; BitToSet : TnxBits) :Bool;
21079: Func SetBit6( Bits : Int; BitToSet : TnxBits) : Int;
21080: Func UnSetBit( Bits : Int; BitToSet : TnxBits) : Int;
21081: Func FlipBit( Bits : Int; BitToSet : TnxBits) : Int;
21082: Func / FontToOleFont
21083: Func / OleFontToFont
21084: Func GetCachedFileFromURL(strUL:Str; var strLocalFile:Str):Bool;
21085: Func IAddrToHostName(const IP:Str):Str;
21086: Func GetIEHandle(WebBrowser: TWebbrowser; ClassName:Str): HWND;
21087: Func GetTextFromHandle(WinHandle: THandle):Str;
21088: Proc Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
21089: Func FillWebForm(WebBrowser:TWebBrowser;FieldName:Str; Value:Str):Bool;
21090: Proc WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode:Str);
21091: Func NetSend(dest, Source, Msg:Str): Longint; overload;
21092: end;
21093: Proc SIRegister_UWANTUtils(CL: TPSPascalCompiler);
21094: begin
21095: TErrorSeverity,(errwantNone,errwantInfo,errwantHint,errwantWarning,errwantError,errwantFatal );
21096:  Func ErrTag( Sev : TErrorSeverity) :Str;
21097:  Func MakeInt64( LowInt32, HiInt32 :Card) : Int64;
21098:  Func FolderContent( const APath :Str) : INode;
21099:  Func FindFiles2(const AFolder,AMask:Str;Attrs:Int;AList:TStrings;ASearchSubdirs:Bool):Bool;
21100:  Func ErrorSeverityStr( ASeverity : TErrorSeverity) :Str;
21101:  Func IsParameterOptional( AParamDef : INode) :Bool;
21102:  Func ExtractMethodName( const S :Str) :Str;
21103:  Func LineNoOf( ANode : INode) : Int;
21104:  Func ColNoOf( ANode : INode) : Int;
21105:  Func WANTBoolToStr( AValue :Bool) :Str;
21106:  Func WANTStrToBool( const AValue :Str; dEFAULT :Bool) :Bool;
21107:  Func StrDefault( const S, Default :Str) :Str;
21108:  Func PluralNoun( Count : Int; const S :Str) :Str;
21109:  Func IsUnqualifiedName( const AName :Str) :Bool;
21110:  Func IsModuleTag( const ATag :Str) :Bool;
21111:  Func IsCallableTag( const ATag :Str) :Bool;
21112:  Func IsConditionalTag( const ATag :Str) :Bool;
21113:  Func IsMethodTag( const ATag :Str) :Bool;
21114:  Func StripDriveFromPath( const AFileName :Str) :Str;
21115:  Func TotalTime( NumTicks : Int64) :Str;
21116: end;
21117:
21118: Proc SIRegister_IdNNTPServer(CL: TPSPascalCompiler);
21119: begin
21120:   TIdNNTPAuthType', '( atUserPass, atSimple, atGeneric );
21121:   TIdNNTPAuthTypes', 'set of TIdNNTPAuthType;
21122:   SIRegister_TIdNNTPThread(CL);
21123:   TypeS('TIdNNTPOnAuth', Proc ( AThread : TIdNNTPThread; var VAccept :Bool);
21124:   TIdNNTPOnNewGroupsList','Procedure(AThread:TIdNNTPThread;const ADateStamp:TDateTime;const
      ADistributions:str);
21125:   TIdNNTPOnNewNews', 'Proc ( AThread : TIdNNTPThread; const Nributions :Str);
21126:   TIdNNTPOnIHaveCheck', 'Procedure(AThread:TIdNNTPThread;const AMsgID String; VAccept : Bool);
21127:   TIdNNTPOnArticleByNo', 'Proc ( AThread : TIdNNTPThread; const AMsgNo : Int);
21128:   TIdNNTPOnArticleByID', 'Proc ( AThread : TIdNNTPThread; const AMsgID :Str);
21129:   TIdNNTPOnCheckMsgNo', 'Procedure(AThread:TIdNNTPThread;const AMsgNo:Int; var VMsgID :Str);
```

```
21130:   TIdNNTPOnCheckMsgID','Proc (AThread:TIdNNTPThread; const AMsgId:str;var VMsgNo : Int);
21131:   TIdNNTPOnMovePointer', 'Proc (AThread TIdNNTPThread;var AMsgNo:Int;var VMsgID:str);
21132:   TIdNNTPOnPost','Procedure(AThread:TIdNNTPThread;var VPostOk:Boolean;var VErrorText :Str);
21133: TIdNNTPOnSelectGroup','Procedure(AThread:TIdNNTPThread;const AGroup:str;var VMsgCount:Int;var
       VMsgFirst:Int;var
21134:    +VMsgLast : Int; var VGroupExists :Bool);
21135:   TIdNNTPOnCheckListGroup', 'Proc ( AThread : TIdNNTPThread; '
21136:    +'const AGroup :Str; var VCanJoin :Bool; var VFirstArticle : Int);
21137:   TIdNNTPOnXHdr', 'Proc ( AThread : TIdNNTPThread; const AHea'
21138:    +derName:Str; const AMsgFirst: Int; const AMsgLast:Int;const AMsgID:str);
21139:   TIdNNTPOnXOver', 'Proc (AThread:TIdNNTPThread;const AMsgFirst:Int;const AMsgLast : Int);
21140:   TIdNNTPOnXPat','Procedure(AThread:TIdNNTPThread;const AHeaderName:str;const AMsgFirst:Int; const
       AMsgLast:Int;const AMsgID:str;const AHeaderPattern:str);
21141: TIdNNTPOnAuthRequired','Procedure(AThread:TIdNNTPThread;const ACommand,AParams:str;var VRequired:Bool);
21142: TIdNNTPOnListPattern', 'Proc (AThread:TIdNNTPThread;const AGroupPattern:str);
21143:   SIRegister_TIdNNTPServer(CL);
21144: end;
21145:
21146: Proc SIRegister_OverbyteIcsAsn1Utils(CL: TPSPascalCompiler);
21147: begin
21148:  'ASN1UtilsVersion','LongInt.SetInt( 101);
21149:  ConstantN('ASN1_BOOL','LongWord.SetUInt( $01);
21150:  'ASN1_INT','LongWord.SetUInt( $02);
21151:  'ASN1_OCTSTR','LongWord').SetUInt( $04);
21152:  'ASN1_NULL','LongWord').SetUInt( $05);
21153:  'ASN1_OBJID','LongWord').SetUInt( $06);
21154:  'ASN1_ENUM','LongWord').SetUInt( $0a);
21155:  'ASN1_SEQ','LongWord').SetUInt( $30);
21156:  'ASN1_SETOF','LongWord').SetUInt( $31);
21157:  'ASN1_IPADDR','LongWord').SetUInt( $40);
21158:  'ASN1_COUNTER','LongWord').SetUInt( $41);
21159:  'ASN1_GAUGE','LongWord').SetUInt( $42);
21160:  'ASN1_TIMETICKS','LongWord').SetUInt( $43);
21161:  'ASN1_OPAQUE','LongWord').SetUInt( $44);
21162:  Func ASNEncOIDItem2( Value : Int) : Ansistr;
21163:  Func ASNDecOIDItem2( var Start : Int; const Buffer : Ansistr) : Int;
21164:  Func ASNEncLen2( Len : Int) : Ansistr;
21165:  Func ASNDecLen2( var Start : Int; const Buffer : Ansistr) : Int;
21166:  Func ASNEncInt2( Value : Int) : Ansistr;
21167:  Func ASNEncUInt2(Value : Int) : Ansistr;
21168:  Func ASNObject2( const Data : Ansistr; ASNType : Int) : Ansistr;
21169:  Func ASNItem2( var Start:Int; const Buffer:Ansistr; var ValueType:Int):Ansistr;
21170:  Func MibToId2( Mib :Str) : Ansistr;;
21171:  //Func MibToId1( Mib : Ansistr) : Ansistr;;
21172:  Func IdToMib2( const Id : Ansistr) :Str;
21173:  Func IntMibToStr2( const Value : Ansistr) : Ansistr;
21174:  Func ASNdump2( const Value : Ansistr) :Str;
21175: end;
21176: const  SM_MEDIACENTER = 87; //metrics flag not defined in Windows unit
21177:        SM_TABLETPC = 86;
21178:   http://snippets.delphidabbler.com/#
21179:
21180: Proc SIRegister_wmiserv(CL: TPSPascalCompiler);
21181: begin
21182:  ConstantN('EOAC_NONE','LongInt').SetInt( 0);
21183:  'RPC_C_AUTHN_WINNT','LongInt').SetInt( 10);
21184:  'RPC_C_AUTHZ_NONE','LongInt').SetInt( 0);
21185:  'RPC_E_CHANGED_MODE','LongInt').SetInt( - 2147417850);
21186:  Func WMIStart : ISWBemLocator;
21187:  Func WMIConnect(WBemLocator:ISWBemLocator;Server,account,password:str):ISWBemServices;
21188:  Func WMIExecQuery( WBemServices : ISWBemServices; query :Str) : ISWbemObjectSet;
21189: Func WMIRowFindFirst(ObjectSet:ISWbemObjectSet;var ENum:IEnumVariant;var tempobj:OleVariant):bool;
21190:  Func WMIRowFindNext( ENum : IENumVariant; var tempobj : OleVariant) :Bool;
21191:  Func WMIColFindFirst(var propENum:IENumVariant; var tempObj : OleVariant):Bool;
21192:  Func WMIColFindNext( propENum : IENumVariant; var tempobj : OleVariant) :Bool;
21193:  Func WMIGetValue(wbemservices:ISWBemServices; tablename,fieldname:str):Str;
21194:  Func WMIConvValue( tempobj : OleVariant; var keyname :Str):Str;
21195:  --- wmiserv2:
21196:  Func WMIRegConnect(WBemLocator:ISWBemLocator;Server,account,password:str): ISWBemServices;
21197:  Proc WMIGetMethodInfo(srv:ISWbemServices;objname,method:str; var regobject,inparms: ISWBemObject);
21198:  Proc WMISetValue(InParam: ISWBemObject; keyvalue:Str; invalue: OleVariant);
21199: end;
21200:
21201: Proc SIRegister_RegSvrUtils(CL: TPSPascalCompiler);
21202: begin
21203:  SIRegister_ERegistryException(CL);
21204:  Func RegOpenKey2( Key : HKey; const SubKey :Str) : HKey;
21205:  Func RegGetKey( Key : HKey; const SubKey :Str) :Str;
21206:  Func RegCanOpenKey( Key : HKey; const SubKey :Str; var OutKey : HKey) :Bool;
21207:  Func RegKeyExists2( Key : HKey; const SubKey :Str) :Bool;
21208:  Func RegCloseAndNilKey( var Key : HKey) :Bool;
21209:  Func RegQuerySubKeyCount( Key : HKey) : Int;
21210:  Func RegEnumKey2( Key : HKey; Index : Int; var Value :Str) :Bool;
21211:  Func RegQueryKey( Key : HKey; const SubKey :Str; var Value :Str) :Bool;
21212:  Func RegGetDefaultValue( Key : HKey) :Str;
21213:  Func RegGetValueEx( Key : HKey; const ValName :Str) :Str;
21214:  Proc ErrorFmt( const Ident :Str; const Args : array of const);
21215:  Proc FmtError( const Ident :Str; const Args : array of const);
21216: end;
```

```
21217:
21218: Proc SIRegister_osFileUtil(CL: TPSPascalCompiler);  //Linux
21219: begin
21220:  Func OsDOS2UnixFileAttributes( Attr : LongWord) : LongWord;
21221:  Func OsUnix2DosFileAttributes( Attr : LongWord) : LongWord;
21222:  Func OsUnixFileTimeToDateTime( UnixTime : LongInt) : TDateTime;
21223:  Func OsDateTimeToUnixFileTime( DateTime : TDateTime) : LongInt;
21224:  Func OsDosFileTimeToDateTime( DosTime : LongInt) : TDateTime;
21225:  Func OsDateTimeToDosFileTime( Value : TDateTime) : LongInt;
21226:  Func OsFileTimeToLocalFileTime( FileTime : LongInt) : LongInt;
21227:  Func OsLocalFileTimeToFileTime( FileTime : LongInt) : LongInt;
21228: end;
21229:
21230: Proc SIRegister_TWebBrowser(CL: TPSPascalCompiler);
21231: begin
21232:   //with RegClassS(CL,'TOleControl', 'TWebBrowser') do
21233:  with ClassN(CL.FindClass('TOleControl'),'TWebBrowser') do begin
21234:   RegisterMethod(Proc GoBack;
21235:  Proc GoForward;
21236:   Proc GoHome;
21237:   Proc GoSearch;
21238:   Proc Navigate7( const URL : WideString);;
21239:   Proc Navigate8( const URL : WideString; const Flags : OleVariant);;
21240:   Proc Navigate9(const URL:WideString;const Flags:OleVariant;const TargetFrameName: OleVariant);;
21241:    Proc Navigate10(const URL:WideString;const Flags:OleVariant;const TargetFrameName:OleVariant;var
       PostData:OleVariant);
21242:   Proc Navigate11(const URL:WideString;const Flags:OleVariant;const TargetFrameName:OleVariant;var
       PostData:OleVariant;const Headers:OleVariant);
21243:   Proc Refresh;
21244:   Proc Refresh212;;
21245:   Proc Refresh213( var Level : OleVariant);;
21246:   Proc Stop;
21247:   Proc Quit;
21248:   Proc ClientToWindow( var pcx : SYSINT; var pcy : SYSINT);
21249:   Proc PutProperty( const Property_ : WideString; vtValue : OleVariant);
21250:   Func GetProperty( const Property_ : WideString) : OleVariant;
21251:   Proc Navigate214( var URL : OleVariant);;
21252:   Proc Navigate215( var URL : OleVariant; var Flags : OleVariant);;
21253:   Proc Navigate216(var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVar);
21254:   Proc Navigate217(var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVariant;var PostData:
       OleVariant);
21255:   Proc Navigate218(var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVar;var
       PostData:OleVariant;var Headers:OleVariant);
21256:  Func QueryStatusWB( cmdID : OLECMDID) : OLECMDF;
21257:  Proc ExecWB19( cmdID : OLECMDID; cmdexecopt : OLECMDEXECOPT);;
21258:  Proc ExecWB20(cmdID:OLECMDID;cmdexecopt: OLECMDEXECOPT; var pvaIn : OleVariant);;
21259:  Proc ExecWB21(cmdID:OLECMDID;cmdexecopt:OLECMDEXECOPT;var pvaIn:OleVariant;var pvaOut:OleVariant);
21260:  Proc ShowBrowserBar22( var pvaClsid : OleVariant);;
21261:  Proc ShowBrowserBar23( var pvaClsid : OleVariant; var pvarShow : OleVariant);;
21262:  Proc ShowBrowserBar24(var pvaClsid:OleVariant;var pvarShow:OleVariant;var pvarSize: OleVariant);;
21263:  ControlInterface', 'IWebBrowser2', iptr);
21264:  RegisterProperty(DefaultInterface, IWebBrowser2', iptr);
21265:  Application', 'IDispatch', iptr);
21266:  Parent', 'IDispatch', iptr);
21267:  Container', 'IDispatch', iptr);
21268:  Document', 'IDispatch', iptr);
21269:  TopLevelContainer', 'WordBool', iptr);
21270:  type_', 'WideString', iptr);
21271:  LocationName', 'WideString', iptr);
21272:  LocationURL', 'WideString', iptr);
21273:  Busy', 'WordBool', iptr);
21274:  Name', 'WideString', iptr);
21275:  FullName', 'WideString', iptr);
21276:  Path', 'WideString', iptr);
21277:  Visible', 'WordBool', iptrw);
21278:  StatusBar', 'WordBool', iptrw);
21279:  StatusText', 'WideString', iptrw);
21280:  ToolBar', 'Int', iptrw);
21281:  MenuBar', 'WordBool', iptrw);
21282:  FullScreen', 'WordBool', iptrw);
21283:  Offline', 'WordBool', iptrw);
21284:  Silent', 'WordBool', iptrw);
21285:  RegisterAsBrowser', 'WordBool', iptrw);
21286:  RegisterAsDropTarget', 'WordBool', iptrw);
21287:  TheaterMode', 'WordBool', iptrw);
21288:  AddressBar', 'WordBool', iptrw);
21289:  Resizable', 'WordBool', iptrw);
21290:  OnStatusTextChange', 'TWebBrowserStatusTextChange', iptrw);
21291:  OnProgressChange', 'TWebBrowserProgressChange', iptrw);
21292:  OnCommandStateChange', 'TWebBrowserCommandStateChange', iptrw);
21293:  OnDownloadBegin', 'TNotifyEvent', iptrw);
21294:  OnDownloadComplete', 'TNotifyEvent', iptrw);
21295:  OnTitleChange', 'TWebBrowserTitleChange', iptrw);
21296:  OnPropertyChange', 'TWebBrowserPropertyChange', iptrw);
21297:  OnBeforeNavigate2', 'TWebBrowserBeforeNavigate2', iptrw);
21298:  OnNewWindow2', 'TWebBrowserNewWindow2', iptrw);
21299:  OnNavigateComplete2', 'TWebBrowserNavigateComplete2', iptrw);
21300:  OnDocumentComplete', 'TWebBrowserDocumentComplete', iptrw);
21301:  OnQuit', 'TNotifyEvent', iptrw);
```

```
21302:  OnVisible', 'TWebBrowserOnVisible', iptrw);
21303:  OnToolBar', 'TWebBrowserOnToolBar', iptrw);
21304:  OnMenuBar', 'TWebBrowserOnMenuBar', iptrw);
21305:  OnStatusBar', 'TWebBrowserOnStatusBar', iptrw);
21306:  OnFullScreen', 'TWebBrowserOnFullScreen', iptrw);
21307:  OnTheaterMode', 'TWebBrowserOnTheaterMode', iptrw);
21308:  OnWindowSetResizable', 'TWebBrowserWindowSetResizable', iptrw);
21309:  OnWindowSetLeft', 'TWebBrowserWindowSetLeft', iptrw);
21310:  OnWindowSetTop', 'TWebBrowserWindowSetTop', iptrw);
21311:  OnWindowSetWidth', 'TWebBrowserWindowSetWidth', iptrw);
21312:  OnWindowSetHeight', 'TWebBrowserWindowSetHeight', iptrw);
21313:  OnWindowClosing', 'TWebBrowserWindowClosing', iptrw);
21314:  OnClientToHostWindow', 'TWebBrowserClientToHostWindow', iptrw);
21315:  OnSetSecureLockIcon', 'TWebBrowserSetSecureLockIcon', iptrw);
21316:  OnFileDownload', 'TWebBrowserFileDownload', iptrw);
21317:  OnNavigateError', 'TWebBrowserNavigateError', iptrw);
21318:  OnPrintTemplateInstantiation', 'TWebBrowserPrintTemplateInstantiation', iptrw);
21319:  OnPrintTemplateTeardown', 'TWebBrowserPrintTemplateTeardown', iptrw);
21320:  OnUpdatePageStatus', 'TWebBrowserUpdatePageStatus', iptrw);
21321:  OnPrivacyImpactedStateChange', 'TWebBrowserPrivacyImpactedStateChange', iptrw);
21322:  end;
21323: end;
21324:
21325: Proc SIRegister_xutils(CL: TPSPascalCompiler);
21326: begin
21327:   //TypeS('PshortString', '^ShortString // will not work;
21328:  'EXIT_DOSERROR','LongInt').SetInt( 2);
21329:  'EXIT_ERROR','LongInt').SetInt( 1);
21330:  'adCmdTxt','LongInt').SetInt($00000001);
21331:  'adExecNoRecords','LongInt').SetInt($00000080);
21332:  {** Byte order mark: UTF-8 encoding signature }
21333:  ('BOM_UTF8','String').SetString( #$EF#$BB#$BF);
21334:  {** Byte order mark: UTF-8 encoding signature }
21335:   BOM_UTF8      = #$EF#$BB#$BF;
21336:   {** Byte order mark: UCS-4 big endian encoding signature }
21337:   BOM_UTF32_BE = #00#00#$FE#$FF;
21338:   {** Byte order mark: UCS-4 little endian encoding signature }
21339:   BOM_UTF32_LE = #$FF#$FE#00#00;
21340:   BOM_UTF16_BE = #$FE#$FF;
21341:   BOM_UTF16_LE = #$FF#$FE;
21342:   //const adCmdTxt = $00000001; //adExecNoRecords = $00000080;
21343:  Func AnsiFileExists( const FName :Str) :Bool;
21344:  Func AnsiDirectoryExists( DName :Str) :Bool;
21345:  Proc SwapLong2( var x : longword);
21346:  Proc SwapWord2( var x : word);
21347:  Func UpString( s :Str) :Str;
21348:  Func LowString( s :Str) :Str;
21349:  Func AddDoubleQuotes( s :Str) :Str;
21350:  Func RemoveDoubleQuotes( s :Str) :Str;
21351:  Proc StreamErrorProcedure( var S : TStream);
21352:  Func StrToken( var Text :Str; Delimiter : Char; UseQuotes:boolean):Str;
21353:  Func StrGetNextLine( var Text :Str) :Str;
21354:  Func TrimLeftx( const S :Str) :Str;
21355:  Func TrimRightx( const S :Str) :Str;
21356:  Func hexstr( val : longint; cnt : byte) :Str;
21357:  Func decstr( val : longint; cnt : byte) :Str;
21358:  Func decstrunsigned( l : longword; cnt : byte) :Str;
21359:  Func boolstr( val :Bool; cnt : byte) :Str;
21360:  Func CompareByte( buf1, buf2 : pchar; len : longint) : Int;
21361:  Func Trimx( const S :Str) :Str;
21362:  Func Printf2( const s :Str; var Buf:Str; size : word) :Str;
21363:  Func Printfx( const s :Str; var Buf:Str; size : word) :Str;
21364:  Func FillTo( s :Str; tolength : Int) :Str;
21365:  Func stringdup( const s :Str) :Str;
21366:  //Proc stringdispose( var p : pShortstring);
21367:  Func EscapeToPascal( const s :Str; var code : Int) :Str;
21368:  Func ValDecimal( const S :Str; var code : Int) : longint;
21369:  Func ValUnsignedDecimal( const S :Str; var code : Int) : longword;
21370:  Func ValOctal( const S :Str; var code : Int) : longint;
21371:  Func ValBinary( const S :Str; var code : Int) : longint;
21372:  Func ValHexadecimal( const S :Str; var code : Int) : longint;
21373:  Func CleanString( const s :Str) :Str;
21374:  Func ChangeFileExt2( const FileName, Extension :Str) :Str;
21375:  Func fillwithzero( s :Str; newlength : Int) :Str;
21376:  Func removenulls( const s :Str) :Str;
21377:  //ConstantN('WhiteSpace','Char').SetString( ' ' or  #10 or  #13 or  #9);
21378: end;
21379:
21380: Proc SIRegister_ietf(CL: TPSPascalCompiler);
21381: begin
21382:  Func mime_isvalidcontenttype( const s : shortstring) :Bool;
21383:  Func langtag_isvalid( const s :Str) :Bool;
21384:  Func langtag_split( const s :Str; var primary, sub :Str) :Bool;
21385:  URI_START_DELIMITER_CHAR','String').SetString( '<;
21386:  URI_END_DELIMITER_CHAR','String').SetString( '>;
21387:  URI_SCHEME_NAME_EMAIL','String').SetString( 'mailto;
21388:  URI_SCHEME_SEPARATOR','String').SetString( ':;
21389:  Func uri_split( url :Str; var scheme, authority, path, query :Str) :Bool;
21390:  Func urn_isvalid( s : shortstring) :Bool;
```

```
21391:  Func urn_isvalidnid( nid :Str) :Bool;
21392:  Func urn_split( urn :Str; var urnidstr, nidstr, nssstr :Str) :Bool;
21393:  Func urn_pathsplit( path :Str; var namespace, nss :Str) :Bool;
21394:  Func http_pathsplit( path :Str; var directory, name :Str) :Bool;
21395:  Func file_pathsplit( path :Str; var directory, name :Str) :Bool;
21396: end;
21397:
21398: Proc SIRegister_dateutilreal(CL: TPSPascalCompiler);
21399: begin
21400:   TypeS('TDatetimeReal', 'real;
21401:   TypeS(TDateInfo', 'record DateTime : TDateTime; UTC :Bool; end;
21402:   //TypeS('float', 'real;
21403:    TypeS('big_Int_t', 'int64;
21404:  'DayMondayR','LongInt').SetInt( 1);
21405:  ConstantN('DayTuesdayR','LongInt').SetInt( 2);
21406:  'DayWednesdayR','LongInt').SetInt( 3);
21407:  'DayThursdayR','LongInt').SetInt( 4);
21408:  'DayFridayR','LongInt').SetInt( 5);
21409:  'DaySaturdayR','LongInt').SetInt( 6);
21410:  'DaySundayR','LongInt').SetInt( 7);
21411:  Func CurrentYearreal : word;
21412:  Func Datereal : TDatetimeReal;
21413:  Func DateOfreal( const AValue : TDatetimeReal) : TDatetimeReal;
21414:  Func DateTimeToStrreal( DateTime : TDatetimeReal) :Str;
21415:  Func DateToStrreal( date : TDatetimeReal) :Str;
21416:  Func DayOfreal( const AValue : TDatetimeReal) : Word;
21417:  Func DaysBetweenreal( const ANow, AThen : TDatetimeReal) : Int;
21418:  Proc DecodeDatereal( Date : TDatetimeReal; var Year, Month, Day : Word);
21419:  Proc DecodeDateTimereal(const AValue:TDatetimeReal;var Year,Month,Day,Hour,Minute,Second,
       MilliSecond:Word);
21420:  Proc DecodeTimereal( Time : TDatetimeReal; var Hour, Min, Sec, MSec : Word);
21421:  Func HourOfreal( const AValue : TDatetimeReal) : Word;
21422:  Func IncDayreal(const AValue:TDatetimeReal;const ANumberOfDays:Int): TDatetimeReal;
21423:  Func IncHourreal(const AValue:TDatetimeReal;const ANumberOfHours:longint):TDatetimeReal;
21424:  Func IncMilliSecondreal(const AValue:TDatetimeReal; const ANumberOfMilliSeconds: big_Int_t):TDatetimeReal;
21425:  Func IncMinutereal(const AValue: TDatetimeReal; const ANumberOfMinutes: big_Int_t) : TDatetimeReal;
21426:  Func IncSecondreal(const AValue: TDatetimeReal; const ANumberOfSeconds: big_Int_t) : TDatetimeReal;
21427:  Func IncWeekreal(const AValue:TDatetimeReal;const ANumberOfWeeks:Int): TDatetimeReal;
21428:  Func IsPMreal( const AValue : TDatetimeReal) :Bool;
21429:  Func IsValidDatereal( const AYear, AMonth, ADay : Word) :Bool;
21430: Func IsValidDateTimereal(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSec:Word):Bool;
21431:  Func IsValidTimereal( const AHour, AMinute, ASecond, AMilliSecond : Word) :Bool;
21432:  Func MinuteOfreal( const AValue : TDatetimeReal) : Word;
21433:  Func MonthOfreal( const AValue : TDatetimeReal) : Word;
21434:  Func Nowreal : TDatetimeReal;
21435:  Func SameDatereal( const A, B : TDatetimeReal) :Bool;
21436:  Func SameDateTimereal( const A, B : TDatetimeReal) :Bool;
21437:  Func SameTimereal( const A, B : TDatetimeReal) :Bool;
21438:  Func SecondOfreal( const AValue : TDatetimeReal) : Word;
21439:  Func Timereal : TDatetimeReal;
21440:  Func GetTimereal : TDatetimeReal;
21441:  Func TimeOfreal( const AValue : TDatetimeReal) : TDatetimeReal;
21442:  Func TimeToStrreal( Time : TDatetimeReal) :Str;
21443:  Func Todayreal : TDatetimeReal;
21444:  Func TryEncodeDatereal( Year, Month, Day : Word; var Date : TDatetimeReal) :Bool;
21445:  Func TryEncodeTimereal(Hour,Min,Sec MSec: Word; var Time : TDatetimeReal) :Bool;
21446:  Func TryEncodeDateTimereal(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond:Word;var
       AValue:TDatetimeReal):Boolean;
21447:  Func TryStrToDatereal( const S :Str; var Value : TDatetimeReal) :Bool;
21448:  Func TryStrToDateTimereal( const S :Str; var Value : TDatetimeReal) :Bool;
21449:  Func TryStrToTimereal( const S :Str; var Value : TDatetimeReal) :Bool;
21450:  Func YearOfreal( const AValue : TDatetimeReal) : Word;
21451:  Func parsetimeISO(timestr:str;var hourval,minval,secval:word;var offsethourval,offsetminval:Int;var
       UTC:bool):bool;
21452:  Func parsedateISO( datestr :Str; var yearval, monthval, dayval : word) :Bool;
21453:  Func datetojd( year, month, day, hour, minute, second, millisecond : word) : float;
21454:  Proc jdtodate(jday:float; var year, month, day, hour, minute, second, msec : word);
21455:  Func converttoisotime( timestr :Str) :Str;
21456:  Func AdobeDateToISODate( s :Str) :Str;
21457:  Func RFC822ToISODateTime( s :Str) :Str;
21458:  Proc getdatedos( var year, month, mday, wday : word);
21459:  Proc gettimedos( var hour, minute, second, sec100 : word);
21460: end;
21461:
21462: Proc SIRegister_dateext4(CL: TPSPascalCompiler);
21463: begin
21464:  TypeS('tfiletime2', 'record LowDateTime: longword; HighDateTime: longword; end;
21465:  // TypeS('big_Int_t', 'int64;
21466:  Func DosToWinTime( DTime : longint; var Wtime : TFileTime) : longbool;
21467:  Func WinToDosTime( const Wtime : TFileTime; var DTime : longint) : longbool;
21468:  Func TryStrToDateTimeExt(const S:str;var Value:TDateTime;var UTC:boolean):Bool);
21469:  Func TryEncodeDateAndTimeToStr(const Year,Month,Day,Hour,Minute,Second,MilliSecond:word;UTC:boolean;var
       AValue:str):boolean;
21470:  Func DateTimeToStrExt( DateTime : TDateTime; utc :Bool) :Str;
21471:  Proc GetCurrentDate( var Year, Month, Day, DayOfWeek : Int);
21472:  Proc GetCurrentTime2( var Hour, Minute, Second, Sec100 : Int);
21473:  Func TryUNIXToDateTimeExt(unixtime:big_Int_t;var DateTime:TDateTime;var UTC:bool): bool;
21474:  Func TryFileTimeToDateTimeExt( ftime : tfiletime; var DateTime : TDateTime; var UTC :Bool): bool;
21475:  Proc JulianToGregorian(JulianDN:big_Int_t;var Year,Month,Day:Word);;
```

```
21476: end;
21477:
21478:
21479: Proc SIRegister_locale(CL: TPSPascalCompiler);
21480: begin
21481:  Func GetISODateString( Year, Month, Day : Word) :Str;
21482:  Func GetISODateStringBasic( Year, Month, Day : Word) :Str;
21483:  Func IsValidISODateString( datestr : shortstring; strict :Bool) :Bool;
21484:  Func IsValidISODateStringExt(datestr:shortstring;strict:bool;var Year,Month,Day:word):bool;
21485:  Func IsValidISOTimeString( timestr : shortstring; strict :Bool) :Bool;
21486:  Func IsValidISOTimeStringExt(timestr:shortstring;strict:boolean;var hour,min,sec:word;var offhour,offmin:
       smallint):bool;
21487:  Func IsValidISODateTimeString( str : shortstring; strict :Bool) :Bool;
21488:  Func GetISOTimeString( Hour, Minute, Second : Word; UTC :Bool) : shortstring;
21489:  Func GetISOTimeStringBasic(Hour,Minute,Second: Word; UTC :Bool) : shortstring;
21490:  Func GetISODateTimeString(Year,Month,Day,Hour,Minute,Second:Word;UTC:Bool):shortstring;
21491:  Proc UNIXToDateTime2(epoch:big_Int_t; var year,month,day,hour,minute,second: Word);
21492:  Func GetCharEncoding( alias :Str; var _name :Str) : Int;
21493:  Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
21494:  Func MicrosoftLangageCodeToISOCode( langcode : Int) :Str;
21495:  CHAR_ENCODING_UTF8','LongInt').SetInt( 0);
21496:  ConstantN(CHAR_ENCODING_UNKNOWN',LongInt').SetInt( - 1);
21497:  'CHAR_ENCODING_UTF32BE','LongInt').SetInt( 1);
21498:  'CHAR_ENCODING_UTF32LE','LongInt').SetInt( 2);
21499:  'CHAR_ENCODING_UTF16LE','LongInt').SetInt( 3);
21500:  'CHAR_ENCODING_UTF16BE','LongInt').SetInt( 4);
21501:  'CHAR_ENCODING_BYTE','LongInt').SetInt( 5);
21502:  'CHAR_ENCODING_UTF16','LongInt').SetInt( 6);
21503:  'CHAR_ENCODING_UTF32','LongInt').SetInt( 7);
21504: end;
21505:
21506: Proc SIRegister_Strings(CL: TPSPascalCompiler);
21507: begin
21508:  DelphiFunction('Func StrLenPchar( Str : PChar) : longint;
21509:  Func StrEndPchar( Str : PChar) : PChar;
21510:  Func StrMovePchar( Dest, Source : Pchar; l : Longint) : pchar;
21511:  Func StrCopyPchar( Dest, Source : PChar) : PChar;
21512:  Func StrECopyPchar( Dest, Source : PChar) : PChar;
21513:  Func StrLCopyPchar( Dest, Source : PChar; MaxLen : Longint) : PChar;
21514:  Func StrPCopyPchar( Dest : PChar; Source :Str) : PChar;
21515:  Func StrCatPchar( Dest, Source : PChar) : PChar;
21516:  Func strlcatPchar( dest, source : pchar; l : Longint) : pchar;
21517:  Func StrCompPchar( Str1, Str2 : PChar) : Int;
21518:  Func StrICompPchar( Str1, Str2 : PChar) : Int;
21519:  Func StrLCompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Int;
21520:  Func StrLICompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Int;
21521:  Func StrScanPchar( Str : PChar; Ch : Char) : PChar;
21522:  Func StrRScanPchar( Str : PChar; Ch : Char) : PChar;
21523:  Func StrPosPchar( Str1, Str2 : PChar) : PChar;
21524:  Func StrUpperPchar( Str : PChar) : PChar;
21525:  Func StrLowerPchar( Str : PChar) : PChar;
21526:  Func StrPasPchar( Str : PChar) :Str;
21527:  Func StrNewPchar( P : PChar) : PChar;
21528:  Proc StrDisposePchar( P : PChar);
21529: end;
21530:
21531: Proc SIRegister_crc_checks(CL: TPSPascalCompiler);
21532: begin
21533:  Func UpdateCrc32_2( InitCrc : longword; b : byte) : longword;
21534:  Func UpdateCrc16_2( InitCrc : word; b : byte) : word;
21535:  Func UpdateAdler32( InitAdler : longword; b : byte) : longword;
21536:  Func UpdateFletcher8( InitFletcher : word; b : byte) : word;
21537:  Func UpdateCRC( InitCrc : word; b : byte) : word;
21538: end;
21539:
21540: Proc SIRegister_extdos(CL: TPSPascalCompiler);
21541: begin
21542:  TypeS('utf8char', 'char;
21543:   TypeS('putf8char', 'pchar;
21544:   TypeS('utf16char', 'word;
21545:   TypeS('ucs4char', 'longword;
21546:   //TypeS('pucs4char', '^ucs4char // will not work;
21547:   TypeS('ucs2char', 'word;
21548:   TypeS('pucs2char', 'ucs2char; // will not work;
21549:   //TypeS('pucs2char', '^ucs2char // will not work;
21550:   TypeS('utf8string', 'string;
21551:  // TypeS('putf8shortstring', '^shortstring // will not work;
21552:   TypeS('_USER_INFO_2', record usri2_name : pucs2char; usri2_password : '
21553:    +pucs2char; usri2_password_age : DWORD; usri2_priv : DWORD; usri2_home_dir '
21554:    +: pucs2char; usri2_comment : pucs2char; usri2_flags : DWORD; usri2_script_'
21555:    +path : pucs2char; usri2_auth_flags : DWORD; usri2_full_name : pucs2char; u'
21556:    +sri2_usr_comment : pucs2char; usri2_parms : pucs2char; usri2_workstations '
21557:    +: pucs2char; usri2_last_logon : DWORD; usri2_last_logoff : DWORD; usri2_ac'
21558:    +ct_expires : DWORD; usri2_max_storage : DWORD; usri2_units_per_week : DWOR'
21559:    +D; usri2_logon_hours : pchar; usri2_bad_pw_count : DWORD; usri2_num_logons'
21560:    +:DWORD; usri2_logon_server:pucs2char;usri2_country_code:DWORD;usri2_code_page:DWORD; end;
21561:   //TypeS('P_USER_INFO_2', '^_USER_INFO_2 // will not work;
21562:   TypeS('SE_OBJECT_TYPE', '( SE_UNKNOWN_OBJECT_TYPE, SE_FILE_OBJECT, SE_S'
21563:    +'ERVICE, SE_PRINTER, SE_REGISTRY_KEY, SE_LMSHARE, SE_KERNEL_OBJECT, SE_WIND'
```

```
21564:    +'OW_OBJECT, SE_DS_OBJECT, SE_DS_OBJECT_ALL, SE_PROVIDER_DEFINED_OBJECT, SE_'
21565:    +'WMIGUID_OBJECT );
21566:   //TypeS('PPSID', '^PSID // will not work;
21567:   TypeS('ASSOCF', 'DWORD;
21568:   TypeS('ASSOCSTR', '( ASSOCSTR_NONE, ASSOCSTR_COMMAND, ASSOCSTR_EXECUTAB'
21569:    +'LE, ASSOCSTR_FRIENDLYDOCNAME, ASSOCSTR_FRIENDLYAPPNAME, ASSOCSTR_NOOPEN, A'
21570:    +'SSOCSTR_SHELLNEWVALUE,ASSOCSTR_DDECOMMAND,ASSOCSTR_DDEIFEXEC,ASSOCSTR_DDEAPPLICATION,
      ASSOCSTR_DDETOPIC);
21571:   TypeS('TCHAR', 'ucs2char;
21572:   {TypeS('TFileTime', 'FILETIME;
21573:   TypeS('TWin32FindDataW', '_WIN32_FIND_DATAW;
21574:   TypeS('TCHAR', 'ucs2char;
21575:   TypeS('TFileTime', 'FILETIME;
21576:   TypeS('TWin32FindDataW', '_WIN32_FIND_DATAW;   }
21577:   //dev: array[0..127] of char;
21578:   {** Unique file serial number, this may change from one boot to the next.}
21579:   //ino: array[0..127] of char;
21580:
21581: TypeS('tresourceattribute', '( attr_any, attr_readonly, attr_hidden, at'
21582:    +'tr_system, attr_archive, attr_link, attr_directory, attr_temporary, attr_e'
21583:    +'ncrypted,attr_no_indexing,attr_device,attr_extended,attr_compressed,attr_offline, attr_sparse);
21584:   TFileAssociation', 'record appname : utf8string; exename : utf8string; end;
21585:   tresourceattributes', 'set of tresourceattribute;
21586:   TypeS('TFileStats', record name : utf8string; size : big_Int_t; ow'
21587:    +ner : utf8string; ctime : TDateTime; mtime : TDateTime; atime : TDateTime;'
21588:    +nlink : Int; attributes : tresourceattributes; association : tfileass'
21589:    +ociation; streamcount:Int;accesses:Int; utc :Bool; dev: array[0..127] of char; '
21590:    +'ino: array[0..127] of char; comment : utf8string; dirstr : utf8string; end;
21591:  TSearchRecExt', 'record Stats : TFileStats; FindHandle : THandle'
21592:    +; W32FindData  TWin32FindData;IncludeAttr: longint; SearchAttr:TResourceAttributes; end;
21593:   Func GetFileOwner( fname : putf8char) : utf8string;
21594:   GetFileATime( fname : putf8char; var atime : TDateTime) : Int;
21595:   GetFileMTime( fname : putf8char; var mtime : TDateTime) : Int;
21596:   GetFileCTime( fname : putf8char; var ctime : TDateTime) : Int;
21597:   GetFilesizeExt( fname : putf8char) : big_Int_t;
21598:   GetFileAttributesExt( fname : putf8char) : tresourceattributes;
21599:   GetFilestats( fname : putf8char; var stats : TFileStats) : Int;
21600:   DirectoryExistsExt( DName : utf8string) :Bool;
21601:   FileExistsExt( const FName : utf8string) :Bool;
21602:   GetCurrentDirectoryExt( var DirStr : utf8string) :Bool;
21603:   SetCurrentDirectoryExt( const DirStr : utf8string) :Bool;
21604:   SetFileATime( fname : putf8char; newatime : tdatetime) : Int;
21605:   SetFileMTime( fname : putf8char; newmtime : tdatetime) : Int;
21606:   SetFileCTime( fname : putf8char; newctime : tdatetime) : Int;
21607:   FindFirstEx(path:putf8char; attr:tresourceattributes;var SearchRec:TSearchRecExt):int;
21608:   FindNextEx( var SearchRec : TSearchRecExt) : Int;
21609:   Proc FindCloseEx( var SearchRec : TSearchRecExt);
21610:   GetUserFullName( account : utf8string) : utf8string;
21611:   GetLoginConfigDirectory : utf8string;
21612:   GetGlobalConfigDirectory : utf8string;
21613:   GetLoginHomeDirectory : utf8string;
21614:  // ucs4strnewstr(str:Str; srctype:Str): pucs4char;;
21615:  // ucs4strnewucs4(src: pucs4char): pucs4char;;
21616: end;
21617:
21618: Proc SIRegister_xBase(CL: TPSPascalCompiler);
21619: begin
21620:  _INADDR_ANY','LongWord').SetUInt($00000000);
21621:  //CL.AddConstantN('DCX_EXCLUDERGN','LongWord').SetUInt( $40);
21622:  INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
21623:  CL.AddConstantN('INVALID_FILE_SIZE','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21624:  INVALID_FILE_ATTRIBUTES','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21625:  INVALID_FILE_TIME','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21626:  INVALID_REGISTRY_KEY','LongWord').SetInt( DWORD ( - 1 ));
21627:  INVALID_VALUE','LongWord').SetInt( DWORD ( - 1 ));
21628:  SleepQuant','LongInt').SetInt( 1 * 60 * 1000);
21629:  MMaxChars','LongInt').SetInt( 250);
21630:   CL.AddTypeS('TBase64Table', '( bsBase64, bsUUE, bsXXE );
21631:   CL.AddTypeS('TWOHandleArray', 'array[0..64 - 1] of THandle;;
21632:  CL.AddTypeS('TWOHandleArray2', 'array[0..MAXIMUM_WAIT_OBJECTS - 1] of THandle;;
21633:  //type TWOHandleArray = array[0..MAXIMUM_WAIT_OBJECTS - 1] of THandle;
21634:   // MAXIMUM_WAIT_OBJECTS = 64;
21635:  SIRegister_TMimeCoder(CL);
21636:   CL.AddTypeS('TSocketOption', '(soBroadcast, soDebug, soDontLinger, soDontRou'
21637:    +'te, soKeepAlive, soOOBInLine, soReuseAddr, soNoDelay, soBlocking,soAcceptConn );
21638:   TSocketOptions', 'set of TSocketOption;
21639:   //TSocketClass', 'class of TSocket;
21640:   SIRegister_TSocketX(CL);
21641:   TObjProcX', 'Procedure;
21642:   TForEachProc', 'Proc ( P : _Pointer);
21643:   //PFileInfo', '^TFileInfo // will not work;
21644:   TFileInfo', 'record Attr : DWORD; Size : DWORD; Time : DWORD; end;
21645:   TuFindData', 'record Info : TFileInfo; FName :Str; end;
21646:   TCreateFileMode', '( cRead, cWrite, cFlag, cEnsureNew, cTruncate'
21647:    +, cExisting, cShareAllowWrite, cShareDenyRead,cOverlapped,cRandomAccess,cSequentialScan,cDeleteOnClose
      );
21648:   TCreateFileModeSet', 'set of TCreateFileMode;
21649:   //PCharSet', '^TCharSet // will not work;
21650:   TCharSetX', 'set of Char;
```

```
21651:   //PCharArray', '^TCharArray // will not work;
21652:   //PByteArray', '^TByteArray // will not work;
21653:   //PIntArray', '^TIntArray // will not work;
21654:   //PDwordArray', '^TDwordArray // will not work;
21655:   //PvIntArr', '^TvIntArr // will not work;
21656:   //TvIntArr', 'record Arr : PIntArray; Cnt : Integer; end;
21657:   //PBoolean', '^Boolean // will not work;
21658:   //PItemList', '^TItemList // will not work;
21659:   TThreadMethodX', 'Procedure;
21660:   TThreadPriorityXX,(tpIdle tpLowest,tpLower,tpNormal,tpHigher,tpHighest,tpTimeCritical);
21661:   SIRegister_TThreadX(CL);
21662:   CL.AddClassN(CL.FindClass('TOBJECT'),'TAdvObject;
21663:   SIRegister_TAdvObject(CL);
21664:   SIRegister_TAdvCpObject(CL);
21665:   //TAdvClass', 'class of TAdvObject;
21666:   //TCollClass', 'class of TColl;
21667:   SIRegister_TColl(CL);
21668:   SIRegister_TSortedColl(CL);
21669:   SIRegister_TStringColl(CL);
21670:  Func AddRightSpaces( const S :Str; NumSpaces : Integer) :Str;
21671:  Proc AddStr( var S :Str; C : char);
21672:  Proc Add_Str( var S : ShortString; C : char);
21673:  Func CompareStrX( const S1, S2 :Str) : Integer;
21674:  Func CopyLeft( const S :Str; I : Integer) :Str;
21675:  Proc DelDoubles( const St :Str; var Source :Str);
21676:  Proc DelFC( var s :Str);
21677:  Proc DelLC( var s :Str);
21678:  Func DelLeft( const S :Str) :Str;
21679:  Func DelRight( const S :Str) :Str;
21680:  Func DelSpaces( const s :Str) :Str;
21681:  Proc DeleteLeft( var S :Str; I : Integer);
21682:  Func DigitsOnly( const AStr :Str) :Bool;
21683:  Proc DisposeStr( P :Str);
21684:  Func ExpandFileNameX( const FileName :Str) :Str;
21685:  Func ExtractFilePathX( const FileName :Str) :Str;
21686:  Func ExtractDir( const S :Str) :Str;
21687:  Func ExtractFileRootX( const FileName :Str) :Str;
21688:  Func ExtractFileExtX( const FileName :Str) :Str;
21689:  Func ExtractFileNameX( const FileName :Str) :Str;
21690:  Func ExtractFileDriveX( const FileName :Str) :Str;
21691:  Func ExtractFileDirX( const FileName :Str) :Str;
21692:  Proc FSplit( const FName :Str; var Path, Name, Ext :Str);
21693:  Proc FillCharSetX( const AStr :Str; var CharSet : TCharSet);
21694:  Func GetWrdStrictUC( var s, w :Str) :Bool;
21695:  Func GetWrdStrict( var s, w :Str) :Bool;
21696:  Func GetWrdD( var s, w :Str) :Bool;
21697:  Func GetWrdA( var s, w :Str) :Bool;
21698:  Func GetWrd( var s, w :Str; c : char) :Bool;
21699:  Func Hex2X( a : Byte) :Str;
21700:  Func Hex4X( a : Word) :Str;
21701:  Func Hex8X( a : DWORD) :Str;
21702:  Func Int2Hex( a : Integer) :Str;
21703:  Func Int2StrX( L : Integer) :Str;
21704:  Func ItoS( I : Integer) :Str;
21705:  Func ItoSz( I, Width : Integer) :Str;
21706:  Func LastDelimiterX( const Delimiters, S :Str) : Integer;
21707:  Func LowerCaseX( const S :Str) :Str;
21708:  Func MakeFullDir( const D, S :Str) :Str;
21709:  Func MakeNormName( const Path, Name :Str) :Str;
21710:  Func MonthE( m : Integer) :Str;
21711:  Func NewStr( const S :Str) :Str;
21712:  Func ReplaceX( const Pattern, ReplaceString:Str; var S:Str):Bool;
21713:  Func StoI( const S :Str) : Integer;
21714:  Func StrEnds( const S1, S2 :Str) :Bool;
21715:  Func StrRightX( const S :Str; Num : Integer) :Str;
21716:  Func UpperCaseX( const S :Str) :Str;
21717:  Func WipeChars( const AStr, AWipeChars :Str) :Str;
21718:  Func _Val( const S :Str; var V : Integer) :Bool;
21719:  Func ProcessQuotes( var s :Str) :Bool;
21720:  Func UnpackPchars( var s :Str) :Bool;
21721:  Func UnpackUchars( var s :Str) :Bool;
21722:  Func __alpha( c : char) :Bool;
21723:  Func __ctl( c : char) :Bool;
21724:  Func __digit( c : char) :Bool;
21725:  Func __extra( c : char) :Bool;
21726:  Func __national( c : char) :Bool;
21727:  Func __pchar( c : char) :Bool;
21728:  Func __reserved( c : char) :Bool;
21729:  Func __safe( c : char) :Bool;
21730:  Func __uchar( c : char) :Bool;
21731:  Func __unsafe( c : char) :Bool;
21732:  Func Buf2Str( const Buffer:Str) :Str;
21733:  //Proc Clear( var Buf, Count : Integer);
21734:  Func CompareMem( P1, P2 : _Pointer; Length : Integer) :Bool;
21735:  Proc FreeObjectX( var O: TObject);
21736:  Proc LowerPrec( var A, B : Integer; Bits : Byte);
21737:  Func MemEqu( const A, B, Sz : Integer) :Bool;
21738:  Func MaxIX( A, B : Integer) : Integer;
21739:  Func MinIX( A, B : Integer) : Integer;
```

```
21740:  Func MaxDX( A, B : DWORD) : DWORD;
21741:  Func MinDX( A, B : DWORD) : DWORD;
21742:  Func NulSearch( const Buffer:Str) : Integer;
21743:  Func NumBits( I : Integer) : Integer;
21744:  //Proc XAdd( var Critical, Normal);
21745:  //Proc XChg( var Critical, Normal);
21746:  Func CreateEvtA : DWORD;
21747:  Func CreateEvt( Initial :Bool) : DWORD;
21748:  Func SignaledEvt( id : DWORD) :Bool;
21749:  Func WaitEvt( const id : TWOHandleArray; Timeout : DWORD) : DWORD;
21750:  //Func WaitEvtA(nCount: Integer;lpHandles: PWOHandleArray; Timeout:DWORD): DWORD;
21751:  Func ClearHandle( var Handle : THandle) :Bool;
21752:  Proc CloseHandles( const Handles : array of DWORD);
21753:  Func FileExistsX( const FName :Str) :Bool;
21754:  Func FindExecutableX( FileName, Directory : PChar; Result : PChar) : HINST;
21755:  Func GetEnvVariable( const Name :Str) :Str;
21756:  Func GetFileNfo(const FName:Str;var Info: TFileInfo; NeedAttr:Boolean):Bool;
21757:  Func GetFileNfoByHandle( Handle : DWORD; var Info : TFileInfo) :Bool;
21758:  Func ZeroHandle( var Handle : THandle) :Bool;
21759:  Func _CreateFile( const FName :Str; Mode : TCreateFileModeSet) : DWORD;
21760:  Func _CreateFileSecurity(const
        FName:str;Mode:TCreateFileModeSet;lpSecurityAttributes:PSecurityAttributes):DWORD;
21761:  Func _GetFileSize( const FName :Str) : DWORD;
21762:  Func _MatchMaskBody( AName, AMask :Str; SupportPercent :Bool) :Bool;
21763:  Func _MatchMask(const AName:str; AMask :Str;SupportPercent:Boolean):Boolean;
21764:  Func MatchMask( const AName, AMask :Str) :Bool;
21765:  Func SysErrorMsg( ErrorCode : DWORD) :Str;
21766:  Func CreateRegKeyX( const AFName :Str) : HKey;
21767:  Func OpenRegKeyEx( const AName :Str; AMode : DWORD) : HKey;
21768:  Func OpenRegKey( const AName :Str) : DWORD;
21769:  Func ReadRegBin(Key: DWORD; const rvn :Str; Bin: _Pointer; Sz : DWORD) :Bool;
21770:  Func ReadRegInt( Key : DWORD; const AStrName :Str) : DWORD;
21771:  Func ReadRegString( Key : DWORD; const AStrName :Str) :Str;
21772:  Func WriteRegBin(Key:DWORD;const rvn :Str; Bin: _Pointer; Sz : DWORD):Bool;
21773:  Func WriteRegInt( Key : DWORD; const AStrName :Str; AValue : DWORD) :Bool;
21774:  Func WriteRegString( Key : DWORD; const AStrName, AStr :Str) :Bool;
21775:  Func AddrInet( i : DWORD) :Str;
21776:  Func GetHostNameByAddr( Addr : DWORD) :Str;
21777:  Func Inet2addr( const s :Str) : DWORD;
21778:  Func InetAddr( const s :Str) : DWORD;
21779:  Proc GlobalFail;
21780:  Func _LogOK( const Name :Str; var Handle : DWORD) :Bool;
21781:  Proc xBaseDone;
21782:  Proc xBaseInit;
21783:  Proc uCvtSetFileTime( T : DWORD; var L, H : DWORD);
21784:  Func uCvtGetFileTime( L, H : DWORD) : DWORD;
21785:  Func uGetSystemTime : DWORD;
21786:  Func VlX( const s :Str) : DWORD;
21787:  Func VlHX( const s :Str) : DWORD;
21788:  Func StrAsg( const Src :Str) :Str;
21789:  SIRegister_TResetterThread(CL);
21790:  CServerVersion','String').SetString( '1.93;
21791:  CServerProductName','String').SetString( 'TinyWeb;
21792: end;
21793:
21794: Proc SIRegister_ImageHistogram(CL: TPSPascalCompiler);
21795: begin
21796:   THistogramChannel', '( hclGray, hclRed, hclGreen, hclBlue );
21797:   THistogramArray', 'array[0..255] of Cardinal;;
21798:   //THistogramArray = array[0..255] of Cardinal;
21799:   SIRegister_THistogram(CL);
21800:  Proc AdjustBitmap(Bmp: TBitmap;Channel: THistogramChannel; Tolerance :Card);
21801:  Proc AdjustBitmap1(Bmp:TBitmap;Hist:THistogram;Channel:THistogramChannel;Tolerance:Card);
21802:  Proc AdjustBitmap2(Bmp: TBitmap; Low, High : Byte; Channel : THistogramChannel);
21803: end;
21804:
21805: Proc SIRegister_cCipherRSA(CL: TPSPascalCompiler);
21806: begin
21807:  CL.AddClassN(CL.FindClass('TOBJECT'),'ERSA;
21808:   TRSAPublicKey', 'record KeySize : Integer; Modulus : HugeWord; Exponent : HugeWord; end;
21809:   TRSAPrivateKey', 'record KeySize : Integer; Modulus : HugeWord; '
21810:   Exponent : HugeWord; PublicExponent : HugeWord; Prime1 : HugeWord; Prime2 '
21811:   : HugeWord;Phi:HugeWord; Exponent1:HugeWord; Exponent2:HugeWord; Coefficient:HugeWord; end;
21812:   CL.AddTypeS('TRSAMessage', 'HugeWord;
21813:  TRSAEncryptionType', '( rsaetPKCS1, rsaetOAEP );
21814:  Proc RSAPublicKeyInit( var Key : TRSAPublicKey);
21815:  Proc RSAPublicKeyFinalise( var Key : TRSAPublicKey);
21816:  Proc RSAPublicKeyAssign( var KeyD : TRSAPublicKey; const KeyS : TRSAPublicKey);
21817:  Proc RSAPublicKeyAssignHex(var Key:TRSAPublicKey;const KeySize:Int;const HexMod,HexExp:Str
21818: Proc RSAPublicKeyAssignBuf(var Key:TRSAPublicKey;const KeySize:Int;const ModBuf:str;const
        ModBufSize:Integer;const ExpBuf:str;const ExpBufSize:Integer;const ReverseByteOrder:Bool);
21819: Proc RSAPublicKeyAssignBufStr(var Key:TRSAPublicKey;const KeySize:Integer;const ModBuf,ExpBuf:Ansistr);
21820:  Proc RSAPrivateKeyInit( var Key : TRSAPrivateKey);
21821:  Proc RSAPrivateKeyFinalise( var Key : TRSAPrivateKey);
21822:  Proc RSAPrivateKeyAssign( var KeyD : TRSAPrivateKey; const KeyS : TRSAPrivateKey);
21823:  Proc RSAPrivateKeyAssignHex(var Key:TRSAPrivateKey;const KeySize:Integer;const HexMod, HexExp:Ansistr);
21824:  Proc RSAPrivateKeyAssignBuf(var Key:TRSAPrivateKey;const KeySize:Integer;const ModBuf:Str; const
        ModBufSize:Integer;const ExpBuf:Str; const ExpBufSize:Integer; const ReverseByteOrder:Boolean);
21825:  Proc RSAPrivateKeyAssignBufStr(var Key:TRSAPrivateKey;const KeySize:Integer;const ModBuf,ExpBuf:Ansistr);
```

```
21826:  Proc RSAGenerateKeys(const KeySize:Int;var PrivateKey:TRSAPrivateKey;var PublicKey:TRSAPublicKey);
21827:  Func RSACipherMessageBufSize(const KeySize: Integer) : Integer;
21828:  Proc RSAEncodeMessagePKCS1(const KeySize:Integer;const Buf:str;const BufSize:Integer;var EncodedMessage:
        TRSAMessage);
21829:  Proc RSAEncodeMessageOAEP(const KeySize:Integer;const Buf:str;const BufSize:Integer;var EncodedMessage:
        TRSAMessage);
21830:  Proc RSAEncryptMessage(const PublicKey:TRSAPublicKey;const PlainMessage:TRSAMessage;var
        CipherMessage:TRSAMessage);
21831:  Func RSACipherMessageToBuf(const KeySize:Integer;const CipherMessage:TRSAMessage;var CipherBuf:str;const
        CipherBufSize:Int):Int;
21832:  Func RSAEncryptC(const EncryptionType:TRSAEncryptionType; const PublicKey:TRSAPublicKey; const
        PlainBuf:str;const PlainBufSize:Int;var CipherBuf:str;const CipherBufSize:Int):Int;
21833:  Func RSAEncryptStr(const EncryptionType:TRSAEncryptionType;const PublicKey:TRSAPublicKey;const
        Plain:Ansistr): Ansistr;
21834:  Proc RSACipherBufToMessage(const KeySize:Integer;const CipherBuf:str;const CipherBufSize:Int;var
        CipherMessage:TRSAMessage);
21835:  Proc RSADecryptMessage(const PrivateKey:TRSAPrivateKey;const CipherMessage:TRSAMessage;var
        EncodedMessag:TRSAMessage);
21836:  Func RSADecodeMessagePKCS1(const KeySize:Integer;const EncodedMessage:HugeWord;var Buf:str;const BufSize:
        Integer):Integer;
21837:  Func RSADecodeMessageOAEP(const KeySize:Integer;const EncodedMessage:HugeWord;var Buf:str;const BufSize:
        Integer):Integer;
21838:  Func RSADecryptC( const EncryptionType : TRSAEncryptionType; const PrivateKey : TRSAPrivateKey; const
        CipherBuf :Str; const CipherBufSize : Integer; var PlainBuf :Str; const PlainBufSize : Integer) : Integer;
21839:  Func RSADecryptStr(const EncryptionType:TRSAEncryptionType;const PrivateKey:TRSAPrivateKey;const
        Cipher:Ansistr):Ansistr;
21840:  Proc SelfTestRSA;
21841:  //CL.AddDelphiFunction('Proc Profile;
21842: end;
21843:
21844: // add to V 4.2.6.10
21845: Proc SIRegister_Streams(CL: TPSPascalCompiler);
21846: begin
21847:  SIRegister_TBaseStream(CL);
21848:   SIRegister_TFilterStream(CL);
21849:   SIRegister_TSeekableStream(CL);
21850:   //CL.AddTypeS('TFileAttributes2', 'set of ( sfaArchive, sfaHidden, sfaSystem, sf'
21851:    //+'aReadOnly, sfaTemporary, sfaCompressed, sfaDirectory );
21852: TFileAttribute2,(sfaArchive,sfaHidden,sfaSystem,sfaReadOnly,sfaTemporary,sfaCompressed, sfaDirectory);
21853:  TFileAttributes2, set of TFileAttribute2');
21854:   TfsMode','(sfsRead, sfsWrite, sfsCreate, sfsShareRead, sfsShareWrite);
21855:   TfsModes', 'set of TfsMode');
21856: // CL.AddTypeS('TfsModes','set of( fsRead, fsWrite,fsCreate,fsShareRead,fsShareWrite );
21857:  fsReset','LongInt).Value.ts32 := ord(fsRead) or ord(fsWrite);
21858:  fsShared','LongInt).Value.ts32:= ord(fsShareRead) or ord(fsShareWrite);
21859:   SIRegister_TFileStream2(CL);
21860:   SIRegister_TDummyStream(CL);
21861:  Proc CopyStream( Source, Dest: TBaseStream; const Size : Integer);
21862:  Func GetLastErrorText :Str');
21863: end;
21864:
21865: Proc SIRegister_CromisStreams(CL: TPSPascalCompiler);
21866: begin
21867:  Proc WriteToStreamAsString( const Stream : TStream; const Content :Str);
21868:  Proc WriteToStreamAsUnicode( const Stream : TStream; const Content :Str);
21869:  Proc WriteToStreamAsUTF80( const Stream : TStream; const Content :Str);
21870:  Proc WriteToStreamAsUTF81( const Stream : TStream; const Content: Ansistr);
21871:  Func ReadFromStreamAsUnicode( const Stream : TStream) :Str');
21872:  Func ReadFromStreamAsString( const Stream : TStream) :Str');
21873:  Func ReadFromStreamAsUTF8( const Stream : TStream) : Ansistr');
21874:  CL.AddConstantN('cDefaultSearchSize','LongInt').SetInt( 50);
21875:   SIRegister_TNamesEnumerator(CL);
21876:   SIRegister_IStorageStream(CL);
21877:   SIRegister_TStreamStorage(CL);
21878: end;
21879:
21880: // new functions on 4.2.6.10
21881: Func ArcTan3(const X : Extended) : Extended');  //bugfix of ArcTan()
21882: Func WMIConnect2(WBemLocator:ISWBemLocator;namespace,Server,account,password:str): ISWBemServices;
21883: Func WMIConnect3(WBemLocator:ISWBemLocator;Server,namespace,account,password:str): ISWBemServices;
21884: Func CLSIDFromProgID(pszProgID: widestring; out clsid: TGUID): Longint'); //stdcall;
21885: Func ProgIDFromCLSID(const clsid:TGUID; out pszProgID: widestring): Longint'); //stdcall;
21886: Func StringFromCLSID(const clsid:TGUID; out psz: widestring): longint'); //stdcall;
21887: //{$EXTERNALSYM CLSIDFromString}
21888:  Func CLSIDFromString(psz: widestring; out clsid: TGUID): longint'); //stdcall;
21889: Func StringFromGUID2(const guid: TGUID;psz:widestring;cbMax:Integer):Integer'); //stdcall;
21890: Func ProgIDExists(const ProgID:WideString):Boolean');
21891: Func ByteArraytostring(const bin: array of byte):Str;
21892: Func ByteArraytoHex(const bin: array of byte):Str');
21893: Proc SaveFile(Contents : Ansistr; const FileName:Ansistr; Append :Bool);
21894: Proc SaveToFile(Contents: Ansistr;const FileName:Ansistr; Append :Bool);
21895: Func LoadFromFile( const FileName : Ansistr) : Ansistr');
21896: Func VariantIsObject(const V: Variant):Bool;
21897: Func DiskSpaceInfo(const Drive:str; out AvailBytes,TotalBytes,FreeBytes:Int64):Boolean;
21898: Func DriveDisplayName(const Drive:Str):Str;
21899: Proc DriveDisplayNames(const List: TStrings);
21900: Proc Log( const Line:Str; amemo: Tmemo);
21901: begin   amemo.Lines.Add( Line)  end;
21902: Proc LogFmtX( const Line:Str; amemo: TMemo; const Args: array of const);
```

```
21903: begin
21904:   Log( Format( Line, Args), amemo)
21905: end;
21906:
21907: Proc SIRegister_USha256(CL: TPSPascalCompiler);
21908: begin
21909:  type TSHA256HASH', 'array[0..7] of Cardinal');
21910:  type TSHAChunk', 'array[0..7] of Cardinal');
21911:  //TSHA256HASH = array[0..7] of Cardinal;
21912:   //TChunk = array[0..15] of Cardinal;
21913:  Func CalcDoubleSHA256( Msg : Ansistr) : TSHA256HASH;
21914:  Func CalcSHA256( Msg : Ansistr) : TSHA256HASH;
21915:  Func CalcSHA2561( Stream : TStream) : TSHA256HASH;
21916:  Func SHA256ToStr( Hash : TSHA256HASH) :Str;
21917:  Func CanBeModifiedOnLastChunk(MessageTotalLength:Int64;var startBytePos:int):Bool;
21918:  Proc PascalCoinPrepareLastChunk(const messageToHash:Ansistr;var stateForLastChunk:TSHA256HASH;var
        bufferForLastChunk : TSHAChunk);
21919:  Func ExecuteLastChunk(const stateForLastChunk:TSHA256HASH;const bufferForLastChunk:
        TSHAChunk;nPos:Integer;nOnce,Timestamp:Card):TSHA256HASH');
21920:  Func ExecuteLastChunkAndDoSha256(const stateForLastChunk:TSHA256HASH;const
        bufferForLastChunk:TSHAChunk;nPos:Integer;nOnce,Timestamp:Card):TSHA256HASH');
21921:  Proc PascalCoinExecuteLastChunkAndDoSha256(const stateForLastChunk:TSHA256HASH;const
        bufferForLastChunk:TSHAChunk;nPos:Integer;nOnce,Timestamp:Card;var ResultSha256:AnsiStr);
21922:  Func Sha256HashToRaw(const hash : TSHA256HASH): Ansistr');
21923: Func GetSHA256( Msg : Ansistr) :Str;
21924: Func GetDriveNumber(const Drive:Str): Integer;
21925: Func HardDiskSerial(const Drive:Str): DWORD;
21926: Func IsDriveReady2(const Drive:Str):Bool;
21927: Func Touchfile(const FileName:Str):Bool;
21928: Func URLFromShortcut(const Shortcut:Str):Str;
21929:
21930: Func GetSHA256(Msg: Ansistr):Str; //overload;
21931: var Stream: TMemoryStream;
21932: begin
21933:   Stream:= TMemoryStream.Create;
21934:   try
21935:     Stream.WriteBuffer(PAnsiChar(Msg)^,Length(Msg));
21936:     Stream.Position:= 0;
21937:     Result:= SHA256ToStr(CalcSHA256(Stream));
21938:   finally
21939:     Stream.Free;
21940:   end;
21941: end;
21942:
21943: Code Example Binary file:
21944:    sr:= filetostring(Exepath+'maXbox4.exe')
21945:    writeln(SHA256ToStr(CalcSHA256(sr)))
21946:    stringtofile(Exepath+'maXbox4.exe',sr,false); //append = false
21947:
21948: Proc SIRegister_uTPLb_BinaryUtils(CL: TPSPascalCompiler);
21949: begin
21950:  Func SwapEndien_u32(Value: uint32) : uint32');
21951:  Func SwapEndien_s64(Value: int64) : int64');
21952:  Func SwapEndien_u64(Value: uint64) : uint64');
21953:  Func RotateLeft1Bit_u32( Value : uint32) : uint32');
21954:  Proc Read_BigEndien_u32_Hex( const Value :Str; BinaryOut: TStream);
21955:  Func Get_TP_LockBox3_HINSTANCE : HMODULE');
21956: end;
21957:
21958: Proc SIRegister_Series(CL: TPSPascalCompiler);
21959: begin
21960:  CL.AddConstantN('PiDegree','Double').setExtended( Pi / 180.0);
21961:  Tee_CircledShadowColor','LongWord).SetUInt( TColor ( $A0A0A0 ));
21962:  CL.AddTypeS('TTreatNullsStyle', '( teetnDontPaint, teetnSkip, teetnIgnore );
21963:  SIRegister_TCustomLineSeries(CL);
21964:  CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomSeries');
21965:  CL.AddTypeS(TSeriesClickPointerEvent','Procedure(Sender:TCustomSeries;ValueIndex:Int;X,Y:Int;
21966:  TCustomSeriesStack','( cssNone, cssOverlap, cssStack, cssStack100 );
21967:  TOnGetPointerStyle','Func ( Sender : TChartSeries; ValueIndex: Integer) : TSeriesPointerStyle');
21968:  SIRegister_TCustomSeries(CL);
21969:   SIRegister_TLineSeries(CL);
21970:   SIRegister_THorizLineSeries(CL);
21971:   SIRegister_TPointSeries(CL);
21972:   CL.AddTypeS(TMultiArea','( mamaNone,maStacked, maStacked100 );
21973:   SIRegister_TAreaSeries(CL);
21974:   SIRegister_THorizAreaSeries(CL);
21975:  TMultiBar,(teembNone,teembSide,teembStacked,teembStacked100,teembSideAll,teembSelfStack);
21976:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomBarSeries');
21977:   TBarStyle,(teebsRectangle, teebsPyramid, teebsInvPyramid, teebsCilinder,teebsEllipse, teebsArrow,
        teebsRectGradient,teebsCone,teebsBevel,teebsSlantCube,teebsDiamond,teebsInvArrow, teebsInvCone);
21978:   TGetBarStyleEvent,Procedure(Sender:TCustomBarSeries;ValueIndex:Integer;var TheBarStyle: TBarStyle);
21979:   SIRegister_TBarSeriesGradient(CL);
21980:   SIRegister_TCustomBarSeries(CL);
21981:   SIRegister_TBarSeries(CL);
21982:   SIRegister_THorizBarSeries(CL);
21983:   SIRegister_TCircledShadow(CL);
21984:   SIRegister_TCircledSeries(CL);
21985:   TPieAngle', 'record StartAngle : Double; MidAngle : Double; EndAngle : Double; end');
21986:   TPieAngles', 'array of TPieAngle');
```

```
21987:   SIRegister_TSliceValueList(CL);
21988:   TPieOtherStyle', '( piepoNone, piepoBelowPercent, piepoBelowValue );
21989:   SIRegister_TPieOtherSlice(CL);
21990:   SIRegister_TPieMarks(CL);
21991:   TMultiPie', '( teempAutomatic, teempDisabled );
21992:   SIRegister_TPieSeries(CL);
21993:   SIRegister_TFastLineSeries(CL);
21994:  //CL.AddConstantN('bsCylinder','').SetString( bsCilinder);
21995:  Proc RegisterTeeStandardSeries');
21996:  Proc TePointerDrawLegend(Pointer:TSeriesPointer;AColor:TColor;const Rect:TRect;DrawPen:Bool;
       AStyle:TSeriesPointerStyle);
21997: end;
21998:
21999: Proc SIRegister_UJSONFunctions(CL: TPSPascalCompiler);
22000: begin
22001:   CL.AddTypeS('TJSONValue', 'TJSONData');
22002:   SIRegister_TPCJSONData(CL);
22003:   //CL.AddTypeS('TPCJSONDataClass', 'class of TPCJSONData');
22004:   SIRegister_TPCJSONVariantValue(CL);
22005:   SIRegister_TPCJSONNameValue(CL);
22006:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPCJSONArray');
22007:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPCJSONObject');
22008:   SIRegister_TPCJSONList(CL);
22009:   SIRegister_TPCJSONArray(CL);
22010:   SIRegister_TPCJSONObject(CL);
22011:   CL.AddClassN(CL.FindClass(TOBJECT),EPCParametresError');
22012: end;
22013:
22014: Proc SIRegister_uTPLb_Hash(CL: TPSPascalCompiler);
22015: begin
22016:   CL.AddTypeS(TOnHashProgress,Function(sender:TObject;CountBytesProcessed:int64):bool;
22017:   SIRegister_IHash(CL);
22018:   SIRegister_IHash_TestAccess(CL);
22019:   SIRegister_TSimpleHash(CL);
22020:   SIRegister_THash(CL);
22021: end;
22022:
22023: Proc SIRegister_UTime(CL: TPSPascalCompiler);
22024: begin
22025:  //Func TzSpecificLocalTimeToSystemTime( lpTimeZoneInformation:PTimeZoneInformation; var lpLocalTime,
       lpUniversalTime : TSystemTime) : BOOL');
22026:  //Func SystemTimeToTzSpecificLocalTime( lpTimeZoneInformation:PTimeZoneInformation;var lpUniversalTime,
       lpLocalTime : TSystemTime) : BOOL');
22027:  Func DateTime2UnivDateTime(d : TDateTime) : TDateTime');
22028:  Func UnivDateTime2LocalDateTime(d: TDateTime): TDateTime');
22029:  Func UnivDateTimeToUnix(dtDate: TDateTime : Longint');
22030:  Func UnixToUnivDateTime(USec: Longint): TDateTime');
22031:  Func UnixTimeToLocalElapsedTime( USec : Longint): Ansistr');
22032:  Func DateTimeElapsedTime(dtDate: TDateTime): Ansistr');
22033: end;
22034:
22035: Proc SIRegister_uTPLb_BlockCipher(CL: TPSPascalCompiler);
22036: begin
22037:   SIRegister_IBlockCodec(CL);
22038:   SIRegister_IBlockCipher(CL);
22039:   SIRegister_TBlockChainLink(CL);
22040:   CL.AddTypeS(TChainingFeature', '( cfNoNounce, cfKeyStream, cfAutoXOR, cf8bit);
22041:   CL.AddTypeS(TChainingFeatureSet', 'set of TChainingFeature');
22042:   SIRegister_IBlockChainingModel(CL);
22043:   SIRegister_IBlockCipherSelector(CL);
22044: end;
22045:
22046: Proc SIRegister_XMLDoc(CL: TPSPascalCompiler);
22047: begin
22048:   FindInterface('IUNKNOWN'),IXMLNode, 'IXMLNode');
22049:   TOBJECT'),'TXMLNode');
22050:   TOBJECT'),'TXMLNodeList');
22051:   TOBJECT'),'TXMLNodeCollection');
22052:   TOBJECT'),'TXMLDocument');
22053:   CL.AddTypeS(TNodeListOperation','( nlInsert, nlRemove, nlCreateNode );
22054:   CL.AddTypeS(TNodeListNotification', 'Proc ( Operation : TNodeListOperat'
22055:    +ion; var Node :IXMLNode; const IndexOrName: OleVariant; BeforeOperation :Bool);
22056:   SIRegister_TXMLNodeList(CL);
22057:    CL.AddTypeS(TAsyncEventHandler,procedure(Sender:TObject;AsyncLoadState:Integer) of Object);
22058:   //CL.AddTypeS('TXMLNodeClass', 'class of TXMLNode');
22059:   TXMLNodeArray', 'array of TXMLNode');
22060:   TNodeClassInfo','record NodeName:DOMString;NamespaceURI:DOMString;NodeClass:TXMLNode; end');
22061:   CL.AddTypeS(TNodeClassArray', 'array of TNodeClassInfo');
22062:   //CL.AddTypeS('TXMLNodeCollectionClass', 'class of TXMLNodeCollection');
22063:   CL.AddTypeS(TNodeChange', '( ncUpdateValue, ncInsertChild, ncRemoveChild, nc'
22064:    +AddAttribute, ncRemoveAttribute );
22065:   SIRegister_IXMLNodeAccess(CL);
22066:   SIRegister_TXMLNode(CL);
22067:   SIRegister_TXMLNodeCollection(CL);
22068:   TNodeChangeEvent','Proc ( const Node : IXMLNode; ChangeType : TNodeChange);
22069:   TXMLPrologItem','( xmlpVersion, xmlpEncoding, xmlpStandalone );
22070:   TXMLDocumentSource','( xdsNone, xdsXMLProperty, xdsXMLData, xdsFile, xdsStream);
22071:   SIRegister_IXMLDocumentAccess(CL);
22072:   SIRegister_TXMLDocument(CL);
```

```
22073: Func CreateDOMNode(Doc:IDOMDocument;const NameOrData:DOMString;NodeType:TNodeType;const
       AddlData:DOMString):IDOMNode;
22074: Func DetectCharEncoding( S : TStream) : TXmlEncodingType');
22075: Proc CheckEncoding(var XMLData: DOMString; const ValidEncodings: array of string);
22076: Func XMLStringToWideString( const XMLString :Str) : WideString;
22077: Func FormatXMLData( const XMLData : DOMString) : DOMString;
22078: Func LoadXMLDocument3( const FileName : DOMString) : IXMLDocument');
22079: Func LoadXMLData( const XMLData : DOMString) : IXMLDocument;
22080: Func LoadXMLData2( const XMLData :Str) : IXMLDocument;
22081: Func NewXMLDocument( Version : DOMString) : IXMLDocument');
22082: Proc XMLDocError( const Msg :Str);
22083: Proc XMLDocError2( const Msg :Str; const Args : array of const);
22084: Func CreateXMLDocument()
22085: end;
22086:
22087: Proc SIRegister_uTPLb_Codec(CL: TPSPascalCompiler);
22088: begin
22089:   SIRegister_TSimpleCodec(CL);
22090:   SIRegister_ICodec_TestAccess(CL);
22091:   SIRegister_TCodec(CL);
22092: end;
22093:
22094: Proc SIRegister_uTPLb_StreamCipher(CL: TPSPascalCompiler);
22095: begin
22096:   CL.AddTypeS(TAlgorithmicFeature', '( afStar, afCryptographicallyWeak, afNotI'
22097:    +mplementedYet, afForTestOnly, afForRunTimeOnly, afEncumberedByPatent, afEn'
22098:    +cumberedByCopyRight, afOpenSourceSoftware, afCommercial, afCompressor, afC'
22099:    +onverter, afBlockAdapter, afDisplayNameOnKeySize, afDoesNotNeedSalt, afAsymetric );
22100:   CL.AddTypeS(TAlgorithmicFeatureSet', 'set of TAlgorithmicFeature');
22101:   SIRegister_ICryptoGraphicAlgorithm(CL);
22102:   SIRegister_TSymetricKey(CL);
22103:   SIRegister_IStreamEncryptor(CL);
22104:   SIRegister_IStreamDecryptor(CL);
22105:   SIRegister_IStreamCipher(CL);
22106:   SIRegister_IisBase64Converter(CL);
22107: end;
22108:
22109: Proc SIRegister_MidasCon(CL: TPSPascalCompiler);
22110: begin
22111:   SIRegister_TRemoteServer(CL);
22112:   CL.AddTypeS(TConnectType', '( ctDCOM, ctSockets, ctOLEnterprise );
22113:   SIRegister_TMidasConnection(CL);
22114: end;
22115:
22116: Proc SIRegister_IXmlDocument2(CL: TPSPascalCompiler);
22117: begin
22118:   //with RegInterfaceS(CL,'IXmlNode', 'IXmlDocument') do
22119:   with CL.AddInterface(CL.FindInterface('IXmlBase'),IXmlNode2,'IXmlDocument2') do begin
22120:   Func Get_DocumentElement : IXmlElement', cdRegister);
22121:   Func Get_BinaryXML : RawByteString', r);
22122:   Func Get_PreserveWhiteSpace :Bool', );
22123:   Proc Set_PreserveWhiteSpace( aValue :Bool)', );
22124:   Func Get_OnTagEnd : THookTag', cdRegister);
22125:   Proc Set_OnTagEnd( aValue : THookTag)', cdRegister);
22126:   Func Get_OnTagBegin : THookTag', cdRegister);
22127:   Proc Set_OnTagBegin( aValue : THookTag)', cdRegister);
22128:   Func NewDocument(const aVersion,anEncoding:TXmlString;aRootElementNameID:NativeInt): IXmlElement;
22129:   Func NewDocument1(const aVersion,anEncoding,aRootElementName:TXmlString): IXmlElement;
22130:   Func CreateElement( aNameID : NativeInt) : IXmlElement;',);
22131:   Func CreateElement1(const aName : TXmlString) : IXmlElement;',);
22132:   Func CreateText(const aData : TXmlString) : IXmlText',);
22133:   Func CreateCDATASection( const aData : TXmlString) : IXmlCDATASection',);
22134:   Func CreateComment( const aData : TXmlString) : IXmlComment',);
22135:   Func CreateProcessingInstruction(const aTarget,aData: TXmlString): IXmlProcessingInstruction;',);
22136:   Func CreateProcessingInstruction1(aTargetID:NativeInt;const aData:TXmlString):IXmlProcessingInstruction);
22137:   Proc LoadXML( const aXML : RawByteString)',);
22138:   Proc LoadBinaryXML( const aXML : RawByteString)',);
22139:   Proc Load( aStream : TStream);',);
22140:   Proc Load1( const aFileName :Str);',);
22141:   Proc LoadResource( aType, aName : PChar)',);
22142:   Proc Save( aStream : TStream);',);
22143:   Proc Save1( const aFileName :Str);',);
22144:   Proc SaveBinary( aStream : TStream; anOptions : LongWord);',);
22145:   Proc SaveBinary1( const aFileName :Str; anOptions : LongWord);',);
22146:   end;
22147: end;
22148:
22149: Proc SIRegister_OmniXMLUtils(CL: TPSPascalCompiler);
22150: begin
22151:  // XmlString = WideString;
22152:  CL.AddTypeS('XmlString', 'WideString');
22153:  Func XMLStrToReal38( nodeValue : XmlString; var value : real) :Bool;
22154:  Func XMLStrToReal39( nodeValue : XmlString) : real;
22155:  Func XMLStrToRealDef( nodeValue : XmlString; defaultValue : real) : real');
22156:  Func XMLStrToExtended40( nodeValue : XmlString; var value : extended):Bool;
22157:  Func XMLStrToExtended41( nodeValue : XmlString) : extended;
22158:  Func XMLStrToExtendedDef( nodeValue : XmlString; defaultValue : extended): extended;
22159:  Func XMLStrToCurrency42( nodeValue : XmlString; var value : Currency):Bool;
22160:  Func XMLStrToCurrency43( nodeValue : XmlString) : Currency;
```

```
22161:  Func XMLStrToCurrencyDef( nodeValue : XmlString; defaultValue : Currency): Currency;
22162:  Func XMLStrToInt44( nodeValue : XmlString; var value : integer) :Bool;
22163:  Func XMLStrToInt45( nodeValue : XmlString) : integer;
22164:  Func XMLStrToIntDef( nodeValue : XmlString; defaultValue : integer) : integer');
22165:  Func XMLStrToInt6446( nodeValue : XmlString; var value : int64) :Bool;
22166:  Func XMLStrToInt6447( nodeValue : XmlString) : int64;
22167:  Func XMLStrToInt64Def( nodeValue : XmlString; defaultValue : int64) : int64');
22168:  Func XMLStrToBool48( nodeValue : XmlString; var value :Bool) :Bool;
22169:  Func XMLStrToBool49( nodeValue : XmlString) :Bool;
22170:  Func XMLStrToBoolDef( nodeValue : XmlString; defaultValue :Bool):Bool;
22171:  Func XMLStrToDateTime50( nodeValue : XmlString; var value : TDateTime):Bool;
22172:  Func XMLStrToDateTime51( nodeValue : XmlString) : TDateTime;
22173:  Func XMLStrToDateTimeDef(nodeValue : XmlString; defaultValue : TDateTime): TDateTime;
22174:  Func XMLStrToDate52( nodeValue : XmlString; var value : TDateTime):Bool;
22175:  Func XMLStrToDate53( nodeValue : XmlString) : TDateTime;
22176:  Func XMLStrToDateDef( nodeValue : XmlString; defaultValue : TDateTime) : TDateTime;
22177:  Func XMLStrToTime54( nodeValue : XmlString; var value : TDateTime):Bool;
22178:  Func XMLStrToTime55( nodeValue : XmlString) : TDateTime;
22179:  Func XMLStrToTimeDef( nodeValue : XmlString; defaultValue : TDateTime): TDateTime;
22180:  Func XMLStrToBinary( nodeValue : XmlString; const value : TStream):Bool');
22181:  Func XMLRealToStr( value : real; precision : byte) : XmlString');
22182:  Func XMLExtendedToStr( value : extended) : XmlString');
22183:  Func XMLCurrencyToStr( value : Currency) : XmlString');
22184:  Func XMLIntToStr( value : integer) : XmlString');
22185:  Func XMLInt64ToStr( value : int64) : XmlString');
22186:  Func XMLBoolToStr( value :Bool; useBoolStrs :Bool) : XmlString');
22187:  Func XMLDateTimeToStr2( value : TDateTime) : XmlString');
22188:  Func XMLDateTimeToStrEx( value : TDateTime) : XmlString');
22189:  Func XMLDateToStr2( value : TDateTime) : XmlString');
22190:  Func XMLTimeToStr( value : TDateTime) : XmlString');
22191:  Func XMLBinaryToStr( value : TStream) : XmlString');
22192:  Func XMLVariantToStr( value : Variant) : XmlString');
22193:  end;
22194:
22195: Proc SIRegister_IdCoderHeader(CL: TPSPascalCompiler);
22196: begin
22197:  CL.AddTypeS('TTransfer', ( bit7, bit8, iso2022jp );
22198: Func EncodeAddressItem(EmailAddr:TIdEmailAddressItem; const HeaderEncoding: Char;
22199:   TransferHeader: TTransfer; MimeCharSet:Str):Str;
22200: Func EncodeHeader(const Header:Str; specials : CSET2; const HeaderEncoding: Char;
22201:  TransferHeader: TTransfer; MimeCharSet:Str):Str;
22202: Func Encode2022JP(const S:Str):Str;
22203: Func EncodeAddress(EmailAddr:TIdEMailAddressList;const HeaderEncoding:Char;
22204:   TransferHeader: TTransfer; MimeCharSet:Str):Str;
22205: Func DecodeHeader(Header:Str):str;
22206: Func Decode2022JP(const S:Str):Str;
22207: Proc DecodeAddress(EMailAddr : TIdEmailAddressItem);
22208: Proc DecodeAddresses(AEMails :Str; EMailAddr : TIdEmailAddressList);
22209: Proc InitializeISO(var TransferHeader:TTransfer;var HeaderEncoding:char;var CharSet:str);
22210: end;
22211:
22212: Proc SIRegister_FannNetwork(CL: TPSPascalCompiler);
22213: begin
22214:  // CL.AddTypeS('PCardinalArray', '^CardinalArray // will not work');
22215:   CL.AddConstantN('DLL_FILE','String').SetString( 'fannfloat.dll');
22216:   CL.AddTypeS('fann_type', 'single');
22217:   Fann_Type_Array', 'array [0..65535] of fann_type');
22218:   TFann_Type_Array', 'array [0..65535] of fann_type');
22219:   TFann_Type_Array2', 'array of fann_type');
22220:   TFann_Type_Array3', 'array of single');
22221:   TTrainingAlgorithm', '( taFANN_TRAIN_INCREMENTAL, taFANN_TRAIN_B'
22222:    +ATCH, taFANN_TRAIN_RPROP, taFANN_TRAIN_QUICKPROP );
22223:   CL.AddTypeS(TActivationFunction', '( afFANN_LINEAR, afFANN_THRESHOLD, afFANN'
22224:    +'_THRESHOLD_SYMMETRIC, afFANN_SIGMOID, afFANN_SIGMOID_STEPWISE, afFANN_SIGM'
22225:    +'OID_SYMMETRIC, afFANN_SIGMOID_SYMMETRIC_STEPWISE, afFANN_GAUSSIAN, afFANN_'
22226:    +'GAUSSIAN_SYMMETRIC, afFANN_GAUSSIAN_STEPWISE, afFANN_ELLIOT, afFANN_ELLIOT'
22227:    +'_SYMMETRIC, afFANN_LINEAR_PIECE, afFANN_LINEAR_PIECE_SYMMETRIC, afFANN_SIN'
22228:    +'_SYMMETRIC, afFANN_COS_SYMMETRIC, afFANN_SIN, afFANN_COS );
22229:   SIRegister_TFannNetwork(CL);  //CL.AddDelphiFunction('Proc Register');
22230: end;
22231:
22232: Proc SIRegister_TFannNetwork(CL: TPSPascalCompiler);
22233: begin
22234:   //with RegClassS(CL,'TComponent', 'TFannNetwork') do
22235:  with CL.AddClassN(CL.FindClass('TComponent'),'TFannNetwork') do begin
22236:   Constructor Create( Aowner : TComponent);
22237:   Proc Build( );
22238:   Proc Free');
22239:   Proc UnBuild( );
22240:   Func Train( Input : array of fann_type; Output : array of fann_type) : single');
22241:   Proc TrainOnFile(FileName:str; MaxEpochs:Card;DesiredError: Single);
22242:   Proc Run(Inputs: array of fann_type; var Outputs : TFann_Type_Array2);
22243:   Proc Run2(Inputs: array of fann_type; var Outputs : array of fann_type);
22244:   Proc Run3( Inputs : array of fann_type; var Outputs : TFann_Type_Array3);
22245:   Proc Run4( Inputs : array of fann_type; var Outputs : TFann_Type_Array3);
22246:   Proc SaveToFile( FileName :Str);
22247:   Proc LoadFromFile( Filename :Str);
22248:   FannObject', 'PFann', iptr);
22249:   Layers', 'TStrings', iptrw);
```

```
22250:   LearningRate', 'Single', iptrw);
22251:   ConnectionRate', 'Single', iptrw);
22252:   LearningMometum', 'single', iptrw);
22253:   MSE', 'Single', iptr);
22254:   TrainingAlgorithm', 'TTrainingAlgorithm', iptrw);
22255:   ActivationFunctionHidden', 'TActivationFunction', iptrw);
22256:   ActivationFunctionOutput', 'TActivationFunction', iptrw);
22257:  end;
22258: end;
22259: TFannNetwork encapsulates the Fast Artificial Neural Network. fann.sourceforge.net
22260:
22261: Proc SIRegister_RTLDateTimeplus(CL: TPSPascalCompiler);
22262: begin
22263:   SIRegister_TNullableDateTime(CL);
22264:  Func IsValidDate2( const Value :Str; out ADate : TDatetime) :Bool;
22265:  Func IsValidTime2( const Value :Str; out ATime : TDateTime) :Bool;
22266:  Func IsValidDateTime2( const Value :Str; out ADateTime : TDateTime) :Bool;
22267:  Proc ChangeSystemTime( const Value : TDateTime);
22268:  Func TryChangeSystemTime( const Value : TDateTime) :Bool');
22269: end;
22270:
22271: // new functions on V 4.2.8.10  III /IV  - V 4.5.8.10 IV
22272: Func IsUTF8File(const FileName:Str):Bool;
22273: Func IsUTF16File(const FileName:Str):Bool;
22274: Proc RichEditToCanvas(RichEdit: TRichEdit; Canvas: TCanvas; PixelsPerInch:Integer);
22275: --> RichEditToCanvas(RichEdit1, Image1.Canvas, Self.PixelsPerInch);
22276: Func {$I ActiveDEP:Card; }
22277: Func {$I ActivateDEP:Card;}
22278: Proc WriteXMLFile(doc: TXMLDocument; const AFileName:Str);
22279: Proc WriteXML(Element: TDOMNode; const AFileName:Str);
22280: Func ReadXMLFile(const AFilename:Str) :  TXMLDocument;
22281: Proc WriteXMLFile(doc: TXMLDocumentDOM; const AFileName:Str);
22282: Proc WriteXML(Element: TDOMNode; const AFileName:Str);
22283: Proc WriteXMLFileStream(ADoc : TXMLDocument; AStream : TStream);
22284:
22285: Func CountColors(const Bitmap: Graphics.TBitmap): Integer;
22286: var  Flags: array[Byte, Byte] of Classes.TBits;
22287:   I, J, K: Integer;
22288:   RowIn: PRGBTripleArray;
22289: begin
22290:   // Be sure bitmap is 24-bits/pixel
22291:   Assert(Bitmap.PixelFormat = Graphics.pf24Bit);
22292:
22293: Func SndPlaySound(const AFilename: pchar; flags:Card):Bool;
22294: Proc PlaySound1(const AFilename:Str);
22295: Func mciSendCommand(mciID, umessage, dwparam1, dwparam2:Card):Card;
22296: Func ProxyActive:Bool;
22297: Func GetProxyServer:Str;
22298: Func GetProxyOverride:Str;
22299: Func GetProxyServerIP:Str;
22300: Func GetProxyServerPort: Integer;
22301: Func GetHtml( const Url :Str) :Str;
22302: Func GetHtml2( const Url, UserAgent :Str) :Str;
22303: Proc FormatXMLFile(const XmlFile:str);
22304: Func FormatXMLFile2String(const XmlFile:str):Str;
22305: Func selectXMLNode(xnRoot: IXmlNode; const nodePath: WideString): IXmlNode;
22306: Func HTMLEncode3(const Data:Str):Str; //vfast
22307: Func IsSpecialFolderSupported(CSIDL: Integer):Bool;
22308: Func SpecialFolderPath(CSIDL: Integer):Str;
22309: Func DataSetToXML(const ADataSet: TDataSet):Str;
22310: Func PackageParams( Params : TParams; Types : TParamTypes) : OleVariant');
22311: Proc UnpackParams( const Source : OleVariant; Dest : TParams);
22312: Proc ActivateDEP2;
22313: Func JavaToDelphiDateTime(const dt: int64): TDateTime;
22314: Func DelphiToJavaDateTime(const dt: TDateTime): int64;
22315: Func GetTimeBias: integer;
22316: Proc WrapLines(VAR a, b:Str; len: Integer);
22317: Func DelphiDateTimeToISO8601Date(dt: TDateTime):Str;
22318: Proc BackgroundWorkerWaitFor(isworking:Bool; fWindow: HWND);
22319:
22320: uPSI_CromisStreams,
22321: uPSI_Streams,
22322: uPSI_BitStream,
22323: uPSI_UJSONFunctions.pas
22324: uPSI_uTPLb_BinaryUtils.pas
22325: uPSI_USha256.pas
22326: uPSI_uTPLb_HashDsc.pas
22327: uPSI_uTPLb_Hash.pas
22328: SIRegister_Series(X);  //4.2.6.10
22329: uPSI_Series,
22330: unit uPSI_mimeinln;
22331: unit uPSI_UTime; //unit UTime;
22332: RIRegister_TStreamStorage(X);
22333: RIRegister_TNamesEnumerator(X);
22334: 1283 uPSI_uTPLb_StreamCipher.pas
22335: 1284 uPSI_uTPLb_BlockCipher.pas
22336: 1285 uPSI_uTPLb_Asymetric.pas
22337: 1286 uPSI_uTPLb_CodecIntf.pas
22338: 1287 uPSI_uTPLb_Codec.pas
```

```
22339: 1288 uPSI_ADOInt.pas
22340: 1289 uPSI_MidasCon.pas
22341: 1290 uPSI_XMLDoc.pas
22342: 1291 uPSI_XMLIntf.pas
22343: 1292 uPSI_ProxyUtils.pas
22344: 1293 unit uPSI_maxXMLUtils2;
22345: 1294 unit_StDict_Routines(S: TPSExec);
22346: 1295 unit uPSI_Hashes2
22347: 1296 unit uPSI_IdCoderHeader;
22348:
22349: Func TRestRequest_createStringStreamFromStringList(strings:TStringList):TStringStream;
22350: {A simple Oscilloscope using  TWaveIn class.
22351:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
22352:  http://www.retroarchive.org/garbo/pc/turbopas/index.html
22353: uses Forms,
22354:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
22355:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
22356:   uColorFunctions in 'uColorFunctions.pas',
22357:   AMixer in 'AMixer.pas',
22358:   uSettings in 'uSettings.pas',
22359:   UWavein4 in 'UWavein4.pas',
22360:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
22361:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
22362:   BSpectrum in unit uPSI_BSpectrum; {SpectraLibrary}
22363:   1250 unit uPSI_U_Oscilloscope4_2
22364:
22365: // new functions on V 4.6.3.10
22366: Proc SIRegister_cHash(CL: TObject);
22367: var sha: TSha256hash2;
22368: begin
22369:  DigestToHexBuf( const Digest :Str; const Size : Integer; const Buf);
22370:  DigestToHex( const Digest :Str; const Size : Integer) : Ansistr');
22371:  Digest128Equal( const Digest1, Digest2 : T128BitDigest) :Bool');
22372:  Digest160Equal( const Digest1, Digest2 : T160BitDigest) :Bool');
22373:  Digest224Equal( const Digest1, Digest2 : T224BitDigest) :Bool');
22374:  Digest256Equal( const Digest1, Digest2 : T256BitDigest) :Bool');
22375:  Digest384Equal( const Digest1, Digest2 : T384BitDigest) :Bool');
22376:  Digest512Equal( const Digest1, Digest2 : T512BitDigest) :Bool');
22377:    hashNoError','LongInt').SetInt( 0);
22378:    hashInternalError','LongInt').SetInt( 1);
22379:    hashInvalidHashType','LongInt').SetInt( 2);
22380:    hashInvalidBuffer','LongInt').SetInt( 3);
22381:    hashInvalidBufferSize','LongInt').SetInt( 4);
22382:    hashInvalidDigest','LongInt').SetInt( 5);
22383:    hashInvalidKey','LongInt').SetInt( 6);
22384:    hashInvalidFileName','LongInt').SetInt( 7);
22385:    hashFileOpenError','LongInt').SetInt( 8);
22386:    hashFileSeekError','LongInt').SetInt( 9);
22387:    hashFileReadError','LongInt').SetInt( 10);
22388:    hashNotKeyedHashType','LongInt').SetInt( 11);
22389:    hashTooManyOpenHandles','LongInt').SetInt( 12);
22390:    hashInvalidHandle','LongInt').SetInt( 13);
22391:    hashMAX_ERROR','LongInt').SetInt( 13);
22392:  GetHashErrorMessage( const ErrorCode : LongWord) : PChar');
22393: /SIRegister_EHashError(CL);
22394:  SecureClear( var Buf :Str; const BufSize : Integer);
22395:  SecureClear512( var Buf : T512BitBuf);
22396:  SecureClear1024( var Buf : T1024BitBuf);
22397:  SecureClearStr2( var S : Ansistr);      --------------------
22398:  CalcChecksum32( const Buf : Ansistr) : LongWord');
22399:  CalcXOR8( const Buf : Ansistr) : Byte');
22400:  CalcXOR16( const Buf : Ansistr) : Word');
22401:  CalcXOR32( const Buf : Ansistr) : LongWord');
22402:  CRC16Init( var CRC16 : Word);
22403:  CRC16Byte( const CRC16 : Word; const Octet : Byte) : Word');
22404:  CRC16Buf( const CRC16 : Word; const Buf :Str; const BufSize : Integer) : Word');
22405:  CalcCRC16( const Buf : Ansistr) : Word');
22406:  SetCRC32Poly( const Poly : LongWord);
22407:  CRC32Init( var CRC32 : LongWord);
22408:  CRC32Byte( const CRC32 : LongWord; const Octet : Byte) : LongWord');
22409:  CRC32Buf( const CRC32 : LongWord; const Buf :Str; const BufSize : Integer): LongWord');
22410:  CRC32BufNoCase(const CRC32: LongWord; const Buf:Str; const BufSize: Integer): LongWord');
22411:  CalcCRC32( const Buf : Ansistr) : LongWord');
22412:  Adler32Init( var Adler32 : LongWord);
22413:  Adler32Byte( const Adler32 : LongWord; const Octet : Byte) : LongWord');
22414:  Adler32Buf( const Adler32:LongWord;const Buf:str; const BufSize:Integer): LongWord');
22415:  CalcAdler32( const Buf : Ansistr) : LongWord');
22416:  ELFInit( var Digest : LongWord);
22417:  ELFBuf( const Digest : LongWord; const Buf :Str; const BufSize : Integer):LongWord');
22418:  CalcELF( const Buf : Ansistr) : LongWord');
22419:  IsValidISBN2( const S : Ansistr) :Bool'); -------------------------
22420:  IsValidLUHN( const S : Ansistr) :Bool');
22421:  KnuthHash( const S : Ansistr) : LongWord');
22422:  MD5InitDigest( var Digest : T128BitDigest);
22423:  MD5Buf( var Digest : T128BitDigest; const Buf :Str; const BufSize : Integer);
22424:  MD5FinalBuf(var Digest:T128BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22425:  CalcMD5( const Buf : Ansistr) : T128BitDigest');
22426:  MD5DigestAsString( const Digest : T128BitDigest) : Ansistr');
22427:  MD5DigestToHex( const Digest : T128BitDigest) : Ansistr');
```

```
22428:  SHA1InitDigest( var Digest : T160BitDigest);
22429:  SHA1Buf( var Digest : T160BitDigest; const Buf :Str; const BufSize : Integer);
22430:  SHA1FinalBuf(var Digest:T160BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22431:  CalcSHA1( const Buf : Ansistr) : T160BitDigest');
22432:  SHA1DigestAsString( const Digest : T160BitDigest) : Ansistr');
22433:  SHA1DigestToHex( const Digest : T160BitDigest) : Ansistr');
22434:  SHA224InitDigest( var Digest : T256BitDigest);
22435:  SHA224Buf( var Digest : T256BitDigest; const Buf:Str;const BufSize : Integer);
22436: SHA224FinalBuf(var Digest:T256BitDigest;const Buf:str;const BufSize:Integer;const TotalSize:Int64;var
       OutDigest:T224BitDigest);
22437:   CalcSHA224( const Buf : Ansistr) : T224BitDigest');
22438:  SHA224DigestAsString( const Digest : T224BitDigest) : Ansistr');
22439:  SHA224DigestToHex( const Digest : T224BitDigest) : Ansistr');
22440:  SHA256InitDigest( var Digest : T256BitDigest);
22441:  SHA256Buf( var Digest : T256BitDigest; const Buf :Str; const BufSize : Integer);
22442:  SHA256FinalBuf(var Digest:T256BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64;
22443:  CalcSHA2562( const Buf : Ansistr) : T256BitDigest');  -------------------------
22444:  SHA256DigestAsString( const Digest : T256BitDigest) : Ansistr');
22445:  SHA256DigestToHex( const Digest : T256BitDigest) : Ansistr');
22446:  SHA384InitDigest( var Digest : T512BitDigest);
22447:  SHA384Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22448:  SHA384FinalBuf(var Digest:T512BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64;var
       OutDigest:T384BitDigest);
22449:  CalcSHA384( const Buf : Ansistr) : T384BitDigest');
22450:  SHA384DigestAsString( const Digest : T384BitDigest) : Ansistr');
22451:  SHA384DigestToHex( const Digest : T384BitDigest) : Ansistr');
22452:  SHA512InitDigest( var Digest : T512BitDigest);
22453:  SHA512Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22454:  SHA512FinalBuf(var Digest:T512BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22455:  CalcSHA512( const Buf : Ansistr) : T512BitDigest');
22456:  SHA512DigestAsString( const Digest : T512BitDigest) : Ansistr');
22457:  SHA512DigestToHex( const Digest : T512BitDigest) : Ansistr');
22458:  HMAC_MD5Init(const Key:str;const KeySize:Integer;var Digest:T128BitDigest;var K:T512BitBuf);
22459:  HMAC_MD5Buf( var Digest : T128BitDigest; const Buf :Str; const BufSize : Integer);
22460:  HMAC_MD5FinalBuf(const K:T512BitBuf;var Digest T128BitDigest;const Buf:str;const BufSize:Integer;const
       TotalSize:Int64);
22461:  CalcHMAC_MD5( const Key, Buf : Ansistr) : T128BitDigest');
22462:   HMAC_SHA1Init(const Key:str;const KeySize:Int;var Digest:T160BitDigest;var K:T512BitBuf);
22463:   HMAC_SHA1Buf( var Digest : T160BitDigest; const Buf :Str; const BufSize : Integer);
22464: HMAC_SHA1FinalBuf(const K:T512BitBuf;ar Digest:T160BitDigest;const Buf:Str;const BufSize:Int;const
       TotalSize Int64);
22465:  CalcHMAC_SHA1( const Key, Buf : Ansistr):T160BitDigest;
22466:  HMAC_SHA256Init(const Key:str;const KeySize:Int;var Digest:T256BitDigest;var K:T512BitBuf);
22467:  HMAC_SHA256Buf( var Digest : T256BitDigest; const Buf :Str; const BufSize : Integer);
22468:  HMAC_SHA256FinalBuf(const K :T512BitBuf;var Digest: T256BitDigest;const Buf:Str; const
       BufSize:Integer;const TotalSize:Int64);
22469:  CalcHMAC_SHA256( const Key, Buf : Ansistr) : T256BitDigest');
22470:  HMAC_SHA512Init(const Key:str;const KeySize:Int;var Digest:T512BitDigest;var K:T1024BitBuf);
22471:  HMAC_SHA512Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22472:  HMAC_SHA512FinalBuf(const K:T1024BitBuf;var Digest:T512BitDigest;const Buf:str;const
       BufSize:Integer;const TotalSize:Int64);
22473:  CalcHMAC_SHA512( const Key, Buf: Ansistr): T512BitDigest');
22474:  AHash(CL);
22475:  CL.AddTypeS('THashClass', 'class of AHash');
22476:  TChecksum32Hash(CL);
22477:   TXOR8Hash(CL);
22478:   TXOR16Hash(CL);
22479:   TXOR32Hash(CL);
22480:   TCRC16Hash(CL);
22481:   TCRC32Hash(CL);
22482:   TAdler32Hash(CL);
22483:   TELFHash(CL);
22484:   TMD5Hash(CL);
22485:   TSHA1Hash(CL);
22486:   TSHA256Hash2(CL);  //------------!!
22487:   TSHA512Hash(CL);
22488:   THMAC_MD5Hash(CL);
22489:   THMAC_SHA1Hash(CL);
22490:   THMAC_SHA256Hash(CL);
22491:   THMAC_SHA512Hash(CL);
22492:   THashType','(hashChecksum32,hashXOR8,hashXOR16,hashXOR32,hashCRC16,hashCRC32,hashAdler32, hashELF,
       hashMD5,hashSHA1 hashSHA256,hashSHA512,hashHMAC_MD5,hashHMAC_SHA1,hashHMAC_SHA256,hashHMAC_SHA512'+');
22493:  GetHashClassByType( const HashType : THashType) : THashClass');
22494:  GetDigestSize( const HashType : THashType) : Integer');
22495:  CalculateHash(const HashType:THashType;const Buf:AnsiStr;const Digest:str;const Key:AnsiStr;
22496:  HashString(const S: Ansistr; const Slots:LongWord; const CaseSensitive:Boolean):LongWord');
22497:  SelfTestHash');
22498: end;
22499:
22500: Unit UMatrix
22501: Proc Determinant( Dimen : integer; Data : TNmatrix; var Det : Float; var Error : byte);
22502: Proc Inverse2( Dimen : integer; Data : TNmatrix; var Inv : TNmatrix; var Error : byte);
22503:  Proc Gaussian_Elimination(Dimen:integer;Coefficients:TNmatrix;Constants:TNvector;var
       Solution:TNvector;var Error:byte);
22504:  Proc Partial_Pivoting(Dimen:integer;Coefficients:TNmatrix;Constants:TNvector;var Solution:TNvector;var
       Error:byte);
22505:  Proc LU_Decompose(Dimen:integer;Coefficients:TNmatrix;var Decomp:TNmatrix;var Permute:TNmatrix;var
       Error:byte);
22506:  Proc LU_Solve(Dimen:integer;var Decomp TNmatrix;Constants:TNvector;var Permute:TNmatrix;var
       Solution:TNvector;var Error:byte);
```

```
22507:  Proc Gauss_Seidel( Dimen: integer; Coefficients: TNmatrix; Constants: TNvector; Tol: Float; MaxIter:
        integer;var Solution:TNvector;var Iter:integer;var Error: byte);
22508: end;
22509:
22510: Proc SIRegister_uTPLb_RSA_Primitives(CL: TPSPascalCompiler);
22511: begin
22512:  //CL.AddDelphiFunction('Func I2OSP( x : THugeCardinal; xLen : integer; XStream : TStream; Pool :
        TMemoryStreamPool) :Bool');
22513:  //'Func OS2IP( XStream : TStream; xLen : integer; var x : THugeCardinal; Pool : TMemoryStreamPool;
        MaxBits : integer) :Bool');
22514:  Proc MGF1( mgfSeed : TStream; maskLen :Card; mask : TStream);
22515:   CL.AddTypeS('TLongOpResult', '( opPass, opFail, opAbort );
22516:  Func RSAES_OAEP_ENCRYPT( n, e : THugeCardinal; M, C : TMemoryStream) :Bool');
22517:  Func RSAES_OAEP_ENCRYPT_MaxByteLen( n : THugeCardinal) : integer');
22518:  Func RSAES_OAEP_DECRYPT( d, n : THugeCardinal; C, M : TStream) :Bool');
22519:  Func EMSA_PSS_ENCODE(M:TStream;emBits,
        sLen:integer;EM:TStream;CheckAbortFunc:TOnHashProgress):TLongOpResult;
22520:  Func RSASSA_PSS_SIGN(d,n:THugeCardinal;M,S:TStream;CheckAbortFunc:TOnHashProgress): TLongOpResult;
22521:  Func EMSA_PSS_VERIFY(M:TStream;emBits,sLen integer;EM:TStream;
        CheckAbortFunc:TOnHashProgress):TLongOpResult);
22522:  Func RSASSA_PSS_VERIFY(n,e THugeCardinal; M, S : TStream; CheckAbortFunc : TOnHashProgress) :
        TLongOpResult;
22523:  Func Generate_RSA_SymetricKey(n,e:THugeCardinal;CipherStream: TStream;const SymetricCipher:IBlockCipher)
        : TSymetricKey;
22524:  Func Extract_RSA_SymetricKey(d,n:THugeCardinal;CipherStream: TStream;const
        SymetricCipher:IBlockCipher):TSymetricKey;
22525: end;
22526:
22527: Proc SIRegister_DXUtil(CL: TPSPascalCompiler);
22528: begin
22529:  Proc SAFE_RELEASE( var i);
22530:  Proc SAFE_DELETE( var Obj);
22531:  Func DXUtil_GetDXSDKMediaPathCb( szDest : PChar; cbDest : Integer) : HRESULT');
22532:  Func DXUtil_FindMediaFileCch(strDestPath:PChar;cchDest:Integer;strFilename:PChar):HRESULT;
22533:  Func DXUtil_FindMediaFileCb(szDestPath: PChar; cbDest Integer;strFilename PChar):HRESULT'
22534:  Func DXUtil_WriteStringRegKey(hKey_ :HKEY; strRegName : PChar; strValue : PChar):HRESULT;
22535:   CL.AddTypeS('TIMER_COMMAND', 'DWORD');
22536:  'TIMER_RESET','LongInt').SetInt( 0);
22537:  'TIMER_START','LongInt').SetInt( 1);
22538:  'TIMER_STOP','LongInt').SetInt( 2);
22539:  'TIMER_ADVANCE','LongInt').SetInt( 3);
22540:  'TIMER_GETABSOLUTETIME','LongInt').SetInt( 4);
22541:  'TIMER_GETAPPTIME','LongInt').SetInt( 5);
22542:  'TIMER_GETELAPSEDTIME','LongInt').SetInt( 6);
22543:  Func DXUtil_Timer( command : TIMER_COMMAND) : Single');
22544:  Func DXUtil_ConvertGenericStringToAnsiCch(strDestination:PAnsiChar;const
        tstrSource:PChar;cchDestChar:Intr): HRESULT');
22545:  Func DXUtil_ConvertAnsistrToGenericCch(tstrDestination:PChar;const strSource:PAnsiChar; cchDestChar:Int):
        HRESULT');
22546:  Func DXUtil_ConvertGenericStringToAnsiCb(strDestination:PAnsiChar;const tstrSource:PChar;cbDestChar:Int:
        HRESULT');
22547:  Func DXUtil_ConvertAnsistrToGenericCb(tstrDestination:PChar;const
        strSource:PChar;cbDestChar:Integer):HRESULT;
22548:  CL.AddTypeS(TArrayListType', '( AL_VALUE, AL_REFERENCE );
22549:  SIRegister_CArrayList(CL);
22550:  Func GETTIMESTAMP : DWORD');
22551: end;
22552:
22553: Proc SIRegister_DCPbase64(CL: TPSPascalCompiler);
22554: begin
22555: Func Base64EncodeStr(const Value: Ansistr): Ansistr');
22556: Func Base64DecodeStr(const Value: Ansistr): Ansistr');
22557: Func Base64Encode( pInput :Str; pOutput :Str; Size : longint): longint;
22558: Func Base64Decode( pInput :Str; pOutput :Str; Size : longint): longint;
22559: end;
22560:
22561: Proc SIRegister_FlyFilesUtils(CL: TPSPascalCompiler);
22562: begin
22563:  Func GetCaseSensitiveFileName( const FileName :Str; RootPath :Str) :Str');
22564:  CL.AddConstantN('OTGDeivceCount','LongInt').SetInt( 16);
22565:  CL.AddConstantN('UsbDiskStartIndex','LongInt').SetInt( 255);
22566:  DeleteDirectories_WaitMinSecond','LongInt').SetInt( 2000);
22567:  Func isPathCanUseNow( const PathOrDir :Str; const Default :Bool) :Bool');
22568:  Func TestPathCanWrite( const PathOrDir :Str) :Bool');
22569:  Func GetSDCardPath( Index : Integer) :Str');
22570:  Func FindSDCardSubPath( SubPath :Str; Index : Integer) :Str');
22571:  Func GetAppPath :Str');
22572:  Func BuildFileListInAPath(const Path:str;const Attr:Int;const List:TStrings;JustFile:Bool):Bool;
22573:  Func BuildFileListInAPath1(const Path:str; const Attr:Integer;ustFile:Bool):Str;
22574:  Func GetFileNamesFromDirectory(const DirName:str;const SearchFilter:str;const FileAttribs:Int;const
        isIncludeSubDirName:Bool;const Recursion:Bool;const FullName:Bool):str;
22575:  Func DeleteDirectoryByEcho(const Source:str;AbortOnFailure:Bool;YesToAll:Bool;WaitMinSecond:Integer):Bool;
22576:  Func GetTotalSpaceSize( Path :Str) : UInt64');
22577:  Func GetAvailableSpaceSize( Path :Str) : UInt64');
22578:  Func GetFreeSpaceSize( Path :Str) : UInt64');
22579:  Func GetTotalMemorySize : UInt64');
22580:  Func GetFreeMemorySize : UInt64');
22581:  Func IsPadOrPC :Bool');
22582:  Func OpenFileOnExtApp(const FileName:str;Https:Bool):Bool');
```

```
22583:  Func NowGMT_UTC : TDateTime');
22584:  Func EncodeURLWithSchemeOrProtocol( const URL :Str):Str');
22585:  Func GetVolumePaths :Str');
22586:  Func GetExternalStoragePath :Str');
22587:  Func GetExterStoragePath :Str');
22588:  Func GetInnerStoragePath :Str');
22589:  Func GetIsExternalStorageRemovable :Bool');
22590:  Func GetScreenClientInches : Single');
22591:  Func IsCanFindJavaClass( const NamePath :Str) :Bool');
22592:  Func IsCanFindJavaMethod(const MethodName,Signature:str;const CalssNamePath:str):Bool;
22593:  Func IsCanFindJavaStaticMethod(const MethodName,Signature:str;const CalssNamePath:Str):Bool;
22594:   CL.AddTypeS(TGetFileNameLIsternerMethod','Procedure(const IsOK:Boolean;const FileName:str);
22595:  Func OpenFileDialog(Title,FileExtension:str; GetFileNameCallBack: TGetFileNameLIsternerMethod):Boolean;
22596:  Func CheckPermission( const APermissionName :Str) :Bool');
22597:  CL.AddConstantN('C_android_permission_EXTERNAL_STORAGE',
       'String').SetString('android.permission.WRITE_EXTERNAL_STORAGE');
22598:  Func CanWriteExterStorage :Bool');
22599:  Proc UpdateAlbum( FileNames :Str);
22600:  Func ReadNoSizeFileToString( const AFileName :Str) :Str');
22601:  Func ReadFileToString( const AFileName :Str) :Str');
22602: end;
22603:
22604: 466_kmean_cluster_correlation_test3.txt
22605:  Proc SIRegister_LatLonDist(CL: TPSPascalCompiler);
22606: begin
22607:  Func EllipticalDistance(llat1,llon1,llat2,llon2:extended;units:integer): extended');
22608:  Proc VDirectLatLon(DGLAT1,DGLON1,DFAZ,S EXTENDED;Units:integer;var DGLAT2,DGLON2,DBAZ : EXTENDED);
22609:  Proc VInverseDistance(dlat1,dlon1,dlat2,dlon2:ext;var AzimuthInit,AzimuthFinal:ext;units:int):ext;
22610:  Func RhumbDistance(llat1,llon1,llat2,llon2:extended;units:integer;var Azimuth:extended):extended;
22611:  Proc RhumbLatLon(LAT1,LON1 Azimuth1,Dist:EXTENDED;Units:integer;var LAT2,LON2:EXTENDED);
22612:  Func ApproxEllipticalDistance(llat1,llon1,llat2,llon2: extended;units:integer):extended;
22613:  Func ApproxRhumbDistance(llat1,llon1,llat2,llon2: extended;units:integer; var Azimuth: extended):
       extended'
22614:  Proc ApproxRhumbLatLon(LAT1,LON1,Azimuth1,Dist:EXTENDED;Units integer;var LAT2,LON2: EXTENDED);
22615:  Func convertunits( fromindex,ToIndex:integer;fromvalue:extended):extended');
22616: end;
22617:
22618: Proc SIRegister_PXLTiming(CL: TPSPascalCompiler);
22619: begin
22620: CL.AddConstantN('OverrunDeltaLimit','Extended').setExtended( 8.0);
22621:  TTimerEvent', 'Proc ( const Sender : TObject);
22622:  SIRegister_TMultimediaTimer2(CL);
22623:  CL.AddTypeS(TSystemTimerValue, 'UInt64');
22624:  Func TimerValueInBetween(const Value1,Value2: TSystemTimerValue):TSystemTimerValue;
22625:  Func TickCountInBetween( const Value1, Value2:Card) :Card');
22626:  Func GetSystemTimerValue: TSystemTimerValue');
22627:  Func GetSystemTickCount:Card');
22628:  Func GetSystemTimeValue: Double');
22629:  Proc MicroSleep( const Microseconds : UInt64);
22630: end;
22631:
22632: (* === compile-time registration functions === *)
22633: (*----------------------------------------------------------------------------*)
22634: Proc SIRegister_TMultimediaTimer2(CL: TPSPascalCompiler);
22635: begin
22636: //with RegClassS(CL,'TOBJECT', 'TMultimediaTimer') do
22637: type  TMultimediaTimer2 = PXLTiming.TMultimediaTimer;
22638:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TMultimediaTimer2') do begin
22639:   Constructor Create');
22640:   Proc Process');
22641:   Proc Reset');
22642:   Proc NotifyTick( AllowSleep :Bool);
22643:   Delta', 'Double', iptr);
22644:   Latency', 'Double', iptr);
22645:   FrameRate', 'Integer', iptr);
22646:   Speed', 'Double', iptrw);
22647:   MaxFPS', 'Integer', iptrw);
22648:   Enabled', 'Boolean', iptrw);
22649:   SingleCallOnly', 'Boolean', iptrw);
22650:   OnTimer', 'TTimerEvent', iptrw);
22651:   OnProcess', 'TTimerEvent', iptrw);
22652:  end;
22653: end;
22654:
22655: Proc SIRegister_TNetCom(CL: TPSPascalCompiler);
22656: begin
22657:   //with RegClassS(CL,'TOBJECT', 'TNetCom') do
22658:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TNetCom') do begin
22659:   Constructor Create');
22660:   Proc Free');
22661:   Func Initialize :Bool');
22662:   Proc Finalize');
22663:   Func ResolveHost( const Host :Str) :Str');
22664:   Func ResolveIP( const IPAddress :Str) :Str');
22665:   Func SendStr(const Host:Str;const Port:Integer;const Data:str;const Size:Int):Bool;
22666:   Func Send2(const Host:str;const Port:Integer;const Data:str;const Size:Int):Bool;
22667:   Func Send(const Host:str;const Port:Integer;const Data:str;const Size:Int):Bool;
22668:   Proc Update');
22669:   Proc ResetStatistics');
```

```
22670:    LocalIP', 'String', iptr);
22671:    Initialized', 'Boolean', iptr);
22672:    Broadcast', 'Boolean', iptrw);
22673:    LocalPort', 'Integer', iptrw);
22674:    /('OnReceive', 'TReceiveEvent', iptrw);
22675:    UpdateRefreshTime', 'Integer', iptrw);
22676:    BytesReceived', 'Integer', iptr);
22677:    BytesSent', 'Integer', iptr);
22678:    SentPackets', 'Integer', iptr);
22679:    ReceivedPackets', 'Integer', iptr);
22680:    BytesPerSec', 'Integer', iptr);
22681:   end;
22682: end;
22683:
22684: (*----------------------------------------------------------------------------*)
22685: Proc SIRegister_PXLNetComs(CL: TPSPascalCompiler);
22686: begin
22687:   CL.AddConstantN('MaximumPacketSize','LongInt').SetInt( 8166);
22688:   CL.AddTypeS('TSocket', 'Integer');   //delphisocks as longint
22689:   CL.AddTypeS('TInAddr', 'record S_addr :Card; end');
22690:   SIRegister_TNetCom(CL);
22691: end;
22692:
22693: unit uPSI_cTCPConnection.pas
22694: Proc RIRegister_cTCPConnection_Routines(S: TPSExec);
22695: Func TCPGetTick : LongWord');
22696: Func TCPTickDelta( const D1, D2 : LongWord) : Integer');
22697: Func TCPTickDeltaW( const D1, D2 : LongWord) : LongWord');
22698:
22699: Proc SIRegister_cTCPConnection(CL: TPSPascalCompiler);
22700: begin
22701:   CL.AddClassN(CL.FindClass('TOBJECT'),'ETCPConnection');
22702:   CL.FindClass('TOBJECT'),'TTCPConnection');
22703:   CL.AddTypeS('TTCPLogType', '( tlDebug, tlInfo, tlError );
22704:   TTCPConnectionProxyState,(prsInit,prsNegotiating,prsFiltering,prsFinished,prsError,prsClosed);
22705:   SIRegister_TTCPConnectionProxy(CL);
22706:   SIRegister_TTCPConnectionProxyList(CL);
22707:   TTCPConnectionState','(cnsInit,cnsProxyNegotiation,cnsConnected,cnsClosed );
22708: TTCPConnectionTransferState,record LastUpdate:LongWord;ByteCount:Int64;TransferRate:LongWord; end;
22709:   TAnsiCharSet', 'set of AnsiChar');
22710:   TTCPConnectionNotifyEvent', 'Proc ( Sender : TTCPConnection);
22711:   TTCPConnectionStateChangeEvent', 'Proc ( Sender : TTCPConnection; State: TTCPConnectionState);
22712:  TTCPConnectionLogEvent','Procedure(Sender:TTCPConnection;LogType:TTCPLogType;LogMsg:str; LogLevel:Int;
22713:   SIRegister_TTCPConnection(CL);
22714:   //CL.AddTypeS('TTCPConnectionClass', 'class of TTCPConnection');
22715:  Func TCPGetTick : LongWord');
22716:  Func TCPTickDelta( const D1, D2 : LongWord) : Integer');
22717:  Func TCPTickDeltaW( const D1, D2 : LongWord) : LongWord');
22718: end;
22719:
22720: Proc RIRegister_cTCPConnection_Routines(S: TPSExec);
22721: begin
22722:  S.RegisterDelphiFunction(@TCPGetTick, 'TCPGetTick', cdRegister);
22723:  S.RegisterDelphiFunction(@TCPTickDelta, 'TCPTickDelta', cdRegister);
22724:  S.RegisterDelphiFunction(@TCPTickDeltaW, 'TCPTickDeltaW', cdRegister);
22725: end;
22726:
22727: var HTMLWindow2: IHTMLWindow2;
22728: const HTMLStrGoogle   'string').SetString(HTMLStrGoogle);     //javascript
22729:
22730: Proc GoogleMapsFormCreate(Sender:TObject;aform:TForm;webbrowser1:TWebbrowser;HTMLStr:str);
22731: Proc GoogleMapsGotoLocationClick(Sender: TObject; LatitudeText, LongitudeText:Str );
22732: begin
22733:   HTMLWindow2.execScript(Format('GotoLatLng(%s,%s)',[LatitudeText,LongitudeText]),'JavaScript');
22734: end;
22735:
22736: Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress:Str);
22737: var address :Str;
22738: begin
22739:   address:= vaddress;
22740:   address:=StringReplace(StringReplace(Trim(address),#13,' ',[rfReplaceAll]),#10,' ',[rfReplaceAll]);
22741:   HTMLWindow2.execScript(Format('codeAddress(%s)',[QuotedStr(address)]), 'JavaScript');
22742: end;
22743:  Func TaskbarHandle: THandle');
22744:  Func TrayHandle: THandle');
22745:  Proc DOSWipeFile(fn:Str);
22746:  Proc DOSFileWipe(fn:Str);
22747:  Func ExecAndWait(sExe, sCommandLine:Str):Bool;
22748:  Proc GoogleMapsFormCreate(Sender:TObject;aform:TForm; webbrowser1:TWebbrowser;HTMLStr,navdoc:str);
22749:  Proc GoogleMapsGotoLocationClick(Sender:TObject;LatitudeText,LongitudeText:Str );
22750:  Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress:Str);
22751:  Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress:Str);
22752:  Proc BitmapToMetafile(const Bmp: {Graphics.}TBitmap; const EMF: {Graphics.}TMetafile);
22753:
22754: Proc SIRegister_DrBobCGI(CL: TPSPascalCompiler);
22755: begin
22756:   CL.AddTypeS('TBobRequestMethod','(bobUnknown, bobGet, bobPost );
22757:  Func bobValue( const Field : ShortString; Convert :Bool) : ShortString');
22758:  Func CookieValue( const Field : ShortString) : ShortString');
```

```
22759:  Func getCGIEnvValues:Str;
22760: end;
22761:
22762: Func getCGIEnvValues:Str;
22763:   begin
22764:     result:= inttoStr(ContentLength)+';'+RemoteAddress+';'+HttpUserAgent+';'+
22765:     Authorization+';'+ScriptName+'.'
22766:   end;
22767:
22768: Proc SIRegister_OverbyteIcsCharsetUtils(CL: TPSPascalCompiler);
22769: begin
22770:  CL.AddConstantN('ERR_CP_NOTMAPPED','LongInt').SetInt( MAX_CODEPAGE + 1);
22771:  'ERR_CP_NOTAVAILABLE','LongInt').SetInt( MAX_CODEPAGE + 2);
22772:   ^'CP_US_ASCII','LongInt').SetInt( 20127);
22773:   CL.AddTypeS('CsuString', 'String');
22774:   CL.AddTypeS('TMimeCharset', '( CS_DEFAULT, CS_NOTMAPPED, UTF_8, WIN_1250, WIN'
22775:    +'_1251, WIN_1252, WIN_1253, WIN_1254, WIN_1255, WIN_1256, WIN_1257, WIN_125'
22776:    +'8, ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6,'
22777:    +' ISO_8859_7, ISO_8859_8, ISO_8859_8_i, ISO_8859_9, ISO_8859_13, ISO_8859_1'
22778:    +'5, ISO_2022_JP, ISO_2022_JP_1, ISO_2022_JP_2, ISO_2022_KR, ISO_2022_CN, X_'
22779:    +'CP50227, EUC_JP, GB_2312_80, GB_2312, HZ_GB_2312, GB_18030, EUC_CN, KOI8_R'
22780:    +', KOI8_U, UTF_16LE, UTF_16BE, UTF_7, SHIFT_JIS, BIG_5, KOREAN_HANGUL, EUC_'
22781:    +'KR, WIN_874, IBM_037, IBM_437, IBM_500, IBM_850, IBM_852, IBM_855, IBM_857'
22782:    +', IBM_00858, IBM_860, IBM_861, IBM_862, IBM_863, IBM_864, IBM_865, IBM_866'
22783:    +', IBM_869, IBM_870, IBM_1026, IBM_01047, IBM_01140, IBM_01141, IBM_01142, '
22784:    +'IBM_01143, IBM_01144, IBM_01145, IBM_01146, IBM_01147, IBM_01148, IBM_0114'
22785:    +'9, MACINTOSH, UTF_32LE, UTF_32BE, US_ASCII, T_61, CS_LAST_ITEM );
22786:   CL.AddTypeS('TMimeCharsets', 'set of TMimeCharset');
22787:   CL.AddTypeS('_cpinfoexA', record MaxCharSize:UINT; DefaultChar:array[0..2 - 1] of Byte;
      LeadByte:array[0..12 - 1] of Byte;'+'UnicodeDefaultChar:WideChar;CodePage:UINT;CodePageName:
      array[0..MAX_PATH] of AnsiChar; end');
22788:   CL.AddTypeS('_cpinfoexW',record MaxCharSize :UINT;DefaultChar: array[0..2 - 1] of Byte; LeadByte
      :array[0..12 - 1] of Byte;'+'UnicodeDefaultChar: WideChar;CodePage:UINT;CodePageName:array[0..MAX_PATH] of
      WideChar;end');
22789:    (*_cpinfoexA = record
22790:         MaxCharSize         : UINT;     { max length (bytes) of a char }
22791:         DefaultChar         : array[0..MAX_DEFAULTCHAR - 1] of Byte;    { default character }
22792:         LeadByte            : array[0..MAX_LEADBYTES - 1] of Byte;      { lead byte ranges }
22793:         UnicodeDefaultChar  : WideChar;
22794:         CodePage            : UINT;
22795:         CodePageName        : array[0..MAX_PATH] of AnsiChar;
22796:     end;   *)
22797: //CL.AddTypeS('PCharsetInfo', '^TCharsetInfo // will not work');
22798:   CL.AddTypeS(TCharsetInfo', 'record MimeCharset : TMimeCharset; CodePage : Lo'
22799:    +ngWord; MimeName : CsuString; FriendlyName :Str; end');
22800:   CL.AddTypeS('TCharsetInfos', 'array of TCharsetInfo');
22801:   //CL.AddTypeS('LPCPINFOEXA', '^CPINFOEXA // will not work');
22802:   CL.AddTypeS('CPINFOEXA', '_cpinfoexA');
22803:   CL.AddTypeS('TCpInfoExA', 'CPINFOEXA');
22804:   //CL.AddTypeS('PCpInfoExA', 'LPCPINFOEXA');
22805:   //CL.AddTypeS('LPCPINFOEXW', '^CPINFOEXW // will not work');
22806:   CL.AddTypeS('CPINFOEXW', '_cpinfoexW');
22807:   CL.AddTypeS('TCpInfoExW', 'CPINFOEXW');
22808:   //CL.AddTypeS('PCpInfoExW', 'LPCPINFOEXW');
22809:   //CL.AddTypeS('PCpInfoEx', 'PCpInfoExW');
22810:   CL.AddTypeS('CPINFOEX', 'CPINFOEXW');
22811:   CL.AddTypeS('TCpInfoEx', 'TCpInfoExW');
22812:   //CL.AddTypeS('LPCPINFOEX', 'LPCPINFOEXW');
22813:   //CL.AddTypeS('PCpInfoEx', 'PCpInfoExA');
22814:   CL.AddTypeS('CPINFOEX', 'CPINFOEXA');
22815:   CL.AddTypeS('TCpInfoEx', 'TCpInfoExA');
22816:   //CL.AddTypeS('LPCPINFOEX', 'LPCPINFOEXA');
22817:   SIRegister_TCodePageObj(CL);
22818:  CL.AddDelphiFunction('Func CodePageToMimeCharset( ACodePage : LongWord) : TMimeCharset');
22819:  Func CodePageToMimeCharsetString( ACodePage : LongWord) : CsuString');
22820:  Func GetMimeInfo0( AMimeCharSet : TMimeCharset) : PCharSetInfo;
22821:  //&&CL.AddDelphiFunc('Func GetMimeInfo1(const AMimeCharSetStr:CsuString):PCharSetInfo;
22822:  //CL.AddDelphiFunction('Func GetMimeInfo2( ACodePage : LongWord) : PCharSetInfo;
22823:  Func MimeCharsetToCharsetString( AMimeCharSet : TMimeCharset) : CsuString');
22824:  ('Func ExtractMimeName( PInfo : PCharSetInfo) : CsuString');
22825:  Func MimeCharsetToCodePage3( AMimeCharSet : TMimeCharset) : LongWord;
22826:  Func MimeCharsetToCodePage4(const AMimeCharSetStr:CsuString;out ACodePage:LongWord):Bool;
22827:  Func MimeCharsetToCodePageDef( const AMimeCharSetString : CsuString) : LongWord');
22828:  Func MimeCharsetToCodePageEx5(const AMimeCharSetStr:CsuString;out ACodePage:LongWord):Bool;
22829:  Func MimeCharsetToCodePageExDef( const AMimeCharSetString : CsuString) : LongWord');
22830:  Func IsValidAnsiCodePage( ACodePage : LongWord) :Bool');
22831:  Func IcsIsValidCodePageID( ACodePage : LongWord) :Bool');
22832:  Func IsSingleByteCodePage( ACodePage : LongWord) :Bool');
22833:  Proc GetSystemCodePageList( AOwnsObjectList : TObjectList);
22834:  Func AnsiCodePageFromLocale( ALcid : LCID) : LongWord');
22835:  Func OemCodePageFromLocale( ALcid : LCID) : LongWord');
22836:  Func GetThreadAnsiCodePage : LongWord');
22837:  Func GetThreadOemCodePage : LongWord');
22838:  Func GetUserDefaultAnsiCodePage : LongWord');
22839:  Func GetUserDefaultOemCodePage : LongWord');
22840:  Func GetCPInfoExA( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEXA) : BOOL');
22841:  Func GetCPInfoExW( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEXW) : BOOL');
22842:  Func GetCPInfoEx( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEX) : BOOL');
22843:  Proc GetFriendlyCharsetList(Items:TStrings;IncludeList: TMimeCharsets; ClearItems : Bool);
```

```
22844:  Proc GetMimeCharsetList( Items : TStrings; IncludeList: TMimeCharsets; ClearItems: Bool);
22845: end;
22846:
22847: Proc SIRegister_OverbyteIcsMimeUtils(CL: TPSPascalCompiler);
22848: begin
22849:  TMimeUtilsVersion','LongInt').SetInt( 802);
22850:  'mimuutilsCopyRight','String').SetString( ' MimeUtils (c) 2003-2014 F. Piette V8.02 ');
22851:  'SmtpDefaultLineLength','LongInt').SetInt( 76);
22852:  'SMTP_SND_BUF_SIZE','LongInt').SetInt( 2048);
22853:  'icsRegContentType','String').SetString( 'MIME\Database\Content Type');
22854:  //CL.AddConstantN('SpecialsRFC822','TSysCharSet').SetString(ord(AnsiChar)( '(' ) or ord(AnsiChar)( ')' )
        or ord(AnsiChar) ( '<' ) or ord(AnsiChar) ( '>' ) or ord(AnsiChar) ( '@' ) or ord(AnsiChar)( ',' ) or
        ord(AnsiChar)( ';' ) or ord(AnsiChar) ( ':' )or ord(AnsiChar)( '\' )or ord(AnsiChar)( '"' )or
        ord(AnsiChar)( '[' ) or ord(AnsiChar)( ']' ) or ord(AnsiChar)( '.' ));
22855:  //CL.AddConstantN('CrLfSet','TSysCharSet').SetString(ord(AnsiChar)(#13)or ord(AnsiChar)(#10));
22856:  //CL.AddConstantN('QuotedCharSet','TSysCharSet').SetString(ord(AnsiChar)('?' )or ord(AnsiChar)('=' )or
        ord(AnsiChar)' ' )or ord(AnsiChar)( '_' ));
22857:  //CL.AddConstantN('BreakCharsSet','TSysCharSet').SetString(ord(AnsiChar)( #9 ) or ord(AnsiChar) (#32 )or
        ord(AnsiChar)(';')or ord(AnsiChar)(',')or ord(AnsiChar)('>')or ord(AnsiChar)(']'));
22858:  Func EncodeQuotedPrintable0( const S : RawByteString) :Str;
22859:  Func EncodeQuotedPrintable1(const S:UnicodeString; ACodePage: LongWord) :UnicodeString;
22860:  Func EncodeQuotedPrintable2( const S : UnicodeString) : UnicodeString;
22861:  Func DecodeQuotedPrintable3( const S : RawByteString) : RawByteString;
22862:  Func DecodeQuotedPrintable4(const S : UnicodeString; ACodePage:LongWord):UnicodeString;
22863:  Func DecodeQuotedPrintable5( const S : UnicodeString) : UnicodeString;
22864:  Func SplitQuotedPrintableString( const S :Str) :Str');
22865:  Func FilenameToContentType( FileName :Str) :Str');
22866:  Func Base64Encode6( const Input : Ansistr) : Ansistr;
22867:  Func Base64Encode7( const Input : PAnsiChar; Len : Integer) : Ansistr;
22868:  Func Base64Encode8( const Input : UnicodeString; ACodePage : LongWord):UnicodeString;
22869:  Func Base64Encode9( const Input : UnicodeString) : UnicodeString;
22870:  Func Base64Encode10( Input :StrBuilder) :StrBuilder;
22871:  Func Base64Decode11( const Input: Ansistr) : Ansistr;
22872:  Func Base64Decode12( const Input: UnicodeString; ACodePage : LongWord):UnicodeString;
22873:  Func Base64Decode13( const Input: UnicodeString) : UnicodeString;
22874:  ('Func Base64Decode14( Input :StrBuilder) :StrBuilder;
22875:  Func InitFileEncBase64( const FileName :Str; ShareMode : Word) : TStream');
22876:  Func DoFileEncBase64( var Stream : TStream; var More :Bool) : Ansistr');
22877:  Func DoFileEncQuotedPrintable( var Stream : TStream; var More :Bool) : Ansistr');
22878:  Func DoTextFileReadNoEncoding( var Stream : TStream; var More :Bool) : Ansistr');
22879:  Func DoFileLoadNoEncoding( var Stream : TStream; var More :Bool) :Str');
22880:  Func Base64EncodeEx15(const Input:RawByteString;MaxCol:Int;var
        cPos:Integer;CodePage:LongWord;IsMultiByteCP:Bool):RawByteString;
22881:  Func Base64EncodeEx16(const Input:UnicodeString;MaxCol:Integer;var cPos:Int):UnicodeString;
22882:  Func Base64EncodeEx17(const Input:UnicodeString;MaxCol:Integer;var cPos:Integer;ACodePage:LongWord):
        UnicodeString;
22883:  Proc EndFileEncBase64( var Stream : TStream);
22884:  Proc DotEscape( var S :Str; OnlyAfterCrLf :Bool);
22885:  //CL.AddDelphiFunction('Func IcsWrapTextEx18(const Line:RawByteString;const BreakStr:RawByteString;const
        BreakingChars: TSysCharSet; MaxCol:Integer;QuoteChars TSysCharSet;var
        cPos:Integer;ForceBreak:oolean;ACodePage:LongWord;IsMultiByteCP:Boolean) RawByteString;
22886:  Func IcsWrapTextEx19(const Line:Str; const BreakStr :Str; const BreakingChars :
        TSysCharSet;MaxCol:Integer;QuoteChars:TSysCharSet;var cPos:Integer;ForceBreak Boolean):str;
22887:  Func UnFoldHdrLine( const S :Str) :Str');
22888:  Func NeedsEncoding20( const S : Ansistr) :Bool;
22889:  Func NeedsEncoding21( const S : UnicodeString) :Bool;
22890:  Func NeedsEncodingPChar( S : PChar) :Bool');
22891:  Func HdrEncodeInLine22(const Input: RawByteString; Specials : TSysCharSet; EncType : AnsiChar; const
        CharSet : Ansistr; MaxCol : Integer; DoFold :Bool; CodePage: LongWord; IsMultiByteCP :Bool) :
        RawByteString;
22892:  Func HdrEncodeInLine23( const Input : UnicodeString; Specials : TSysCharSet; EncType : WideChar; const
        CharSet : UnicodeString; MaxCol:Integer; DoFold:Bool; Codepage: LongWord; IsMultiByteCP :Bool) :
        UnicodeString;
22893:  Func HdrEncodeInLineEx( const Input : UnicodeString; Specials : TSysCharSet; EncType : WideChar;
        CodePage:LongWord;MaxCol:Integer;DoFold:Bool;IsMultiByteCP:Bool): RawByteString');
22894:  Func StrEncodeQP24( const Input:RawByteString; MaxCol:Integer;Specials:TSysCharSet; CodePage : LongWord;
        IsMultibyteCP:Bool) :Str;
22895:  Func StrEncodeQP25(const Input:UnicodeString;MaxCol:Integer;Specials:TSysCharSet;ACodePage:LongWord;
        IsMultibyteCP:Bool):UnicodeString;
22896:  Func StrEncodeQPEx26(const Buf:RawByteString; MaxCol:Integer; Specials:TSysCharSet; ShortSpace:Bool;var
        cPos:Int;DoFold:Bool;CodePage:LongWord;IsMultibyteCP:Bool):RawByteString;
22897:  Func StrEncodeQPEx27(const Buf:UnicodeString;MaxCol:Integer;Specials:TSysCharSet;ShortSpace:Boolean;var
        cPos: Integer;DoFold:Boolean):UnicodeString;
22898:  Proc FoldHdrLine28( HdrLines : TStrings; const HdrLine :Str);
22899:  Proc FoldHdrLine29(HdrLines:TStrings;const HdrLine:RawByteString;
        ACodePage:LongWord;IsMultiByteCP:Boolean);
22900:  Func FoldString30(const Input:RawByteString; BreakCharsSet:TSysCharSet; MaxCol:Integer;
        ACodePage:LongWord; IsMultiByteCP:Bool):RawByteString;
22901:  Func FoldString31(const Input:UnicodeString;BreakCharsSet:TSysCharSet;MaxCol:Int):UnicodeString;
22902:  Func CalcBase64AttachmentGrow( FileSize : Int64) : Int64');
22903:  //Func EncodeMbcsInline32(CodePage:LongWord;const
        Charset:str;EncType:Char;Body:PWideChar;Len:Integer;DoFold:Boolean;MaxLen  Integer): Ansistr;
22904:  //Func EncodeMbcsInline33(CodePage:LongWord;const
        Charset:str;EncType:Char;Body:PAnsiChar;Len:Integer;DoFold:Boolean;MaxLen: Integer): Ansistr;
22905:  Func ContentTypeGetExtn( const Content :Str; var CLSID :Str) :Str');
22906:  'Func ContentTypeFromExtn( const Extension :Str) :Str');
22907:   CL.AddTypeS('TMimeTypeSrc','( MTypeList, MTypeOS, MTypeMimeFile, MTypeKeyFile, MTypeRes );
22908:  SIRegister_TMimeTypesList(CL);
22909: end;
```

```
22910:
22911:  Returns the MIME name of a Windows code page identifier or an empty     }
22912: { string if the code page identifier is not mapped.                      }
22913: Func  CodePageToMimeCharsetString(ACodePage: LongWord): CsuString;
22914:
22915: Proc SIRegister_TMimeTypesList(CL: TPSPascalCompiler);
22916: begin
22917:   //with RegClassS(CL,'TComponent', 'TMimeTypesList') do
22918:   with CL.AddClassN(CL.FindClass('TComponent'),'TMimeTypesList') do begin
22919:     Constructor Create( AOwner : TComponent);
22920:     Proc Free');
22921:     Func LoadTypeList :Bool');
22922:     Proc Clear');
22923:     Func CountExtn : integer');
22924:     Func CountContent : integer');
22925:     Func AddContentType(const AExtn, AContent:Str) :Bool');
22926:     Func LoadWinReg :Bool');
22927:     Func LoadFromOS :Bool');
22928:     Func LoadFromList :Bool');
22929:     Func LoadMimeFile( const AFileName :Str) :Bool');
22930:     Func LoadFromResource( const AResName :Str) :Bool');
22931:     Func LoadFromFile( const AFileName :Str) :Bool');
22932:     Func SaveToFile( const AFileName :Str) :Bool');
22933:     Proc LoadContentTypes( AList : TStrings);
22934:     Proc AddContentTypes( AList : TStrings);
22935:     Proc GetContentTypes( AList : TStrings);
22936:     Func TypeFromExtn( const AExtn :Str) :Str');
22937:     Func TypeFromFile( const AFileName :Str) :Str');
22938:     Func TypeGetExtn( const AContent :Str) :Str');
22939:     RegisterProperty('LoadOSonDemand', 'boolean', iptrw);
22940:     RegisterProperty('MimeTypesFile', 'string', iptrw);
22941:     RegisterProperty('DefaultTypes', 'TStringList', iptrw);
22942:     RegisterProperty('MimeTypeSrc', 'TMimeTypeSrc', iptrw);
22943:     RegisterProperty('UnknownType', 'string', iptrw);
22944:   end;
22945: end;
22946:
22947: Proc SIRegister_uWebSocket(CL: TPSPascalCompiler);
22948: begin
22949:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWebSocketConnection');
22950:   CL.AddTypeS(TWebSocketMessageEvent','Proc (AConnection:TWebSocketConnection;const AMessage:str);
22951:   TWebSocketConnectEvent', 'Proc ( AConnection : TWebSocketConnection);
22952:   TWebSocketDisconnectEvent', 'Proc ( AConnection : TWebSocketConnection);
22953:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWebSocketException');
22954:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWebSocketHandshakeException');
22955:   SIRegister_TWebSocketRequest(CL);
22956:   SIRegister_TWebSocketConnection(CL);
22957:   SIRegister_TWebSocketServer(CL);
22958: end;
22959:
22960: (*----------------------------------------------------------------------------*)
22961: Proc SIRegister_OverbyteIcsUrl(CL: TPSPascalCompiler);
22962: begin
22963:  CL.AddConstantN('IcsUrlVersion','LongInt').SetInt( 800);
22964:  CL.AddConstantN('uriCopyRight','String').SetString(' TIcsURL (c) 1997-2012 F. Piette V8.00 ');
22965:  Proc icsParseURL( const URL :Str; var Proto, User, Pass, Host, Port, Path:Str);
22966:  Func icsPosn( const s, t :Str; count : Integer) : Integer');
22967:  Func icsUrlEncode( const S:str;DstCodePage:LongWord) :Str');
22968:  Func icsUrlDecode0(const S:str;SrcCodePage:LongWord;DetectUtf8:Boolean):str;
22969:  Func icsUrlDecode1(const S:RawByteString;SrcCodePage:LongWord;DetectUtf8:Bool):UnicodeString;
22970:  Func icsUrlEncodeToA( const S:Str; DstCodePage : LongWord) : Ansistr');
22971: end;
22972:
22973: (*----------------------------------------------------------------------------*)
22974: Proc SIRegister_OverbyteIcsLogger(CL: TPSPascalCompiler);
22975: begin
22976:  CL.AddConstantN('TIcsLoggerVersion','LongInt').SetInt( 800);
22977:   IcsCopyRight','String').SetString( ' IcsLogger (c) 2005-2012 by François PIETTE V8.00 ');
22978:   CL.AddClassN(CL.FindClass('TOBJECT'),'ELoggerException');
22979:   CL.AddTypeS('TLogOption',(loDestEvent,loDestFile,loDestOutDebug,loAddStamp,loWsockErr,loWsockInfo,
      loWsockDump,loSslErr,loSslInfo,loSslDump,loProtSpecErr,loProtSpecInfo, loProtSpecDump);
22980:   TLogOptions', 'set of TLogOption');
22981:   TLogFileOption', '( lfoAppend, lfoOverwrite );
22982:   TLogFileEncoding', '( lfeUtf8, lfeUtf16 );
22983:   CL.AddConstantN(LogAllOptErr','LongInt).Value.ts32:=ord(loWsockErr) or ord(loSslErr) or
      ord(loProtSpecErr);
22984:   TNTEventType', '( etError, etWarning, etInformation, etAuditSuccess, etAuditFailure );
22985:   TIcsLogEvent', 'Proc ( Sender : TObject; LogOption : TLogOption; const Msg :Str);
22986:   SIRegister_TIcsLogger(CL);
22987: end;
22988:
22989: Proc MoveCardinal2String(const Source:Card; SrcStartIdx: Integer; var Dest:Str;
22990:   DstStartIdx: Integer; Count: Integer);
22991: Proc MoveString2Cardinal(const Source:Str; SrcStartIdx: Integer; var Dest:Card;
22992:   DstStartIdx: Integer; Count: Integer);
22993: Proc MoveCardinal(const Source:Card; var Dest:Card; Count: Integer);
22994: Func ExplodeStr(S:Str; Delimiter: Char; Index: Integer):Str; //overload;
22995:   var I: Integer; P, P1: PChar;
22996: Func ImplodeStr(S:Str; Delimiter: Char):Str;
```

```
22997: Func ExplodeStr(S:Str; Delimiter: Char; Index: Integer):Str; //overload;
22998: Func InSetWord(n:word):boolean;
22999:
23000:      borland video frame
23001: <iframe width="560" height="315" src="https://www.youtube.com/embed/m1Gn4echQIE" frameborder="0"
       allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
23002:   resourcestring
23003:     { Charsets, user-friendly names, known from OE and IE }
23004:     sArabicISO                  = 'Arabic (ISO)'; //28596
23005:     sArabicWindows              = 'Arabic (Windows)'; //1256
23006:     sBalticISO                  = 'Baltic (ISO)'; //28594
23007:     sBalticWindows              = 'Baltic (Windows)'; //1257
23008:     sCentralEuropeanISO         = 'Central European (ISO)'; //28592
23009:     sCentralEuropeanWindows     = 'Central European (Windows)'; //1250
23010:     sChineseTraditionalBig5     = 'Chinese Traditional (Big5)'; //950
23011:     sChineseSimplifiedGB18030   = 'Chinese Simplified (GB18030)'; //54936
23012:     sChineseSimplifiedGB2312    = 'Chinese Simplified (GB2312)';//936
23013:     sChineseSimplifiedHZ        = 'Chinese Simplified (HZ)'; //52936
23014:     sCyrillicISO                = 'Cyrillic (ISO)'; //28595
23015:     sCyrillicKOI8R              = 'Cyrillic (KOI8-R)'; //20866
23016:     sCyrillicKOI8U              = 'Cyrillic (KOI8-U)'; //21866
23017:     sCyrillicWindows            = 'Cyrillic (Windows)'; //1251
23018:     sEstonianISO                = 'Estonian (ISO)'; //28603
23019:     sGreekISO                   = 'Greek (ISO)'; //28597
23020:     sGreekWindows               = 'Greek (Windows)'; //1253
23021:     sHebrewISOLogical           = 'Hebrew (ISO-Logical)'; //38598
23022:     sHebrewISOVisual            = 'Hebrew (ISO-Visual)'; //28598
23023:     sHebrewWindows              = 'Hebrew (Windows)'; //1255
23024:     sJapaneseJIS                = 'Japanese (JIS)'; //932
23025:     sKorean                     = 'Korean'; //949
23026:     sKoreanEUC                  = 'Korean (EUC)'; //51949
23027:     sLatin9                     = 'Latin 9 (ISO)'; //28605
23028:     sThaiWindows                = 'Thai (Windows)'; //874
23029:     sTurkishISO                 = 'Turkish (ISO)'; //28599
23030:     sTurkishWindows             = 'Turkish (Windows)'; //1254
23031:     sUnicodeUTF7                = 'Unicode (UTF-7)'; //65001
23032:     sUnicodeUTF8                = 'Unicode (UTF-8)'; //65000
23033:     sVietnameseWindows          = 'Vietnamese (Windows)'; //1258
23034:     sWesternEuropeanISO         = 'Western European (ISO)'; //28591
23035:     sWesternEuropeanWindows     = 'Western European (Windows)'; //1252
23036:
23037: Proc SIRegister_psULib(CL: TPSPascalCompiler); // Prometheus
23038: begin
23039:  'clMoneyGreenps','LongWord').SetUInt( TColor ( $00C0DCC0 ));
23040:  ('clMedGrayps','LongWord').SetUInt( TColor ( $00A4A0A0 ));
23041:  ('clRxCreamps','LongWord').SetUInt( TColor ( $00A6CAF0 ));
23042:  ('clCreamps','LongWord').SetUInt( TColor ( $00F0FBFF ));
23043:  ('clRxSkyBlueps','LongWord').SetUInt( TColor ( $00FFFBF0 ));
23044:  ('clSkyBlueps','LongWord').SetUInt( TColor ( $00F0CAA6 ));
23045:  ('clInfoBkps','LongWord').SetUInt( TColor ( $02E1FFFF ));
23046:  ('clNoneps','LongWord').SetUInt( TColor ( $02FFFFFF ));
23047:  ('clSystemColor','LongWord').SetUInt( $FF000000);
23048:  ('MinutesPerDay','LongInt').SetInt( 24 * 60);
23049:  ('DefaultBeepDelay','LongInt').SetInt( 500);
23050:  ('cDoNotBringToTop','String').SetString( '-LaunchInBackground');
23051:   CL.AddTypeS('TFilenameEvent', 'Proc ( Sender : TObject; var Filename :Str);
23052:  //CL.AddTypeS('TWriteDebugProc','Procedure(const LogName,RoutineName,Commen:str;DebugData:array of
       string);
23053:   CL.AddTypeS('TMaxIntSize', '( mbsInt8, mbsWord8, mbsInt16, mbsWord16, mbsInt32, mbsWord32 );
23054:  Proc WriteDebugps(const LogName,RoutineName,Comment:str;DebugData:array of string);
23055:  Func Betweenps( Value, BoundA, BoundB : integer; AllowFlip :Bool) :Bool');
23056:  Func BooleanMatchsp( Value1 :Bool; Value2 : integer) :Bool;
23057:  Func BooleanMatch1ps( Value1 : integer; Value2 :Bool) :Bool;
23058:  Func BooleanMatch2ps( Value1, Value2 : integer) :Bool;
23059:  Func IDMatchps( RequiredID, TestID : integer) :Bool;
23060:  Func IDMatch1ps( RequiredID : TField; TestID : integer) :Bool;
23061:  Func IDMatch2ps( RequiredID, TestID : TField) :Bool;
23062:  Func Blendps( Color1, Color2 : TColor; Weight1 : integer; Weight2 : integer) : TColor');
23063:  Func MergeRGBps( Red, Green, Blue : integer) : TColor');
23064:  Proc SplitRGBps( Color : TColor; var Red, Green, Blue : integer);
23065:  Func VisibleContrastps( BackgroundColor : TColor) : TColor');
23066:  Func SortedStrps( Value :Str) :Str');
23067:  Func IntSortStrps( const Value : integer) :Str');
23068:  //Func AllAssignedps( Values : array of pointer) :Bool');
23069:  Func GetTokenps( var SourceStr :Str; Delim :Str) :Str');
23070:  Func URLizeps( SourceStr :Str) :Str');
23071:  Func UnURLizeps( const SourceStr :Str) :Str');
23072:  Func UnQuoteStrps( const Value :Str; QuoteChar : char) :Str');
23073:  Proc SetAndSaveBoolps( var OldVar :Bool; Value :Bool; var SaveVar :Bool);
23074:  Proc SetAndSaveIntps( var OldVar : integer; Value : integer; var SaveVar : integer);
23075:  Proc SetAndSaveStrps( var OldVar :Str; const Value :Str; var SaveVar :Str);
23076:  Func GetCharFromVKeyps( VKey : word) :Str');
23077:  Func IsControlKeyDownps :Bool');
23078:  Proc PushScreenCursorps( const aCursor : TCursor);
23079:  Proc PopScreenCursorps');
23080:  Func PeekScreenCursorps : TCursor');
23081:  Func CharIsNumericSymbolps( aChar : char) :Bool');
23082:  Func PrecisionMultiplierps( DecimalPrecision : byte) :Card');
23083:  Func TextToFloatValps(const aText:str;const DecimalPrecision:byte;const DefaultValue:extended):extended;
```

```
23084: Func TextToWordValps(const aText:str;const DefaultVal:cardi;const MaxIntSize:TMaxIntSize):cardi;
23085: Func TextToIntValps(const aText:str;const DefaultVal:int;const MaxIntSize:TMaxIntSize):int;
23086:  Func BankersRoundingps( const Num : currency; const DecimalPrecision : byte) : currency');
23087:  Func RoundDownps( const Num : Extended; const DecimalPrecision : byte) : Extended');
23088:  Func RoundUpps( const Num : Extended; const DecimalPrecision : byte) : Extended');
23089:  Func RoundNearestps( const Num : Extended; const DecimalPrecision : byte) : Extended');
23090:  Func NumDigitsps( const aNum : integer) : integer');
23091:  Func ZeroToMaxps( Value : integer) : integer');
23092:  Proc CopyIfPrefixMatchps(Source,Dest:TStrings;Prefix:str;ClearDest:bool;ValuesOnly:bool);
23093:  Func HexToCharps( const aValue : byte) : AnsiChar');
23094:  Func CharToHexps( const aValue : AnsiChar) : byte');
23095:  Func HexToTextps( const Source : Ansistr) : Ansistr');
23096:  Func TextToHexps( const Source : Ansistr) : Ansistr');
23097:  Proc SaveGraphicToStreamps( aGraphic : TGraphic; aStream : TStream);
23098:  Func LoadGraphicFromStreamps( aStream : TStream) : TGraphic');
23099:  Func GraphicToTextps( aGraphic : TGraphic) :Str');
23100:  Func TextToGraphicps( aGraphicAsText :Str) : TGraphic');
23101:  Func GetCustNameStrps( aFirstName, aLastName :Str) :Str');
23102:  Func GetLongCustNameStrps( aTitle, aFirstName, aLastName :Str) :Str');
23103:  Func GetPriceFromMarginps( const aMargin :Str; aCost : currency): currency');
23104:  Func GetMarginFromPriceps( aPrice, aCost : currency) :Str');
23105:  Func GetDriveSerialNumps( Path :Str) : longint');
23106:  Func GetCCCardTypeFromNameps( aCardName :Str) : integer');
23107:  Func GetItemMarginPercentps( Price, Cost : Currency) : double');
23108:  Func ValidateCreditCardps( aCardNum :Str) :Bool');
23109:  Func ValidateCreditCardRangeps( const aCardNum, aLowNum, aHighNum :Str) :Bool');
23110:  Func HHMMToTimeps( Value :Str) : TTime');
23111:  Func MMDDToDateps( Value :Str) : TDate');
23112:  Func TimeToHHMMSSps( Value : TTime) :Str');
23113:  Func DateToYYYYMMDDps( Value : TDate) :Str');
23114:  Func FontColorToStringps( SourceFont : TFont; SourceColor : TColor) :Str');
23115:  Proc StringToFontps( aStr :Str; DestFont : TFont);
23116:  Proc StringToFontColorps( aStr :Str; DestFont : TFont; var DestColor : TColor);
23117:  Proc ShowDebugStringsps( DebugStrings : TStrings);
23118:  Proc ShowStackDumpps');
23119:  Proc RegisterFileExtensionps(const Extension,Description,DefaultActionName,DefaultActionDescription,
       DefaultActionCommand:str; IconIndex:integer);
23120:  Func LaunchCommandps(aCommand:str; aParams:str; WaitForTerminate:boolean):bool;
23121:  Func GetWindowsDirectoryStrps :Str');
23122:  Func IsRemoteSessionps :Bool');
23123:  Func IsWinXPps :Bool');
23124:  Proc AssignTextFileToStreamps( var ATextFile : TextFile; AStream : TStream);
23125:  Func DeleteFileMaskps( const Mask :Str) :Bool');
23126:  Func FileMaskCountps( const Mask :Str) : integer');
23127:  Func FileMaskExistsps( const Mask :Str) :Bool');
23128:  Func FileMaskListps( const Mask :Str) :Str;
23129:  Func FileMaskList1ps(const Mask:str; Dest : TStrings; ClearList :Bool) :Bool;
23130:  Func GetSafeNumericVariantps( Value : Variant) : Variant');
23131:  Func GetPasswordCharStrps( aStr :Str) :Str');
23132:  Func GetDriversLicenseHeightDisplayTextps( aStr :Str) :Str');
23133:  Func IncludePathDelimiterps( const aFilePath :Str) :Str');
23134:  Func ExcludePathDelimiterps( const aFilePath :Str) :Str');
23135:  //Func GetEnvironmentVariable( const Name :Str) :Str');
23136:  Func ArrayToStrps( SourceArray : array of string; Separator : AnsiChar) :Str');
23137:  Func psStrReplaceps( var S : Ansistr; const Search, Replace : Ansistr; Flags : TReplaceFlags):boolean;
23138:  Func StrReplaceInStringsps(StrObject:TStrings;const Search,Replace:Ansistr;Flags:TReplaceFlags):bool;
23139:  Proc ShowMessageps( aMsg :Str);
23140:  Proc ShowErrorMessageps( aMsg :Str);
23141:  Func StandardUserQueryps(aMsg:str;DialogType:Card;DialogCaption:Str):Card;
23142:  Func ParamValueps( ParamName :Str) :Str');
23143:  Func ParamFlagSetps( ParamName :Str) :Bool');
23144:  Func ConcatenateNonBlanksps( Strings : array of string) :Str');
23145:  Func ConcatenateWithDelimiterps( Strings : array of string; Delimiter : PChar) :Str');
23146:  Func GetMaxStringLengthps( Strings : array of string) : integer');
23147:  Proc PlayBeepps( BeepActionType : TMsgDlgType; NumBeeps : integer; MSDelay : integer);
23148:  Func DelTreeps( const DirectoryName :Str) :Bool');
23149:  Proc UpdateActionCaptionps( anAction : TAction; aCaption :Str);
23150:  Proc GetDirectoryListps( const DirectoryName :Str; StringList : TStrings);
23151:  Func DecodePathps( const Path :Str; PathVars : TStrings) :Str');
23152:  Func KillEXEps( ExeName :Str) :Bool');
23153:  Func GetWindowsDefaultPrinterNameps :Str');
23154:  Func GetBackupFilenameps( const OrigFilename :Str) :Str');
23155:  Func GetRestoreFilenameps( const BackupFilename :Str) :Str');
23156:  Func MakeBackupFileps( const OrigFilename :Str; OverwriteExisting :Bool):Str');
23157:  Func RestoreBackupFileps( const BackupFilename:str;OverwriteExisting:boolean):Str');
23158:  Func CalcDistanceps( Lat1, Long1, Lat2, Long2 : double) : double');
23159:  Func GetDelimiterCountps( aString, aDelimiter :Str) : integer');   //}
23160: end;
23161:
23162: (*----------------------------------------------------------------------------*)
23163: Proc SIRegister_psUFinancial(CL: TPSPascalCompiler);
23164: begin
23165:  CL.AddTypeS(TLoanType', '( ltAmortized, ltSimple );
23166:  CL.AddTypeS(TLoanPayType','( lptMonthly, lptBiMonthly, lptBiWeekly );
23167: TypeS(TLoanPaymentEvent','Procedure(PayDate:TDate;PayNum:int;PaymentAmt,Balance,InterestAmt,
       TotalInterestAmt,AmortizedAmt,TotalAmortizedAmt:currency;var ExtraPayment: currency);
23168:  Func
       LoanPaymentAmt(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;LoanType
       :TLoanType):currency');
```

```
23169:  Func LoanNumPayments(Principal,
       PaymentAmt:currency;InterestRate:extended;PayType:TLoanPayType;LoanType:TLoanType):int;
23170:  Func LoanTotalEstimate(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;
       LoanType:TLoanType):currency;
23171:  Func
       LoanInterestEstimate(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;
       LoanType:TLoanType):currency;
23172:  Proc AmortizationSchedule( OnLoanPayment:TLoanPaymentEvent;StartDate:TDate;Principal
       :currency;InterestRate:extended;TotalNumPay:integer;PayType TLoanPayType;LoanType:TLoanType);
23173: Proc AmortizationSchedulePaymentAmt(OnLoanPayment TLoanPaymentEvent;StartDate:TDate; Principal,
       PaymentAmt:currency;InterestRate:extended;TotalNumPay:int;PayType:TLoanPayType; LoanType:TLoanType);
23174: end;
23175:
23176: (*----------------------------------------------------------------------------*)
23177: Proc SIRegister_TSimpleRSS(CL: TPSPascalCompiler);
23178: begin
23179:   //with RegClassS(CL,'TComponent', 'TSimpleRSS') do
23180: with CL.AddClassN(CL.FindClass('TComponent'),'TSimpleRSS') do begin
23181:   Constructor Create( AOwner : TComponent);
23182:   Proc Free');
23183:   Proc SaveToFile( Filename :Str);
23184:   Func SaveToString :Str');
23185:   Proc SaveToStream( Stream : TStream);
23186:   Func SaveToStrings : TStrings;
23187:   Proc LoadFromHTTP( URL :Str)
23188:   Proc LoadFromString( S :Str);
23189:   Proc LoadFromFile( Filename :Str);
23190:   Proc LoadFromStream( Stream : TStream);
23191:   Proc LoadFromStrings( Strings : TStrings);
23192:   Proc GenerateXML;
23193:   Proc GenerateXML1( FeedType : TXMLTypeRSS);
23194:   Proc GenerateComponent');
23195:   Proc ClearXML');
23196:   Property('Channel', 'TRSSChannel', iptrw);
23197:   Property('Items', 'TRSSItems', iptrw);
23198:   Property('Version', 'string', iptrw);
23199:   Property('XMLType', 'TXMLTypeRSS', iptrw);
23200:   Property('XMLFile', 'TXMLDocument', iptr);
23201:   Property('IndyHTTP', 'TIdHTTP', iptrw);
23202:   Property('SimpleRSSVersion', 'String', iptr);
23203:   Property('OnCreate', 'TNotifyEvent', iptrw);
23204:   Property('OnGenerateXML', 'TNotifyEvent', iptrw);
23205:   Property('OnParseXML', 'TNotifyEvent', iptrw);
23206:  end;
23207: end;
23208:
23209: (*----------------------------------------------------------------------------*)
23210: Proc SIRegister_SimpleRSSTypes(CL: TPSPascalCompiler);
23211: begin
23212: CL.AddClassN(CL.FindClass('TOBJECT'),'ESimpleRSSException');
23213: CL.AddTypeS('TLanguagesRSS', '( langAF, langSQ, langEU, langBE, langBG, langCA, '
23214:  +'langZH_CN, langZH_TW, langHR, langCS, langDA, langNL, langNL_BE, langNL_NL'
23215:  +',langEN, langEN_AU, langEN_BZ, langEN_CA, langEN_IE, langEN_JM, langEN_NZ'
23216:  +',langEN_PH, langEN_ZA, langEN_TT, langEN_GB, langEN_US, langEN_ZW, langET'
23217:  +',langFO, langFI, langFR, langFR_BE, langFR_CA, langFR_FR, langFR_LU, lang'
23218:  +'FR_MC, langFR_CH, langGL, langGD, langDE, langDE_AT, langDE_DE, langDE_LI,'
23219:  +' langDE_LU, langDE_CH, langEL, langHAW, langHU, langIS, langIN, langGA, la'
23220:  +'ngIT, langIT_IT, langIT_CH, langJA, langKO, langMK, langNO, langPL, langPT'
23221:  +', langPT_BR, langPT_PT, langRO, langRO_MO, langRO_RP, langRU, langRU_MO, l'
23222:  +'angRU_RU, langSR, langSK, langSL, langES, langES_AR, langES_BO, langES_CL,'
23223:  +' langES_CO, langES_CR, langES_DO, langES_EC, langES_SV, langES_GT, langES_'
23224:  +'HN, langES_MX, langES_NI, langES_PA, langES_PY, langES_PE, langES_PR, lang'
23225:  +'ES_ES, langES_UY, langES_VE, langSV, langSV_FI, langSV_SE, langTR, langUK,langX );
23226: CL.AddTypeS('TXMLTypeRSS', '( xtRDFrss, xtRSSrss, xtAtomrss, xtiTunesrss );
23227: TContentTypeRSS', '( ctTextrss, ctHTMLrss, ctXHTMLrss );
23228: TEncodingTypeRSS', '( etBase64rss, etEscapedrss, etXMLrss );
23229: SIRegister_TRFC822DateTime(CL);
23230: SIRegister_TRSSChannelReq(CL);
23231: SIRegister_TRSSImageReq(CL);
23232: SIRegister_TRSSImageOpt(CL);
23233: SIRegister_TRSSImage(CL);
23234: SIRegister_TRSSCloud(CL);
23235: SIRegister_TRSSTextInput(CL);
23236: SIRegister_TRSSChannelCategory(CL);
23237: SIRegister_TRSSChannelCategories(CL);
23238: SIRegister_TRSSChannelSkipHours(CL);
23239: SIRegister_TRSSChannelSkipDays(CL);
23240: SIRegister_TRSSChannelOpt(CL);
23241: SIRegister_TRSSChannel(CL);
23242: SIRegister_TRSSItemSource(CL);
23243: SIRegister_TRSSItemEnclosure(CL);
23244: SIRegister_TRSSItemCategory(CL);
23245: SIRegister_TRSSItemCategories(CL);
23246: SIRegister_TRSSItemGUID(CL);
23247: SIRegister_TRSSAuthor(CL);
23248: SIRegister_TRSSItem(CL);
23249: SIRegister_TRSSItems(CL);
23250: CL.AddTypeS(TFormatSettingsRSS', 'record ShortDateFormat :Str; LongDateForm'
23251:       +at :Str; LongTimeFormat :Str; ShortTimeFormat :Str; end');
```

```
23252: end;
23253:
23254: RSS Feed Snippet:
23255: const RSS_NewsFeed = 'http://feeds.bbci.co.uk/news/world/rss.xml';
23256:  with TSimpleRSS.create(self) do begin
23257:    XMLType:= xtRDFrss;
23258:    IndyHTTP:= TIdHTTP.create(self);
23259:    LoadFromHTTP(RSS_NewsFeed);
23260:    //LoadFromHTTP(Climatefeed);
23261:    writeln('RSSVersion: '+Version)
23262:    writeln('SimpleRSSVersion: '+SimpleRSSVersion)
23263:    for it:= 0 to items.count-1 do
23264:      writeln(itoa(it)+': '+Items[it].title+': '+items[it].pubdate.getdatetime);
23265:  end;
23266:
23267: (*----------------------------------------------------------------------------*)
23268: Proc SIRegister_JTools(CL: TPSPascalCompiler);
23269: begin
23270:  Func ParseCmdLine( ACmdLine : PChar; List : TStrings; QuoteChar : char) :Bool');
23271: Func GetCmdSwitchValue(const Switch:str;SwitchChars:TSysCharSet;var Value:str;IgnoreCase:boolean):bool;
23272:  //Proc ShowError( const S :Str; const Args : array of const);
23273: end;
23274:
23275: Proc SIRegister_rfc1213util(CL: TPSPascalCompiler);
23276: begin
23277:  CL.AddConstantN('rfc1213version','String').SetString( '0.1.2');
23278:  Func UptimeToDays( const Uptime :Str) :Str');
23279:  Func getSNMP( const mib, community, host :Str; var ErrNo : Integer) :Str');
23280:  Func sysServicesString( const sysServicesStr :Str) :Str');
23281:  Func ipForwardingString( const ipForwarding :Str) :Str');
23282:  Func ipRouteTypeString( const ipRouteType :Str) :Str');
23283:  Func ipRouteProtoString( const ipRouteProto :Str) :Str');
23284:  Func ipNetToMediaTypeString( const ipNetToMediaType :Str) :Str');
23285:  Func GetColFromTableRow( const EntryCols : TStringList; index : integer) :Str');
23286:  Func ifTypeString( ifTypeStringNo :Str) :Str;
23287:  Func ifStatusString( ifStatusStringNo :Str) :Str;
23288:  Func tcpRtoAlgorithmString( const tcpRtoAlgorithm :Str) :Str;
23289:  Func tcpConnStateString( const tcpConnState :Str) :Str;
23290:  Func egpNeighStateString( const egpNeighState :Str) :Str;
23291:  Func egpNeighModeString( const egpNeighMode :Str) :Str;
23292:  Func egpNeighEventTriggerString( const egpNeighEventTrigger :Str) :Str;
23293:  Func snmpEnableAuthenTrapsString( const snmpEnableAuthenTraps :Str) :Str;
23294:  Func FormatMac( const Mac :Str) :Str;
23295:  Proc JumpToRegistryKey(Key:Str);
23296: end;
23297:
23298: Proc SIRegister_neuralab(CL: TPSPascalCompiler);
23299: begin
23300:  SIRegister_TABHash(CL);
23301:  Func ABKey( S : array of byte; Divisor : longint) : longint');
23302:  Func ABCmp( var X, Y : array of byte) :Bool');
23303:  Func ABGetNext1( var AB : array of byte; ST : word) : word');
23304:  Func ABCountDif( var X, Y : array of byte) : longint');
23305:  Func ABCountDifZero( var X : array of byte) : longint');
23306:  Proc ABAnd( var A, B : array of byte);
23307:  Func ABGetEqual( var Equal, X, Y : array of byte) : longint');
23308:  Proc ABShiftLogicalLeft( var X : array of byte);
23309:  Proc ABShiftLogicalRight( var X : array of byte);
23310:  Func ABGetDif( var Dif, X, Y : array of byte) : longint');
23311:  Func ABToString( var AB : array of byte) :Str');
23312:  Func ABToStringR( var AB : array of byte) :Str');
23313:  Proc ABClear( var AB : array of byte);
23314:  Proc ABFull( var AB : array of byte);
23315:  Proc ABBitOnPos( var AB : array of byte; POS : longint);
23316:  Proc ABBitOnPosAtPos( var AB : array of byte; X, Start, Len : longint);
23317:  Func ABReadBitOnPosAtPos( var AB : array of single; Start, Len:longint): longint');
23318:  Proc ABCopy( var A, B : array of byte);
23319:  Proc ABTriPascal( var A, B : array of byte);
23320:  Proc ABSet( var A : array of byte; B : array of byte);
23321: end;
23322:
23323: Proc SIRegister_neuralbit(CL: TPSPascalCompiler);
23324: begin
23325:   //CL.AddTypeS('TArrOf2BytesPtr', '^TArrOf2Bytes // will not work');
23326:   //CL.AddTypeS('TArrOf3BytesPtr', '^TArrOf3Bytes // will not work');
23327:   //CL.AddTypeS('TArrOf4BytesPtr', '^TArrOf4Bytes // will not work');
23328:   //CL.AddTypeS('TArrBytePtr', '^TLongByteArray // will not work');
23329:  Func POT( numero, elevado : extended) : extended');
23330:  Func LongintBitTest( Data : longint; P : longint) :Bool');
23331:  Func LongintBitFlip( Data : longint; P : longint) : longint');
23332:  Proc BAClear( var VARS : array of byte);
23333:  Proc BAMake1( var VARS : array of byte);
23334:  Func BARead( var A : array of byte; P : longint) : byte');
23335:  Proc BAFlip( var A : array of byte; P : longint);
23336:  Proc BAWrite( var A : array of byte; P : longint; Data : byte);
23337:  Func BATest( var A : array of byte; P : longint) :Bool');
23338:  Proc BASum( var x, y : array of byte);
23339:  Proc BASub( var x, y : array of byte);
23340:  Proc BAIncPos( var x : array of byte; POS : longint);
```

```
23341:  Proc BADecPos( var x : array of byte; POS : longint);
23342:  Proc BAInc( var x : array of byte);
23343:  Proc BADec( var x : array of byte);
23344:  Func BAToString( VARS : array of byte) :Str');
23345:  Func BAToFloat( var VARS : array of byte) : extended');
23346:  Proc PFloatToBA( var VARS : array of byte; Valor : extended);
23347:  Proc BANot( var VARS : array of byte);
23348:  Proc BAAnd( var r, x, y : array of byte);
23349:  Proc BAOr( var r, x, y : array of byte);
23350:  Proc BAXOr( var r, x, y : array of byte);
23351:  Func BAGrater( var x, y : array of byte) :Bool;
23352:  Func BALower( var x, y : array of byte) :Bool;
23353:  Func BAEqual( var x, y : array of byte) :Bool;
23354:  Proc BAPMul( var r, x, y : array of byte);
23355:  Func nnRAnd( A, B : extended) : extended;
23356:  Func nnROr( A, B : extended) : extended;
23357:  Func nnRNot( A : extended) : extended;
23358:  Func nnRXor( A, B : extended) : extended;
23359:  Func REqual( A, B : extended) : extended;
23360:  Proc RSum( x, y, z : extended; var R, C : extended);
23361:  Proc RegSum( var x, y : array of extended);
23362:  Func RegEqual( var x, y : array of extended) : extended');
23363:  Func RegOrdEqual( var x, y : array of extended) : extended');
23364:  Func RegToString( var VARS : array of extended) :Str');
23365:  Func ROrer( var VARS : array of extended) : extended');
23366:  Func RAnder( var VARS : array of extended) : extended');
23367:  Func RCNot( X : extended; var VARS : array of extended) : extended');
23368:  Func ROrMaxTerm( var VARS : array of extended; NumMaxTerm : longint) : extended');
23369:  Func ROrMaxTermStr( NumVars : longint; NumMaxTerm : longint) :Str');
23370:  Func RSatFunc( var VARS : array of extended; NumFunc : longint) : extended');
23371:  Func RSatFuncStr( NumVars : longint; NumFunc : longint) :Str');
23372:  Proc RRegen( var VARS : array of extended);
23373:  Proc RDegen( var VARS : array of extended);
23374:  Proc RDegenP( var VARS : array of extended; P : extended);
23375:  Proc nnClear( var VARS : array of extended);
23376:  Proc BARAnd( var R, A, B : array of byte);
23377:  Proc BAROr( var R, AUX, A, B : array of byte);
23378:  Proc BARNot( var R, A : array of byte);
23379: end;
23380:
23381: (*----------------------------------------------------------------------------*)
23382: Proc SIRegister_winsvc2(CL: TPSPascalCompiler);
23383: begin
23384:  CL.AddConstantN('SERVICE_AUTO_START','LongWord').SetUInt( $00000002);
23385:  CL.AddConstantN('SERVICE_CONFIG_DELAYED_AUTO_START_INFO','LongInt').SetInt( 3);
23386:  ('SERVICE_CONFIG_FAILURE_ACTIONS_FLAG','LongInt').SetInt( 4);
23387:  ('SERVICE_CONFIG_PREFERRED_NODE','LongInt').SetInt( 9);
23388:  ('SERVICE_CONFIG_PRESHUTDOWN_INFO','LongInt').SetInt( 7);
23389:  ('SERVICE_CONFIG_REQUIRED_PRIVILEGES_INFO','LongInt').SetInt( 6);
23390:  ('SERVICE_CONFIG_SERVICE_SID_INFO','LongInt').SetInt( 5);
23391:  ('SERVICE_CONFIG_TRIGGER_INFO','LongInt').SetInt( 8);
23392:  ('SC_MANAGER_ALL_ACCESS','LongWord').SetUInt($F003F);
23393:  ('SERVICE_ALL_ACCESS','LongWord').SetUInt($F01FF);
23394:  Func ChangeServiceType( ServiceName :Str; TypeID : DWord) :Bool');
23395:  Func GetServiceStatus2( ServiceName :Str; ErrorState :Bool) :Bool');
23396:  Func StartService2( ServiceName :Str) :Bool');
23397:  Func StopService2( ServiceName :Str) :Bool');
23398: end;
23399:
23400: Proc SIRegister_neuralbyteprediction(CL: TPSPascalCompiler);
23401: begin
23402:   CL.AddTypeS('TNeuralCountings', 'array of longint');
23403:   SIRegister_TNeuronGroupBase(CL);
23404:   SIRegister_TNeuronGroup(CL);
23405:   CL.AddTypeS('TNeuralNetwork', 'array of TNeuronGroup');
23406:   SIRegister_TStatePredictionClass(CL);
23407:   SIRegister_TLabeledState(CL);
23408:   SIRegister_TClassifier(CL);
23409:   SIRegister_TEasyLearnAndPredictClass(CL);
23410: end;
23411:
23412: Proc SIRegister_neuralcache(CL: TPSPascalCompiler);
23413: begin
23414:  CL.AddConstantN('NeuralMaxStates','LongInt').SetInt( 400000);
23415:  CL.AddTypeS('TNeuralState', 'TBytes');
23416:  TProcPred', 'Proc ( var ST : TBytes; Acao : byte);
23417:  SIRegister_TCacheMem(CL);
23418: end;
23419:
23420: (*----------------------------------------------------------------------------*)
23421: Proc SIRegister_USolarSystem(CL: TPSPascalCompiler);
23422: begin
23423:  CL.AddTypeS('TOrbitalElements',record N:Double;i:Double;w:Double;a:Double;e:Double;M:Double; end;
23424:   TOrbitalElementsData', 'record NConst : Double; NVar : Double; i'
23425:    +Const : Double; iVar : Double; wConst : Double; wVar : Double; aConst : Do'
23426:    +uble; aVar : Double; eConst : Double; eVar : Double; MConst : Double; MVar:Double; end');
23427:  CL.AddConstantN('cAUToKilometers','Extended').setExtended( 149.6e6);
23428:  CL.AddConstantN('cEarthRadius','LongInt').SetInt( 6400);
23429:  Func GMTDateTimeToJulianDay( const dt : TDateTime) : Double');
```

```
23430:  Func ComputeOrbitalElements(const oeData:TOrbitalElementsData;const d:Double): TOrbitalElements;
23431:  Func ComputePlanetPosition( const orbitalElements : TOrbitalElements) : TAffineVector;
23432:  Func ComputePlanetPosition1(const orbitalElementsData:TOrbitalElementsData;const d:Double):TAffineVector;;
23433: end;
23434:
23435: Proc SIRegister_TSearchAnagrams(CL: TPSPascalCompiler);
23436: begin
23437:   //with RegClassS(CL,'tcontrol', 'TSearchAnagrams') do
23438:  with CL.AddClassN(CL.FindClass('tcontrol'),'TSearchAnagrams') do begin
23439:    RegisterProperty('useabbrevs', 'boolean', iptrw);
23440:    RegisterProperty('useforeign', 'boolean', iptrw);
23441:    RegisterProperty('usecaps', 'boolean', iptrw);
23442:    RegisterProperty('tag', 'integer', iptrw);
23443:    Proc Init(newletters:str;NewMinLen,NewMaxLen:word;newa,newf,newc:bool;pubDic:TDic);
23444:    Func FindMissingLetter( const s :Str) : TTestWords');
23445:    Proc Findallwords( const s :Str; list : Tstrings);
23446:  end;
23447: end;
23448:
23449: unit U_Invertedtext;
23450:   Proc InitInvertedText(canvas:TCanvas; pagewidth,pageheight:integer);
23451:   Proc DrawInvertedText(Canvas:TCanvas;pagewidth,pageheight:integer;linenbr:ints:str);
23452:   Proc MemoTextFixUp(memo:TMemo {var text:str});
23453:
23454: unit hashunit
23455: Proc SIRegister_THashStr(CL: TPSPascalCompiler);
23456: begin
23457:  //with RegClassS(CL,'TObject', 'THashStr') do
23458:  with CL.AddClassN(CL.FindClass('TObject'),'THashStr') do begin
23459:    y('test', 'array of THashObject', iptrw);
23460:    y('maxhash', 'integer', iptrw);
23461:    y('maxloadfactor', 'single', iptrw);
23462:    y('maxused', 'integer', iptrw);
23463:    y('used', 'integer', iptrw);
23464:    y('nbrcollisions', 'integer', iptrw);
23465:    y('maxcollisions', 'integer', iptrw);
23466:    Constructor create( newmaxhash : integer; newmaxloading : single);
23467:    Proc Free');
23468:    Func exists( s :Str; var T : TObject) :Bool');
23469:    Func AddIfNotDup( s :Str; t : Tobject) :Bool');
23470:    Proc resethash');
23471:    Proc rehash');
23472:    Func hash( s :Str) : integer');
23473:  end;
23474: end;
23475:
23476: Proc SortString(var SortSt : Ansistr);
23477: SortSt:= 'this is maXbox4';  SortString2(SortSt);  >>> 4Xabhiimosstx
23478: * === compile-time registration functions === *)
23479: (*----------------------------------------------------------------------------*)
23480: Proc SIRegister_JsonsUtilsEx(CL: TPSPascalCompiler);
23481: begin
23482:  Func FixedFloatToStr( const Value : Extended) :Str');
23483:  Func FixedTryStrToFloat( const S :Str; out Value : Extended) :Bool');
23484:  Func FixedStrToFloat( const S :Str) : Extended');
23485:  Func __ObjectToJson( aObject : TObject) :Str');
23486:  Proc __jsonToObject( const aJSONString :Str; var aObject : TObject);
23487:   CL.AddTypeS('TObjectDynArray', 'array of TObject');
23488:   TStringDynArray2', 'array of string');
23489:   TIntegerDynArray2', 'array of Integer');
23490:  CL.AddConstantN('GLB_JSON_STD_DECIMALSEPARATOR','String').SetString( '.');
23491: end;
23492:
23493: Proc SIRegister_TJson(CL: TPSPascalCompiler);
23494: begin
23495:  //with RegClassS(CL,'TJsonBase', 'TJson') do
23496:  with CL.AddClassN(CL.FindClass('TJsonBase'),'TJson') do begin
23497:    Constructor Create');
23498:    Proc Free');
23499:    Proc Parse( JsonString :Str);
23500:    Func Stringify :Str');
23501:    Proc Assign( Source : TJsonBase);
23502:    Proc Delete21( const Index : Integer);
23503:    Proc Delete22( const Name :Str);
23504:    Proc Clear');
23505:    Func Get23( const Index : Integer) : TJsonValue;
23506:    Func Get24( const Name :Str) : TJsonValue;
23507:    Func Put25( const Value : TJsonEmpty) : TJsonValue;
23508:    Func Put26( const Value : TJsonNull) : TJsonValue;
23509:    Func Put27( const Value :Bool) : TJsonValue;
23510:    Func Put28( const Value : Integer) : TJsonValue;
23511:    Func Put29( const Value : Extended) : TJsonValue;
23512:    Func Put30( const Value :Str) : TJsonValue;
23513:    Func Put31( const Value : TJsonArray2) : TJsonValue;
23514:    Func Put32( const Value : TJsonObject2) : TJsonValue;
23515:    Func Put33( const Value : TJsonValue) : TJsonValue;
23516:    Func Put34( const Value : TJson) : TJsonValue;
23517:    Func Putbool( const Value :Bool) : TJsonValue;  ----> alias
23518:    Func Putint( const Value : Integer) : TJsonValue;
```

```
23519:    Func Putext( const Value : Extended) : TJsonValue;
23520:    Func Putstr( const Value :Str) : TJsonValue;
23521:    Func Putarr( const Value : TJsonArray2) : TJsonValue;
23522:    Func Putobj( const Value : TJsonObject2) : TJsonValue;
23523:    Func Putval( const Value : TJsonValue) : TJsonValue;
23524:    Func Putjson( const Value : TJson) : TJsonValue;
23525:    Func Put35( const Name :Str; const Value : TJsonEmpty) : TJsonValue;
23526:    Func Put36( const Name :Str; const Value : TJsonNull) : TJsonValue;
23527:    Func Put37( const Name :Str; const Value :Bool) : TJsonValue;
23528:    Func Put38( const Name :Str; const Value : Integer) : TJsonValue;
23529:    Func Put39( const Name :Str; const Value : Extended) : TJsonValue;
23530:    Func Put40( const Name :Str; const Value :Str) : TJsonValue;
23531:    Func Put41( const Name :Str; const Value : TJsonArray2) : TJsonValue;
23532:    Func Put42( const Name :Str; const Value : TJsonObject2) : TJsonValue;
23533:    Func Put43( const Name :Str; const Value : TJsonValue) : TJsonValue;
23534:    Func Put44( const Name :Str; const Value : TJson) : TJsonValue;
23535:    Func Put45( const Value : TJsonPair) : TJsonValue;
23536:    RegisterProperty('StructType', 'TJsonStructType', iptr);
23537:    'JsonObject', 'TJsonObject2', iptr);
23538:    'JsonArray', 'TJsonArray2', iptr);
23539:    'Count', 'Integer', iptr);
23540:    'Values', 'TJsonValue String', iptr);
23541:    SetDefaultPropery('Values');
23542:  end;
23543: end;
23544:
23545: (*----------------------------------------------------------------------------*)
23546: Proc SIRegister_TJsonBase(CL: TPSPascalCompiler);
23547: begin
23548:   //with RegClassS(CL,'TObject', 'TJsonBase') do
23549:   with CL.AddClassN(CL.FindClass('TObject'),'TJsonBase') do  begin
23550:     Constructor Create( AOwner : TJsonBase);
23551:     Proc Free');
23552:     Proc Parse( JsonString :Str);
23553:     Func Stringify :Str');
23554:     Proc Assign( Source : TJsonBase);
23555:     Func Encode( const S :Str) :Str');
23556:     Func Decode( const S :Str) :Str');
23557:     Proc Split( const S :Str; const Delimiter: Char; Strings:TStrings);
23558:     Func IsJsonObject( const S :Str) :Bool');
23559:     Func IsJsonArray( const S :Str) :Bool');
23560:     Func IsJsonString( const S :Str) :Bool');
23561:     Func IsJsonNumber( const S :Str) :Bool');
23562:     Func IsJsonBoolean( const S :Str) :Bool');
23563:     Func IsJsonNull( const S :Str) :Bool');
23564:     Func AnalyzeJsonValueType( const S :Str) : TJsonValueType');
23565:     RegisterProperty('Owner', 'TJsonBase', iptr);
23566:   end;
23567: end;
23568:
23569: (*----------------------------------------------------------------------------*)
23570: Proc SIRegister_Jsons(CL: TPSPascalCompiler);
23571: begin
23572:   TJsonValueType','( jvNone, jvNull, jvString, jvNumber, jvBoolean, jvObject,jvArray );
23573:   TJsonStructType','( jsNone, jsArray, jsObject );
23574:   //CL.AddTypeS('TJsonNull', '( null );
23575:   //CL.AddTypeS('TJsonEmpty', '( empty );
23576:   SIRegister_TJsonBase(CL);
23577:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJsonObject2');
23578:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJsonArray2');
23579:   SIRegister_TJsonValue(CL);
23580:   SIRegister_TJsonArray2(CL);
23581:   SIRegister_TJsonPair(CL);
23582:   SIRegister_TJsonObject2(CL);
23583:   SIRegister_TJson(CL);
23584: end;
23585:    var jOb:= ajt.JsonObject;
23586:     for cnt:= 2 to job.count-2 do begin
23587:        Clabel:= job.items[cnt].name;
23588:        writeln('iterate '+clabel)
23589:        JsArr:= job.values[Clabel].asArray;
23590:        for cnt2:= 0 to jsarr.count-1 do
23591:          jsobj:= jsarr.items[cnt2].asobject;
23592:          for cnt3:= 0 to jsobj.count do
23593:            writeln(jsobj['elements'].asarray[0].asobject.items[cnt3].name)
23594:     end;
23595:
23596: println('elements status:
       '+ajt['rows'].asarray[0].asObject['elements'].asarray[0].asobject['status'].asstring);
23597:
23598: Proc SIRegister_Bricks(CL: TPSPascalCompiler);
23599: begin
23600:  CL.AddConstantN('kError101','String').SetString( 'Runtime error: Negative parameter(s);
23601:  kError102','String').SetString( 'Runtime error: Parameter(s) out of range');
23602:  kError103','String').SetString( 'Runtime error: min > max');
23603:  kError104','String').SetString( 'Runtime error: max = 0');
23604:  kError105','String').SetString( 'Runtime error: Denominator is zero');
23605:  CL.AddTypeS('TVectorE', 'array of extended');
23606:  complexreal', 'record re : real; im : real; end');
```

```
23607:  TFR', 'record M : extended; phi : extended; F : complexreal; end');
23608:  //CL.AddTypeS('complexreal', 'record re : real; im : real; end');
23609:  SIRegister_TBlock(CL);
23610:  SIRegister_TP2(CL);
23611:  SIRegister_TPT0(CL);
23612:  SIRegister_TPT1(CL);
23613:  SIRegister_TPT2(CL);
23614:  SIRegister_TInt2(CL);
23615:  SIRegister_TIT1(CL);
23616:  SIRegister_TDT1(CL);
23617:  SIRegister_TIT2(CL);
23618:  SIRegister_TPAdd(CL);
23619:  SIRegister_TPSub(CL);
23620:  SIRegister_TPMul(CL);
23621:  SIRegister_TPDiv(CL);
23622: end;
23623:
23624: Proc SIRegister_lifeblocks(CL: TPSPascalCompiler);
23625: begin
23626:   SIRegister_TASIA(CL);
23627:   SIRegister_TMiMe(CL);
23628:   SIRegister_TNoCoDI(CL);
23629: end;
23630:
23631: ****************************************************************
23632: Release Notes maXbox 4.7.4.64 June 2020 mX47
23633: ****************************************************************
23634: 1254 unit uPSI_MaskEdit.pas    FCL
23635: 1255 unit uPSI_SimpleRSSTypes; BlueHippo
23636: 1256 unit uPSI_SimpleRSS;      BlueHippo
23637: 1257 unit uPSI_psULib.pas      Prometheus
23638: 1258 unit uPSI_psUFinancial;   Prometheus
23639: 1259 uPSI_PsAPI_2.pas          mX4
23640: 1260 uPSI_PersistSettings_2    mX4
23641: 1261 uPSI_rfc1213util2.pas     IP
23642: 1262 uPSI_JTools.pas           JCL
23643: 1263 unit uPSI_neuralbit.pas   CAI
23644: 1264 unit uPSI_neuralab.pas    CAI
23645: 1265 unit uPSI_winsvc2.pas     TEK
23646: 1266 unit uPSI_wmiserv2.pas    TEK
23647: 1267 uPSI_neuralcache.pas      CAI
23648: 1268 uPSI_neuralbyteprediction CAI
23649: 1269 unit uPSI_USolarSystem;   glscene.org
23650: 1270 uPSI_USearchAnagrams.pas  DFF
23651: 1271 uPSI_JsonsUtilsEx.pas     Randolph
23652: 1272 unit uPSI_Jsons.pas       Randolph
23653: 1273 unit uPSI_HashUnit;       DFF
23654: 1274 uPSI_U_Invertedtext.pas   DFF
23655: 1275 unit uPSI_Bricks;         Dendron
23656: 1276 unit uPSI_lifeblocks.pas  Dendron
23657:
23658: Totals of Func Calls: 32633
23659: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
23660: CRC32: 3EB27A87:  28.2 MB (29,608,248) bytes
23661: Totals of Func Calls: 32614
23662:
23663: Functions_max hex in the box maXbox
23664: functionslist.txt
23665: FunctionsList1
       3.9.9./88/91/92/94/95/96/98/100/192/195/V402/V420/V422/V424.60.80/V426/V428/V458/V463/V471/V472/V474/V475
23666:
23667: Proc SIRegister_JclStringLists(CL: TPSPascalCompiler);
23668: begin
23669:  CL.AddClassN(CL.FindClass('TOBJECT'),'EJclStringListError');
23670:  CL.AddInterface(CL.FindInterface('IUNKNOWN'),IJclStringList, 'IJclStringList');
23671:  CL.AddTypeS('TJclStringListObjectsMode', '( omNone, omObjects, omVariants, omInterfaces );
23672:  SIRegister_IJclStringList(CL);
23673:  SIRegister_TJclInterfacedStringList(CL);
23674:  SIRegister_TJclStringList(CL);
23675:  Func JclStringList : IJclStringList;
23676:  Func JclStringListStrings( AStrings : TStrings) : IJclStringList;
23677:  Func JclStringListStrings1( const A : array of string) : IJclStringList;
23678:  Func JclStringList1( const A : array of const) : IJclStringList;
23679:  Func JclStringList2( const AText :Str) : IJclStringList;
23680: end;
23681:
23682: (*----------------------------------------------------------------------------*)
23683: Proc SIRegister_cInternetUtils(CL: TPSPascalCompiler);
23684: begin
23685:  ('ctHTML','String').SetString( 'text/html');
23686:  ('ctText','String').SetString( 'text/plain');
23687:  ('ctXML','String').SetString( 'text/xml');
23688:  ('ctJPG','String').SetString( 'image/jpeg');
23689:  ('ctGIF','String').SetString( 'image/gif');
23690:  ('ctBMP','String').SetString( 'image/bmp');
23691:  ('ctPNG','String').SetString( 'image/png');
23692:  ('ctTIFF','String').SetString( 'image/tiff');
23693:  ('ctMPG','String').SetString( 'video/mpeg');
23694:  ('ctAVI','String').SetString( 'video/avi');
```

```
23695:  ('ctQT','String').SetString( 'video/quicktime');
23696:  ('ctBinary','String').SetString( 'application/binary');
23697:  ('ctPDF','String').SetString( 'application/pdf');
23698:  ('ctPostscript','String').SetString( 'application/postscript');
23699:  ('ctBasicAudio','String').SetString( 'audio/basic');
23700:  ('ctMP3','String').SetString( 'audio/mpeg');
23701:  ('ctRA','String').SetString( 'audio/x-realaudio');
23702:  ('ctURLEncoded','String').SetString( 'application/x-www-form-urlencoded');
23703:  ('ctZIP','String').SetString( 'application/zip');
23704:  ('ctJavaScript','String').SetString( 'application/javascript');
23705:  ('ctPascal','String').SetString( 'text/x-source-pascal');
23706:  ('ctCPP','String').SetString( 'text/x-source-cpp');
23707:  ('ctINI','String').SetString( 'text/x-windows-ini');
23708:  ('ctBAT','String').SetString( 'text/x-windows-bat');
23709:  Func flcMIMEContentTypeFromExtention( const Extention :Str):Str');
23710:  ('protoHTTP','String').SetString( 'http');
23711:  ('protoNNTP','String').SetString( 'news');
23712:  ('protoFTP','String').SetString( 'ftp');
23713:  ('protoGopher','String').SetString( 'gopher');
23714:  ('protoEMail','String').SetString( 'mailto');
23715:  ('protoHTTPS','String').SetString( 'https');
23716:  ('protoIRC','String').SetString( 'irc');
23717:  ('protoFile','String').SetString( 'file');
23718:  ('protoTelnet','String').SetString( 'telnet');
23719:  Proc flcDecodeURL( const URL :Str; var Protocol, Host, Path :Str);
23720:  Func flcEncodeURL( const Protocol, Host, Path :Str) :Str');
23721:  Proc flcDecodeHost(const Address:str;var Host,Port:str;const DefaultPort:str);
23722:  Func flcEncodeDotLineTerminated( const S : Ansistr) : Ansistr');
23723:  Func flcEncodeEmptyLineTerminated( const S : Ansistr) : Ansistr');
23724:  Func flcDecodeDotLineTerminated( const S : Ansistr) : Ansistr');
23725:  Func flcDecodeEmptyLineTerminated( const S : Ansistr) : Ansistr');
23726:  //CL.AddDelphiFunction('Proc StreamDotLineTerminated0(const Source,Destination: AStream; const
        ProgressCallback:TCopyProgressProcedure);
23727:  //CL.AddDelphiFunction('Proc StreamDotLineTerminated1(const Source:str;const Destination:AStream;const
        ProgressCallback:TCopyProgressProcedure);
23728:  Func flchtmlCharRef( const CharVal : LongWord; const UseHex :Bool):Str');
23729:  Func flchtmlSafeAsciiText( const S :Str) :Str');
23730:  Proc flchtmlSafeWideText( var S : WideString);
23731:  Func flchtmlSafeQuotedText( const S :Str) :Str');
23732:  Func flcEncodeHeaderField( const Name, Body :Str) :Str');
23733:  Func flcDecodeHeaderField( const S :Str; var Name, Body :Str) :Bool');
23734:  Proc flcDecodeEMailAddress( const S :Str; var User, Domain :Str);
23735:  Proc flcDecodeEMailField( const S :Str; var EMailAddress, Name :Str);
23736:  Func flcDateFieldBody :Str');
23737:  Func flcDateField :Str');
23738:  Func flcMessageIDFieldBody( const ID :Str; const Host :Str) :Str');
23739:   SIRegister_AHeaderField(CL);
23740:   //CL.AddTypeS('AHeaderFieldClass', 'class of AHeaderField');
23741:   CL.AddTypeS('AHeaderFieldArray', 'array of AHeaderField');
23742:   SIRegister_THeaderCls(CL);
23743:   //CL.AddTypeS('THeaderClass', 'class of THeader');
23744:   CL.AddTypeS('THeaderArray', 'array of THeaderCls');
23745:   SIRegister_THeaderField(CL);
23746:   SIRegister_TInvalidField(CL);
23747:   SIRegister_TDateFieldClass(CL);
23748:   SIRegister_TEMailField(CL);
23749:  CL.AddDelphiFunction('Proc SelfTestInternetUtils');
23750:  maxform1.JumptoOutput1Click(self)
23751: end;
23752:
23753: Proc SIRegister_cWindows(CL: TPSPascalCompiler);
23754: Func InvMod(const A, N: Integer): Integer;
23755: Func ExpMod(A, Z: Integer; const N: Integer): Integer;
23756: Func CopyFrom(const S:Str; const Index: Integer):Str;
23757: Func CopyEx(const S:Str; const Start, Count: Integer):Str;
23758: Func loadForm2(vx, vy: smallint; acolor: TColor; aname:Str): TForm;
23759: PlaySound('INDIGO_WAV', hInstance, SND_RESOURCE or SND_ASYNC);
23760: Proc flcRaiseLastOSError');
23761:  Func flcStrEqualNoCase( const A,B:Str; const AsciiCaseSensitive :Bool):Bool');
23762:  Func flcStrMatchNoAsciiCase( const S, M :Str; const Index : Integer) :Bool');
23763:  Func flcStrMatchLeft( const S, M :Str; const AsciiCaseSensitive :Bool):Bool');
23764:  Func flcStrMatchRight( const S,M:Str; const AsciiCaseSensitive :Bool) :Bool');
23765:  Func flcPosChar( const F : Char; const S :Str; const Index : Integer) : Integer');
23766:  Func flcPosStr(const F,S:str;const Index:Integer;const AsciiCaseSensitive:Bool):nteger;
23767:  Func flcStrMatch( const S, M :Str; const Index : Integer) :Bool');
23768:  Func flcStrPMatch( const A, B : PChar; const Len : Integer) :Bool');
23769:  Func flcflcCopyFrom( const S :Str; const Index : Integer) :Str');
23770:  Func flcCopyRange( const S :Str; const StartIndex, StopIndex : Integer) :Str');
23771:  Func flcCopyLeft( const S :Str; const Count : Integer) :Str');
23772:  Func flcCopyRight( const S :Str; const Count : Integer) :Str');
23773:  Func flcStrIsNumeric( const S :Str) :Bool');
23774:  Func flcStrMatchChar( const S :Str; const M : TSYsCharSet) :Bool');
23775: Func flcStrSplitAtChar(const S:Str;const C:Char;var Left,Right:Str const Option1:Bool):Bool;
23776:  Func flcStrReplaceChar( const Find, Replace : Char; const S :Str) :Str');
23777: Func flcStrInclSuffix(const S:Str;const Suffix:Str;const AsciiCaseSensitive:Bool):str;
23778:  Func flcStrRemoveCharDelimited(var S:str;const FirstDelim,SecondDelim:Char) :Str');
23779:  Func flcStrSplit( const S, D :Str) : TStringArray');
23780:  Proc flcTrimInPlace( var S :Str; const C : TSYsCharSet);
23781:  Func flcTrimIn( S :Str; const C : TSYsCharSet) :Str');
```

```
23782:  Func flcStrAfterCharSet( const S :Str; const D : TSYsCharSet) :Str');
23783:  Func flcStrAfterChar( const S :Str; const D : Char) :Str');
23784:  Proc StrSplit(Delimiter: Char; Str:Str; ListOfStrings: TStrings) ;
23785:  (*----------------------------------------------------------------------------*)
23786: Proc SIRegister_cWindows(CL: TPSPascalCompiler);
23787: begin
23788:  CL.AddTypeS('TWindowsVersionflc', '( Win16_31, Win32_95, Win32_95R2, Win32_98, W'
23789:   +'in32_98SE, Win32_ME, Win32_Future, WinNT_31, WinNT_35, WinNT_351, WinNT_40'
23790:   +', WinNT5_2000, WinNT5_XP, WinNT5_2003, WinNT5_Future, WinNT_Future, Win_Future );
23791:  CL.AddTypeS('TWindowsVersions', 'set of TWindowsVersionflc');
23792:  Func flcGetWindowsVersion : TWindowsVersionflc');
23793:  Func flcIsWinPlatform95 :Bool');
23794:  Func flcIsWinPlatformNT :Bool');
23795:  Func flcGetWindowsProductID :Str');
23796:  Func flcGetWindowsTemporaryPath :Str');
23797:  Func flcGetWindowsPath :Str');
23798:  Func flcGetWindowsSystemPath :Str');
23799:  Func flcGetProgramFilesPath :Str');
23800:  Func flcGetCommonFilesPath :Str');
23801:  Func flcGetApplicationPath :Str');
23802:  Func flcGetUserName :Str');
23803:  Func flcGetLocalComputerName :Str');
23804:  Func flcGetLocalHostName :Str');
23805: CL.AddTypeS('TVersionInfoflc',(viFileVersion,viFileDescription,viLegalCopyrght,viComments,viCompanyName,
       viInternalNme,viLegalTrademarks,viOriginalFilename,viProductNme,viProductVersion;
23806:  Func flcGetAppVersionInfo( const VersionInfo : TVersionInfoflc) :Str');
23807:  Func flcWinExecute(const ExeName,Params:str;const ShowWin:Word;const Wait:Bool):Bool);
23808:  Func flcGetEnvironmentStrings : TStringArray');
23809:  Func flcContentTypeFromExtention( Extention :Str) :Str');
23810:  Func flcIsApplicationAutoRun( const Name :Str) :Bool');
23811:  Proc flcSetApplicationAutoRun( const Name :Str; const AutoRun :Bool);
23812:  Func flcGetWinPortNames : TStringArray');
23813:  Func flcGetKeyPressed( const VKeyCode : Integer) :Bool');
23814:  Func flcGetHardDiskSerialNumber( const DriveLetter : Char) :Str');
23815:  Func flcReboot :Bool');
23816: //CL.AddDelphiFunction('Func ConvertThreadToFiber( lpParameter : Pointer) : Pointer');
23817: //Func CreateFiber(dwStackSize:DWORD;lpStartAddress:TFNFiberStartRoutine;lpParameter:Ptr):Ptr');
23818:   CL.AddTypeS(TIEProxy', '( iepHTTP, iepHTTPS, iepFTP, iepGOPHER, iepSOCKS );
23819:  CL.AddDelphiFunction('Func GetIEProxy( const Protocol : TIEProxy) :Str');
23820:   CL.AddTypeS(TWindowHandleMessageEvent','Func(const Msg:Card;const wParam,lParam:Int;var
       Handled:Bool):Int;
23821:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWindowHandle');
23822:   TWindowHandleEvent', 'Proc ( const Sender : TWindowHandle);
23823:   (TWindowHandleErrorEvent','Procedure(const Sender:TWindowHandle;const E:Exception);
23824:   SIRegister_TWindowHandle(CL);
23825:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWindowHandle');
23826:   SIRegister_TfndWindowHandle(CL);
23827:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTimerHandle');
23828:   TTimerEventflc', 'Proc ( const Sender : TTimerHandle);
23829:   SIRegister_TTimerHandle(CL);
23830:   SIRegister_TfndTimerHandle(CL);
23831:  Proc flcRaiseLastOSError');
23832:  Func flcStrEqualNoCase(const A,B:Str; const AsciiCaseSensitive :Bool):Bool');
23833:  Func flcStrMatchNoAsciiCase( const S, M :Str; const Index : Integer) :Bool');
23834:  Func flcStrMatchLeft( const S, M :Str; const AsciiCaseSensitive :Bool):Bool');
23835:  Func flcStrMatchRight(const S,M :Str; const AsciiCaseSensitive :Bool):Bool');
23836:  Func flcPosChar( const F : Char; const S :Str; const Index : Integer): Integer');
23837:  Func flcPosStr(const F,S:str;const Index:Integer;const AsciiCaseSensitive:Bool):Integer;
23838:  Func flcStrMatch( const S, M :Str; const Index : Integer):Bool');
23839:  Func flcStrPMatch( const A, B : PChar; const Len : Integer):Bool');
23840:  Func flcflcCopyFrom( const S :Str; const Index : Integer):Str');
23841:  Func flcCopyRange( const S :Str; const StartIndex, StopIndex :Integer):str');
23842:  Func flcCopyLeft( const S :Str; const Count : Integer):Str');
23843:  Func flcCopyRight( const S :Str; const Count : Integer):Str');
23844:  Func flcStrIsNumeric( const S :Str) :Bool');
23845:  Func flcStrMatchChar( const S :Str; const M : TSYsCharSet) :Bool');
23846: Func flcStrSplitAtChar(const S Str;const C:Char;var Left,Right:Str;const Optional:Bool):Bool
23847:  Func flcStrReplaceChar( const Find, Replace : Char; const S :Str) :Str');
23848:  Func flcStrInclSuffix(const S:Str;const Suffix:Str;const AsciiCaseSensitive:Bool):str;
23849:  Func flcStrRemoveCharDelimited(var S:str;const FirstDelim,SecondDelim:Char) :Str');
23850:  Func flcStrSplit( const S, D :Str) : TStringArray');
23851:  Proc flcTrimInPlace( var S :Str; const C : TSYsCharSet);
23852:  Func flcTrimIn( S :Str; const C : TSYsCharSet) :Str');
23853:  Func flcStrAfterCharSet( const S :Str; const D : TSYsCharSet) :Str');
23854:  Proc StrSplit(Delimiter: Char; Str:Str; ListOfStrings: TStrings) ;
23855:  Func StrSplitF(Delimiter: Char; Str:Str): TStringlist;
23856:  Func Int32toBytes(const Cint: Integer): TBytes;
23857:  Proc FGPlayASound(const AResName:Str);
23858:  //CL.AddDelphiFunction('Proc FGPlayASound(const AResName:Str);
23859:  Proc FGPlayASound2(const AResName:Str; atype:Str);
23860:  Proc PlayResWav(const AResName:Str; atype:Str);
23861:  Proc PlayResWav2(const AResName:Str; atype:Str);  //sync tick1.wav
23862:  procedure PlaySoundStreamSync(const AName: TMemoryStream; atype: string);
23863:  procedure PlaySoundStreamAsync(const AName: TMemoryStream; atype: string);
23864:  procedure PlaySoundStreamAsyncNoStop(const AName: TMemoryStream; atype: string);
23865:  Func queryPerformanceCounter2(var mse: int64):Bool;
23866:  Func AddQuantumToDateTime(const dt: TDateTime): TDateTime;
23867:  Func AddQuantumToDateTime(const dt: TDateTime): TDateTime;
23868:  var overlay: Int64 absolute Result;
```

```
23869:    begin
23870:      Result:= dt;
23871:      overlay:= overlay+1;
23872:    end;
23873:  Func GetRTFtext(Const RichEdit: TRichEdit):Str;
23874:  Func StrRemoveCharSet(const S:Str; const C: CharSet):Str;
23875:  Func GetTextBetween(const Start, Stop:Str;var S, Between:Str):Boolean;
23876: end;
23877:
23878: Proc SIRegister_flcSysUtils(CL: TPSPascalCompiler);
23879: begin
23880:  SIRegister_TfreqObj(CL);
23881:  CL.AddTypeS('MSArray','array[0..1] of tmemorystream;
23882:  Func GetLastOSErrorCode : NativeInt');
23883:  Func GetLastOSErrorMessage :Str');
23884:  Proc TForm1msPlaySound( MS : MSArray; streaminuse : integer; aloop:boolean);
23885:  Proc BytesSetLengthAndZero( var V : TBytes; const NewLength : NativeInt);
23886:  Proc BytesInit( var V : TBytes; const R : Byte);
23887:  Proc BytesInit1( var V : TBytes; const S :Str);
23888:  Func BytesAppend( var V : TBytes; const R : Byte) : NativeInt');
23889:  Func BytesAppend1(var V : TBytes; const R : TBytes) : NativeInt');
23890:  Func BytesAppend2(var V : TBytes; const R : array of Byte) : NativeInt');
23891:  Func BytesAppend3(var V : TBytes; const R :Str) : NativeInt');
23892:  Func BytesCompare(const A, B : TBytes) : Integer');
23893:  Func BytesEqual( const A, B : TBytes) :Bool');
23894:  Func StringRefCount2(const S: Ansistr): Integer;
23895: Func  hton16(const A:Word): Word;
23896: Func  ntoh16(const A:Word): Word;
23897: Func  hton32(const A:LongWord): LongWord;
23898: Func  ntoh32(const A:LongWord): LongWord;
23899: Func  hton64(const A:Int64): Int64;
23900: Func  ntoh64(const A:Int64): Int64;
23901: CL.AddTypeS(TPoint3D2', 'record x : single; y : single; z : single; end;
23902: Func SphereTPoint3D(Phi, Lambda: Double): TPoint3D2;
23903: Func RotateAroundXTPoint3D(const P: TPoint3D2; Alfa: Double): TPoint3D2;
23904: Func RotateAroundYTPoint3D(const P: TPoint3D2; Beta: Double): TPoint3D2;
23905: Func LinkerTimeStamp(const FileName:Str): TDateTime; overload;
23906: Func ByteCharDigitToInt(const A: Char): Integer;
23907: Func WideCharDigitToInt(const A: WideChar): Integer;
23908: Func CharDigitToInt(const A: Char): Integer;
23909: Func IntToByteCharDigit(const A: Integer): Char;
23910: Func IntToWideCharDigit(const A: Integer): WideChar;
23911: Func IntToCharDigit(const A: Integer): Char;
23912: Func flcMinInt(const A, B: Int64): Int64;
23913: Func flcMaxInt(const A, B: Int64): Int64;
23914: end;
23915:
23916: Proc SIRegister_RotImg(CL: TPSPascalCompiler);
23917: begin
23918:  SIRegister_TRotateImage(CL);
23919:  Func CreateRotatedBitmap(Bitmap: TBitmap; const Angle:Extended; bgColor: TColor): TBitmap;
23920: end;
23921:
23922: *----------------------------------------------------------------------------*)
23923: Proc SIRegister_HSLUtils(CL: TPSPascalCompiler);
23924: begin
23925:  Func ARGB( a, r, g, b : Byte) :Card');
23926:  Func GetAValue2( argb : DWORD) : Byte');
23927:  Func GetRValue2( argb : DWORD) : Byte');
23928:  Func GetGValue2( argb : DWORD) : Byte');
23929:  Func GetBValue2( argb : DWORD) : Byte');
23930:  Func GetABGRAValue(abgr: DWORD) : Byte');
23931:  Func GetABGRRValue(abgr: DWORD) : Byte');
23932:  Func GetABGRGValue(abgr: DWORD) : Byte');
23933:  Func GetABGRBValue(abgr: DWORD) : Byte');
23934:  Func HSLtoRGB2( H, S, L : double) : TColor');
23935:  Func AHSLtoARGB( A, H, S, L : double) :Card');
23936:  Func HSLRangeToRGB( H, S, L : integer) : TColor');
23937:  Proc ARGBtoAHSL( ARGB : TColor; var A, H, S, L : double);
23938:  Proc ABGRtoAHSL( ABGR : TColor; var A, H, S, L : double);
23939:  Proc RGBtoHSLRange( RGB : TColor; var H, S, L : integer);
23940: end;
23941:
23942: Proc SIRegister_GraphicsMathLibrary(CL: TPSPascalCompiler);
23943: begin
23944:  gmlsizeUndefined','LongInt').SetInt( 1);
23945:  gmlsize2D','LongInt').SetInt( 3);
23946:  gmlsize3D','LongInt').SetInt( 4);
23947:  CL.AddClassN(CL.FindClass('TOBJECT'),'EVectorError');
23948:  CL.AddClassN(CL.FindClass('TOBJECT'),'EMatrixError');
23949:  gmlTAxis', '( axisX, axisY, axisZ );
23950:  gmlTCoordinate', '( coordCartesian, coordSpherical, coordCylindrical );
23951:  gmlTDimension', '( dimen2D, dimen3D );
23952:  gmlTIndex', 'Integer');
23953:  gmlTrotation', '( rotateClockwise, rotateCounterClockwise );
23954:  gmlTVector','record size: gmlTIndex; vector: ARRAY[1..4] OF DOUBLE; x,y,z,h:double; end;
23955:  gmlTMatrix', 'record size: gmlTIndex; matrix: ARRAY[1..4] of Array[1..4] OF DOUBLE; end;
23956:  Func gmlVector2D( const xValue, yValue : DOUBLE) : gmlTVector');
23957:  Func gmlVector3D( const xValue, yValue, zValue : DOUBLE) : gmlTVector');
```

```
23958:  Func gmlAddVectors(const u, v : gmlTVector) : gmlTVector');
23959:  Func gmlTransform( const u : gmlTVector; const a : gmlTMatrix) : gmlTVector');
23960:  Func gmlMatrix2D(const m11,m12 m13,m21,m22,m23,m31,m32,m33 : DOUBLE) : gmlTMatrix');
23961:  Func gmlMatrix3D(const m11,m12,m13,m14,m21,m22,m23,m24,m31,m32,m33,m34,m41,m42,m43,m44:DOUBLE):gmlTMatrix;
23962:  Func gmlMultiplyMatrices( const a, b : gmlTMatrix) : gmlTMatrix');
23963:  Func gmlInvertMatrix( const a, b : gmlTMatrix; var determinant : DOUBLE): gmlTMatrix');
23964:  Func gmlRotateMatrix(const dimension:gmlTDimension;const xyz:gmlTAxis;const angle:DOUBLE;const rotation
23965:                      :gmlTrotation):gmlTMatrix;
23966:  Func gmlScaleMatrix( const s : gmlTVector) : gmlTMatrix');
23967:  Func gmlTranslateMatrix( const t : gmlTVector) : gmlTMatrix');
23968:  Func gmlViewTransformMatrix(const coordinate:gmlTCoordinate;const azimuth,elevation,distance:DOUBLE;const
       ScreenX,ScreenY,ScreenDistance:DOUBLE):gmlTMatrix');
23969:  Func gmlFromCartesian(const ToCoordinate:gmlTCoordinate;const u gmlTVector): gmlTVector;
23970:  Func gmlToCartesian(const FromCoordinate: gmlTCoordinate;const u:gmlTVector): gmlTVector;
23971:  Func gmlToDegrees( const angle : DOUBLE) : DOUBLE');
23972:  Func gmlToRadians( const angle : DOUBLE) : DOUBLE');
23973:  Func gmlDefuzz( const x : DOUBLE) : DOUBLE');
23974:  Func gmlGetFuzz : DOUBLE');
23975:  Proc gmlSetFuzz( const x : DOUBLE);
23976: end;
23977:
23978: Proc SIRegister_umodels(CL: TPSPascalCompiler);
23979: begin
23980:  CL.AddTypeS('TRegType', '( REG_LIN, REG_MULT, REG_POL, REG_FRAC, REG_EXPO, RE'
23981:   +'G_IEXPO, REG_EXLIN, REG_LOGIS, REG_POWER, REG_GAMMA, REG_MICH, REG_MINT, R'
23982:   +'EG_HILL, REG_PK, REG_EVAL );
23983:   //CL.AddTypeS('TModel', 'record [..] end');
23984:  Func FirstParam( Model : TModel) : Integer');
23985:  Func LastParam( Model : TModel) : Integer');
23986:  Func FuncName( Model : TModel) :Str');
23987:  Func ParamName( Model : TModel; I : Integer) :Str');
23988:  Func RegFunc( Model : TModel; X : Float; B: TVector): Float');
23989:  Proc FitModel(Model:TModel;X,Y,Ycalc:TVector;U:TMatrix;Lb,Ub:Integer;MaxIter:Integer;Tol,
       SVDTol:Float;B:TVector;V:TMatrix;var Test:TRegTest);
23990:  Proc WFitModel(Model:TModel;X,Y,S:TVector;Ycalc:TVector;U:TMatrix;Lb,Ub:Integer;MaxIter:Integer;Tol,
       SVDTol:Float;B:TVector;V:TMatrix;var Test:TRegTest);
23991: end;
23992:
23993: (*----------------------------------------------------------------------------*)
23994: Proc SIRegister_TStatisticClass(CL: TPSPascalCompiler);
23995: begin
23996:   //with RegClassS(CL,'TOBJECT', 'TStatisticClass') do
23997:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TStatisticClass') do begin
23998:    Proc Assign( const S : TStatisticClass);
23999:    Func Duplicate : TStatisticClass');
24000:    Proc Clear');
24001:    Func IsEqual( const S : TStatisticClASS):Bool');
24002:    Proc Add( const V : MFloat);
24003:    Proc Add1( const V : array of MFloat);
24004:    Proc Add2( const V : TStatisticClass);
24005:    Proc AddNegated( const V : TStatisticClass);
24006:    Proc Negate');
24007:    Count', 'Integer', iptr);
24008:    Min', 'MFloat', iptr);
24009:    Max', 'MFloat', iptr);
24010:    Sum', 'MFloat', iptr);
24011:    SumOfSquares', 'MFloat', iptr);
24012:    SumOfCubes', 'MFloat', iptr);
24013:    SumOfQuads', 'MFloat', iptr);
24014:    Func Range : MFloat');
24015:    Func Mean : MFloat');
24016:    Func PopulationVariance : MFloat');
24017:    Func PopulationStdDev : MFloat');
24018:    Func Variance : MFloat');
24019:    Func StdDev : MFloat');
24020:    Func M1 : MFloat');
24021:    Func M2 : MFloat');
24022:    Func M3 : MFloat');
24023:    Func M4 : MFloat');
24024:    Func Skew : MFloat');
24025:    Func Kurtosis : MFloat');
24026:    Func GetAsString :Str');
24027:   end;
24028: end;
24029:
24030: (*----------------------------------------------------------------------------*)
24031: Proc SIRegister_flcStatistics(CL: TPSPascalCompiler);
24032: begin
24033:   CL.AddClassN(CL.FindClass('TOBJECT'),'EStatistics');
24034:   CL.FindClass('TOBJECT'),'EStatisticsInvalidArgument');
24035:   CL.FindClass('TOBJECT'),'EStatisticsOverflow');
24036:   //CL.AddTypeS('MFloat', 'double');
24037:   Proc SIRegister_flcStatistics(CL: TPSPascalCompiler);  begin
24038:   TOBJECT'),'EStatistics');
24039:   TOBJECT'),'EStatisticsInvalidArgument');
24040:   TOBJECT'),'EStatisticsOverflow');
24041:   //CL.AddTypeS('MFloat', 'double');        //check
24042:   CL.AddTypeS('MFloat', 'extended');        //checked
24043:  Func flcBinomialCoeff( N, R : Integer) : MFloat');
```

```
24044:  Func flcerf( x : MFloat) : MFloat');
24045:  Func flcerfc( const x : MFloat) : MFloat');
24046:  Func flcCummNormal( const u, s, X : MFloat) : MFloat');
24047:  Func flcCummNormal01( const X : MFloat) : MFloat');
24048:  Func flcInvCummNormal01( Y0 : MFloat) : MFloat');
24049:  Func flcInvCummNormal( const u, s, Y0 : MFloat) : MFloat');
24050:  Func flcCummChiSquare( const Chi, Df : MFloat) : MFloat');
24051:  Func flcCumF( const f, Df1, Df2 : MFloat) : MFloat');
24052:  Func flcCummPoisson( const X : Integer; const u : MFloat) : MFloat');
24053:   SIRegister_TStatisticClass(CL);
24054:   CL.AddClassN(CL.FindClass('TOBJECT'),'EStatistic');
24055:   CL.FindClass('TOBJECT'),'EStatisticNoSample');
24056:   CL.FindClass('TOBJECT'),'EStatisticDivisionByZero');
24057:  Proc TestStatisticClass');
24058: end;
24059:
24060: Proc Include(var aset: Charset; achar: Char);  ---> cFundamentUtils
24061: Proc Exclude(var aset: Charset; achar: Char);
24062: Proc SIRegister_StdFuncs(CL: TPSPascalCompiler);
24063:  begin
24064:   CL.AddClassN(CL.FindClass('TOBJECT'),'EParserError');
24065:   CL.AddTypeS('TCharSet', 'set of Char');
24066:  Proc SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
24067:  CL.AddTypeS('CharSet', 'set of Char');       //!!!
24068:  CL.AddTypeS('AnsiCharSet', 'TCharSet');
24069:  CL.AddTypeS('ByteSet', 'set of Byte');
24070:  CL.AddTypeS('AnsiChar', 'Char');
24071:
24072: Proc SIRegister_flcMaths(CL: TPSPascalCompiler);
24073: begin
24074:  flcPi','Extended).setExtended(3.1415926535897932384 6+0.2643383279502884197 1e-20+0.6939937510582097494 4e-
       40+0.5923078164062862089 9e-60+0.8628034825342117067 9e-80+
       100+0.0938446095505822317 2e-120+0.5359408128481117450 2e-140+
       160+0.6446229489549303819 6e-180);
24075:  'flcPi2','Extended').setExtended( 6.2831853071795864769252867665590 06);
24076:  'flcPi3','Extended').setExtended( 9.4247779607693797153879301498385 09);
24077:  'flcPi4','Extended').setExtended( 12.5663706143591729538505735331180 1);
24078:  'flcPiOn2','Extended').setExtended( 1.5707963267948966192313216916397 51);
24079:  'flcPiOn3','Extended').setExtended( 1.0471975511965977461542144610931 68);
24080:  'flcPiOn4','Extended').setExtended( 0.7853981633974483096156608458198 76);
24081:  'flcPiSq','Extended').setExtended( 9.8696044010893586188344909998761 51);
24082:  'flcPiE','Extended').setExtended( 22.4591577183610454734271522045437 4);
24083:  'flcLnPi','Extended').setExtended( 1.1447298858494001741434273513530 59);
24084:  'flcLogPi','Extended').setExtended( 0.4971498726941338543512682882908 99);
24085:  'flcSqrtPi','Extended').setExtended( 1.7724538509055160272981674833411 45);
24086:  'flcSqrt2Pi','Extended').setExtended( 2.5066282746310005024157652848110 45);
24087:  'flcLnSqrt2Pi','Extended').setExtended( 0.9189385332046727417803297364056 18);
24088:  'flcDegPerRad','Extended').setExtended( 57.2957795130823208767981548141051 7);
24089:  'flcDegPerGrad','Extended').setExtended( 0.9);
24090:  'flcDegPerCycle','Extended').setExtended( 360.0);
24091:  'flcGradPerCycle','Extended').setExtended( 400.0);
24092:  'flcGradPerDeg','Extended').setExtended( 1.1111111111111111111111111111111111);
24093:  'flcGradPerRad','Extended').setExtended( 63.6619772367581343075535053490 06);
24094:  'flcRadPerDeg','Extended').setExtended( 0.0174532925199432957692369076848 86);
24095:  'flcRadPerGrad','Extended').setExtended( 0.0157079632679489661923132169163 98);
24096:  'flcRadPerCycle','Extended').setExtended( 6.2831853071795864769252867665 59);
24097:  'flcCyclePerDeg','Extended').setExtended( 0.0027777777777777777777777777777778);
24098:  'flcCyclePerRad','Extended').setExtended( 0.1591549430918953357688837633725 1);
24099:  'flcCyclePerGrad','Extended').setExtended( 0.0025);
24100:  'flcE','Extended').setExtended( 2.7182818284590452353602874713526 63);
24101:  'flcE2','Extended').setExtended( 7.3890560989306502272304274605750 08);
24102:  'flcExpM2','Extended').setExtended( 0.1353352832366126918939994949724 84);
24103:  'flcLn2','Extended').setExtended( 0.6931471805599453094172321214581 77);
24104:  'flcLn10','Extended').setExtended( 2.3025850929940456840179914546843 64);
24105:  'flcLogE','Extended').setExtended( 0.4342944819032518276511289189166 05);
24106:  'flcLog2','Extended').setExtended( 0.3010299956639811952137388947244 93);
24107:  'flcLog3','Extended').setExtended( 0.4771212547196624372950279032551 15);
24108:  'flcSqrt2','Extended').setExtended( 1.4142135623730950488016887242096 98);
24109:  'flcSqrt3','Extended').setExtended( 1.7320508075688772935274463415058 72);
24110:  'flcSqrt5','Extended').setExtended( 2.2360679774997896964091736687312 76);
24111:  'flcSqrt7','Extended').setExtended( 2.6457513110645905905016157536392 60);
24112:  //CL.AddTypeS('MFloat', 'Double');
24113:  //CL.AddTypeS('MFloatArray', 'DoubleArray');
24114:  //CL.AddTypeS('MFloat', 'Extended');       / check t<pe
24115:  CL.AddTypeS('ExtendedArray', 'array of MFloat');
24116:  //ExtendedArray = array of Extended;
24117:  CL.AddTypeS('MFloatArray', 'ExtendedArray');
24118:  CL.AddTypeS('Int64Array', 'array of Int64');
24119:  flcfx', 'Func (const x: Float): Float;
24120:  MFloatArray2', 'array of Mfloat');
24121:  //Int64Array  = array of Int64;
24122:  // CL.AddTypeS('PMFloat', '^MFloat // will not work');
24123:  Proc SetFPUPrecisionSingle');
24124:  Proc SetFPUPrecisionDouble');
24125:  Proc SetFPUPrecisionExtended');
24126:  Proc SetFPURoundingNearest');
24127:  Proc SetFPURoundingDown');
24128:  Proc SetFPURoundingUp');
24129:  Proc SetFPURoundingTruncate');
```

```
24130:  Func flcPolyEval(const X:MFloat;const Coef:array of MFloat;const N:Integer): MFloat;
24131:  Func flcSign( const R : Integer) : Integer;
24132:  Func flcSign1( const R : Int64) : Integer;
24133:  Func flcSign2( const R : Single) : Integer;
24134:  Func flcSign3( const R : Double) : Integer;
24135:  Func flcSign4( const R : Extended) : Integer;
24136:  Func flcFloatMod( const A, B : MFloat) : MFloat');
24137:  Func flcATan360( const X, Y : MFloat) : MFloat');
24138:  Func flcInverseTangentDeg( const X, Y : MFloat) : MFloat');
24139:  Func flcInverseTangentRad( const X, Y : MFloat) : MFloat');
24140:  Func flcInverseSinDeg( const Y, R : MFloat) : MFloat');
24141:  Func flcInverseSinRad( const Y, R : MFloat) : MFloat');
24142:  Func flcInverseCosDeg( const X, R : MFloat) : MFloat');
24143:  Func flcInverseCosRad( const X, R : MFloat) : MFloat');
24144:  Func flcDMSToReal( const Degs, Mins, Secs : MFloat) : MFloat');
24145:  Proc flcRealToDMS( const X : MFloat; var Degs, Mins, Secs : MFloat);
24146:  Proc flcPolarToRectangular( const R, Theta : MFloat; var X, Y : MFloat);
24147:  Proc flcRectangularToPolar( const X, Y : MFloat; var R, Theta : MFloat);
24148:  Func flcDistance( const X1, Y1, X2, Y2 : MFloat) : MFloat');
24149:  Func flcIsPrime( const N : Int64) :Bool');
24150:  Func flcIsPrimeFactor( const N, F : Int64) :Bool');
24151:  Func flcPrimeFactors( const N : Int64) : Int64Array');
24152:  Func flcGCD( const N1, N2 : Integer) : Integer;
24153:  Func flcGCD1( const N1, N2 : Int64) : Int64;
24154:  Func flcIsRelativePrime( const X, Y : Int64) :Bool');
24155:  Func flcInvMod( const A, N : Integer) : Integer;
24156:  Func flcInvMod1( const A, N : Int64) : Int64;
24157:  Func flcExpMod( A, Z : Integer; const N : Integer) : Integer;
24158:  Func flcExpMod1( A, Z : Int64; const N : Int64) : Int64;
24159:  Func flcJacobi( const A, N : Integer) : Integer');
24160:  CL.AddConstantN('FactorialMaxN','LongInt').SetInt( 1754);
24161:  //CL.AddConstantN('FactorialMaxN','LongInt').SetInt( 170);
24162: Func flcFactorial( const N : Integer) : MFloat');
24163: Func flcCombinations( const N, C : Integer) : MFloat');
24164: Func flcPermutations( const N, P : Integer) : MFloat');
24165: Func flcFibonacci( const N : Integer) : Int64');
24166: Func flcGammaLn( X : Extended) : Extended');
24167: Proc flcFourierTransform(const AngleNumerator:MFloat;const RealIn,ImagIn:array of MFloat;var RealOut,
       ImagOut: MFloatArray);
24168:  Proc flcFFT( const RealIn,ImagIn:array of MFloat; var RealOut, ImagOut : MFloatArray);
24169:  Proc flcInverseFFT(const RealIn,ImagIn:array of MFloat;var RealOut,ImagOut:MFloatArray);
24170: Proc flcCalcFrequency(const FrequencyIndex:Int;const RealIn,ImagIn:array of MFloat;var RealOut,
       ImagOut:MFloat);
24171:  Func flcSecantSolver( const f : flcfx; const y, Guess1, Guess2 : MFloat) : MFloat');
24172:  Func flcNewtonSolver( const f, df : flcfx; const y, Guess : MFloat) : MFloat');
24173:  Func flcFirstDerivative( const f : flcfx; const x : MFloat) : MFloat');
24174:  Func flcSecondDerivative( const f : flcfx; const x : MFloat) : MFloat');
24175:  Func flcThirdDerivative( const f : flcfx; const x : MFloat) : MFloat');
24176:  Func flcFourthDerivative( const f : flcfx; const x : MFloat) : MFloat');
24177:  Func flcSimpsonIntegration( const f : flcfx; const a, b : MFloat; N : Integer): MFloat');
24178:  Proc TestMathClass');
24179: end;
24180:
24181: Proc SIRegister_flcCharSet(CL: TPSPascalCompiler);
24182: begin
24183: //type ByteSet = set of Byte;
24184: //ByteCharSet = Charset; /
24185:    CL.AddTypeS('ByteCharSet', 'Charset');
24186:    // CL.AddTypeS('ByteCharSet', 'charset');
24187:  Func flcAsAnsiCharSet( const C : array of Char) : ByteCharSet');
24188:  Func flcAsByteSet( const C : array of Byte) : ByteSet');
24189:  Func flcAsByteCharSet( const C : array of Char) : ByteCharSet');
24190:  Proc flcComplementChar( var C : ByteCharSet; const Ch : Char);
24191:  Proc flcClearCharSet( var C : ByteCharSet);
24192:  Proc flcFillCharSet( var C : ByteCharSet);
24193:  Proc flcComplementCharSet( var C : ByteCharSet);
24194:  Proc flcAssignCharSet( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24195:  Proc flcUnion( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24196:  Proc flcDifference( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24197:  Proc flcIntersection( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24198:  Proc flcXORCharSet( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24199:  Func flcIsSubSet( const A, B : ByteCharSet) :Bool');
24200:  Func flcIsEqual( const A, B : ByteCharSet) :Bool');
24201:  Func flcIsEmpty( const C : ByteCharSet) :Bool');
24202:  Func flcIsComplete( const C : ByteCharSet) :Bool');
24203:  Func flcCharCount( const C : ByteCharSet) : Integer');
24204:  Proc flcConvertCaseInsensitive( var C : ByteCharSet);
24205:  Func flcCaseInsensitiveCharSet( const C : ByteCharSet) : ByteCharSet');
24206:  Func flcCharSetToStrB( const C : ByteCharSet) : RawByteString');
24207:  Func flcStrToCharSetB( const S :Str) : ByteCharSet');
24208:  Func flcCharSetToCharClassStr( const C : ByteCharSet) : Ansistr');
24209:  Proc TestCharset');
24210: end;
24211:
24212: Proc SIRegister_flcTimers(CL: TPSPascalCompiler);
24213: begin
24214:  TOBJECT'),'ETimerError');
24215:  Func GetHighResolutionTick : Word64');
24216:  Func GetHighResolutionFrequency : Word64');
```

```
24217:  Func HighResolutionTickDelta( const T1, T2 : Word64) : Int64');
24218:  Func HighResolutionTickDeltaU( const T1, T2 : Word64) : Word64');
24219:  'MicroTickFrequency','LongInt').SetInt( 1000000);
24220:  Func GetMicroTick : Word64');
24221:  Func MicroTickDelta( const T1, T2 : Word64) : Int64');
24222:  Func MicroTickDeltaU( const T1, T2 : Word64) : Word64');
24223: 'MilliTickFrequency','LongInt').SetInt( 1000);
24224:  Func GetMilliTick : Word64');
24225:  Func MilliTickDelta( const T1, T2 : Word64) : Int64');
24226:  Func MilliTickDeltaU( const T1, T2 : Word64) : Word64');
24227:  Func DateTimeToMicroDateTime( const DT : TDateTime) : Word64');
24228:  Func MicroDateTimeToDateTime( const DT : Word64) : TDateTime');
24229:  Func GetMicroDateTimeNow : Word64');
24230:  Func GetNowUT : TDateTime');
24231:  Func GetMicroDateTimeNowUT : Word64');
24232:  Func GetNowUTC( const ReInit :Bool) : TDateTime');
24233:  Func GetMicroDateTimeNowUTC( const ReInit :Bool) : Word64');
24234:  Proc TestTimerClass;
24235: end;
24236:
24237: Proc SIRegister_TBlaiseLexer(CL: TPSPascalCompiler);
24238: begin
24239: //with RegClassS(CL,'TOBJECT', 'TBlaiseLexer') do
24240:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TBlaiseLexer') do begin
24241:   TokenCount', 'Integer', iptrw);
24242:   IdenCount', 'Integer', iptrw);
24243:   SymbolCount', 'Integer', iptrw);
24244:   IgnoreCount', 'Integer', iptrw);
24245:   ReservedCount', 'Integer', iptrw);
24246:   Constructor Create');
24247:   Proc Reset');
24248:   Proc SetData( const Data : pchar; const Size : Integer);
24249:   Proc SetText( const S :Str; const StartPosition : Integer);
24250:   Func EOF :Bool');
24251:   Proc GetToken');
24252:   TokenLen', 'Integer', iptr);
24253:   TokenType', 'Integer', iptr);
24254:   Func TokenText :Str');
24255:   TokenPos', 'Integer', iptr);
24256:   LastTokenEnd', 'Integer', iptr);
24257:   Position', 'Integer', iptrw);
24258:   LineNr', 'Integer', iptr);
24259:   Column', 'Integer', iptr);
24260:   CurrentLine', 'String', iptr);
24261:   Func MatchType( const TokenType : Integer; const Skip :Bool) :Bool');
24262:   Func MatchTokenText(const Token:str;const Skip:Bool;const CaseSensitive:Bool):Bool;
24263:   Func TokenAsInteger : Int64');
24264:  end;
24265: end;
24266:
24267: Proc SIRegister_TRationalClass(CL: TPSPascalCompiler);
24268: begin
24269: //with RegClassS(CL,'TOBJECT', 'TRationalClass') do
24270:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TRationalClass') do begin
24271:   Constructor Create;
24272:   Constructor Create1( const Numerator : Int64; const Denominator: Int64);
24273:   Constructor Create2( const R : Extended);
24274:   Numerator', 'Int64', iptrw);
24275:   Denominator', 'Int64', iptrw);
24276:   AsString', 'String', iptrw);
24277:   AsStringB', 'RawByteString', iptrw);
24278:   AsStringU', 'UnicodeString', iptrw);
24279:   AsFloat', 'MFloat', iptrw);
24280:   Func Duplicate : TRationalClass');
24281:   Proc Assign( const R : TRationalClass);
24282:   Proc Assign1( const R : MFloat);
24283:   Proc Assign2( const Numerator : Int64; const Denominator : Int64);
24284:   Proc AssignZero');
24285:   Proc AssignOne');
24286:   Func IsEqual( const R : TRationalClass) :Bool;
24287:   Func IsEqual1(const Numerator:Int64;const Denominator:Int64):Bool;
24288:   Func IsEqual2( const R : Extended) :Bool;
24289:   Func IsZero :Bool');
24290:   Func IsOne :Bool');
24291:   Proc Add( const R : TRationalClass);
24292:   Proc Add1( const V : Extended);
24293:   Proc Add2( const V : Int64);
24294:   Proc Subtract( const R : TRationalClass);
24295:   Proc Subtract1( const V : Extended);
24296:   Proc Subtract2( const V : Int64);
24297:   Proc Negate');
24298:   Proc Abs');
24299:   Func Sgn : Integer');
24300:   Proc Multiply( const R : TRationalClass);
24301:   Proc Multiply1( const V : Extended);
24302:   Proc Multiply2( const V : Int64);
24303:   Proc Divide( const R : TRationalClass);
24304:   Proc Divide1( const V : Extended);
24305:   Proc Divide2( const V : Int64);
```

```
24306:   Proc Reciprocal');
24307:   Proc Sqrt');
24308:   Proc Sqr');
24309:   Proc Power( const R : TRationalClass);
24310:   Proc Power1( const V : Int64);
24311:   Proc Power2( const V : Extended);
24312:   Proc Exp');
24313:   Proc Ln');
24314:   Proc Sin');
24315:   Proc Cos');
24316:  end;
24317: end;
24318:
24319: Proc SIRegister_TComplexClass(CL: TPSPascalCompiler);
24320: begin
24321:  //with RegClassS(CL,'TOBJECT', 'TComplexClass') do
24322:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TComplexClass') do begin
24323:    Constructor Create( const ARealPart : MFloat; const AImaginaryPart : MFloat);
24324:    RealPart', 'MFloat', iptrw);
24325:    ImaginaryPart', 'MFloat', iptrw);
24326:    AsString', 'String', iptrw);
24327:    AsStringB', 'RawByteString', iptrw);
24328:    AsStringU', 'UnicodeString', iptrw);
24329:    Proc Assign( const C : TComplex);
24330:    Proc Assign1( const V : MFloat);
24331:    Proc AssignZero');
24332:    Proc AssignI');
24333:    Proc AssignMinI');
24334:    Func Duplicate : TComplexClass');
24335:    Func IsEqual2( const C : TComplexClass) :Bool;
24336:    Func IsEqual3( const R, I : MFloat) :Bool;
24337:    Func IsReal :Bool');
24338:    Func IsZero :Bool');
24339:    Func IsI :Bool');
24340:    Proc Add4( const C : TComplex);
24341:    Proc Add5( const V : MFloat);
24342:    Proc Subtract6( const C : TComplexClass);
24343:    Proc Subtract7( const V : MFloat);
24344:    Proc Multiply8( const C : TComplexClass);
24345:    Proc Multiply9( const V : MFloat);
24346:    Proc MultiplyI');
24347:    Proc MultiplyMinI');
24348:    Proc Divide10( const C : TComplexClass);
24349:    Proc Divide11( const V : MFloat);
24350:    Proc Negate');
24351:    Func Modulo : MFloat');
24352:    Func Denom : MFloat');
24353:    Proc Conjugate');
24354:    Proc Inverse');
24355:    Proc Sqrt');
24356:    Proc Exp');
24357:    Proc Ln');
24358:    Proc Sin');
24359:    Proc Cos');
24360:    Proc Tan');
24361:    Proc Power( const C : TComplexClass);
24362:  end;
24363: end;
24364:
24365: Proc SIRegister_TMatrixClass(CL: TPSPascalCompiler);
24366: begin
24367: //with RegClassS(CL,'TOBJECT', 'TMatrixClass') do
24368:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TMatrixClass') do begin
24369:   Constructor CreateSize( const RowCount, ColCount : Integer);
24370:   Constructor CreateSquare( const N : Integer; const Identity :Bool);
24371:   //RegisterMethod('Constructor CreateDiagonal( const D : TVector);
24372:   ColCount', 'Integer', iptrw);
24373:   RowCount', 'Integer', iptrw);
24374:   Proc SetSize( const RowCount, ColCount : Integer);
24375:   Proc Clear');
24376:   Item', 'MFloat Integer Integer', iptrw);
24377:   SetDefaultPropery('Item');
24378:   Proc AssignZero');
24379:   Proc AssignIdentity');
24380:   Proc Assign( const Value : MFloat);
24381:   Proc Assign1( const M : TMatrix);
24382:   //RegisterMethod('Proc Assign2( const V : TVector);
24383:   Func Duplicate : TMatrixClass;
24384:   Func DuplicateRange( const R1, C1, R2, C2 : Integer) : TMatrixClass;
24385:   //RegisterMethod('Func DuplicateRow( const Row : Integer) : TVector');
24386:   //RegisterMethod('Func DuplicateCol( const Col : Integer) : TVector');
24387:   //RegisterMethod('Func DuplicateDiagonal : TVector');
24388:   Func IsEqual( const M : TMatrixClass) :Bool;
24389:   //RegisterMethod('Func IsEqual6( const V : TVector) :Bool;
24390:   Func IsSquare :Bool');
24391:   Func IsZero :Bool');
24392:   Func IsIdentity :Bool');
24393:   AsString', 'String', iptr);
24394:   AsStringB', 'RawByteString', iptr);
```

```
24395:   AsStringU', 'UnicodeString', iptr);
24396:   Func Trace : MFloat');
24397:   //RegisterMethod('Proc SetRow7(const Row: Integer; const V : TVector);
24398:   Proc SetRow( const Row : Integer; const Values : array of MFloat);
24399:   //RegisterMethod('Proc SetCol( const Col: Integer; const V : TVector);
24400:   Func Transpose : TMatrixClass');
24401:   Proc TransposeInPlace');
24402:   Proc Add( const M : TMatrixClass);
24403:   Proc Subtract( const M : TMatrixClass);
24404:   Proc Negate');
24405:   Proc MultiplyRow( const Row : Integer; const Value : MFloat);
24406:   Proc Multiply( const Value : MFloat);
24407:   Func Multiply1( const M : TMatrixClass) : TMatrixClass;
24408:   Proc MultiplyInPlace( const M : TMatrixClass);
24409:   Func IsOrtogonal :Bool');
24410:   Func IsIdempotent :Bool');
24411:   Func Normalise( const M : TMatrixClass) : MFloat');
24412:   Func SolveMatrix( var M : TMatrixClass) : MFloat');
24413:   Func Determinant : MFloat');
24414:   Func Inverse : TMatrixClass');
24415:   Proc InverseInPlace');
24416:   //RegisterMethod('Func SolveLinearSystem( const V : TVector) : TVector');
24417:  end;
24418: end;
24419:
24420: Proc SIRegister_TStringBuilder(CL: TPSPascalCompiler);
24421: begin
24422: //with RegClassS(CL,'TOBJECT', 'TStringBuilder') do
24423:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TStringBuilder') do begin
24424:   Constructor Create( const S :Str);
24425:   Constructor Create2( const Capacity : Integer);
24426:   RegisterProperty('Length', 'Integer', iptr);
24427:   RegisterProperty('AsString', 'String', iptrw);
24428:   Proc Clear');
24429:   Proc Assign( const S : TStringBuilder);
24430:   Proc Append( const S :Str);
24431:   Proc AppendCRLF');
24432:   Proc AppendLn( const S :Str);
24433:   Proc Append1( const S :Str; const Count : Integer);
24434:   Proc AppendCh( const C : Char);
24435:   Proc AppendCh1( const C : Char; const Count : Integer);
24436:   Proc Append2( const S : TStringBuilder);
24437:   Proc Pack');
24438:  end;
24439:  Proc TestStringBuilderClass
24440: end;
24441:
24442: * === compile-time registration functions === *)
24443: (*----------------------------------------------------------------------------*)
24444: Proc SIRegister_flcASCII(CL: TPSPascalCompiler);
24445: begin
24446: 'AsciiNULL','Char').SetString( ByteChar ( #0 ));
24447: 'AsciiSOH','Char').SetString( ByteChar ( #1 ));
24448: 'AsciiSTX','Char').SetString( ByteChar ( #2 ));
24449: 'AsciiETX','Char').SetString( ByteChar ( #3 ));
24450: 'AsciiEOT','Char').SetString( ByteChar ( #4 ));
24451: 'AsciiENQ','Char').SetString( ByteChar ( #5 ));
24452: 'AsciiACK','Char').SetString( ByteChar ( #6 ));
24453: 'AsciiBEL','Char').SetString( ByteChar ( #7 ));
24454: 'AsciiBS','Char').SetString( ByteChar ( #8 ));
24455: 'AsciiHT','Char').SetString( ByteChar ( #9 ));
24456: 'AsciiLF','Char').SetString( ByteChar ( #10 ));
24457: 'AsciiVT','Char').SetString( ByteChar ( #11 ));
24458: 'AsciiFF','Char').SetString( ByteChar ( #12 ));
24459: 'AsciiCR','Char').SetString( ByteChar ( #13 ));
24460: 'AsciiSO','Char').SetString( ByteChar ( #14 ));
24461: 'AsciiSI','Char').SetString( ByteChar ( #15 ));
24462: 'AsciiDLE','Char').SetString( ByteChar ( #16 ));
24463: 'AsciiDC1','Char').SetString( ByteChar ( #17 ));
24464: 'AsciiDC2','Char').SetString( ByteChar ( #18 ));
24465: 'AsciiDC3','Char').SetString( ByteChar ( #19 ));
24466: 'AsciiDC4','Char').SetString( ByteChar ( #20 ));
24467: 'AsciiNAK','Char').SetString( ByteChar ( #21 ));
24468: 'AsciiSYN','Char').SetString( ByteChar ( #22 ));
24469: 'AsciiETB','Char').SetString( ByteChar ( #23 ));
24470: 'AsciiCAN','Char').SetString( ByteChar ( #24 ));
24471: 'AsciiEM','Char').SetString( ByteChar ( #25 ));
24472: 'AsciiEOF','Char').SetString( ByteChar ( #26 ));
24473: 'AsciiESC','Char').SetString( ByteChar ( #27 ));
24474: 'AsciiFS','Char').SetString( ByteChar ( #28 ));
24475: 'AsciiGS','Char').SetString( ByteChar ( #29 ));
24476: 'AsciiRS','Char').SetString( ByteChar ( #30 ));
24477: 'AsciiUS','Char').SetString( ByteChar ( #31 ));
24478: 'AsciiSP','Char').SetString( ByteChar ( #32 ));
24479: 'AsciiDEL','Char').SetString( ByteChar ( #127 ));
24480: 'AsciiXON','char').SetString( AsciiDC1);
24481: 'AsciiXOFF','char').SetString( AsciiDC3);
24482: 'AsciiDecimalPoint','Char').SetString( ByteChar ( #46 ));
24483: 'AsciiComma','Char').SetString( ByteChar ( #44 ));
```

```
24484: 'AsciiBackSlash','Char').SetString( ByteChar ( #92 ));
24485: 'AsciiForwardSlash','Char').SetString( ByteChar ( #47 ));
24486: 'AsciiPercent','Char').SetString( ByteChar ( #37 ));
24487: 'AsciiAmpersand','Char').SetString( ByteChar ( #38 ));
24488: 'AsciiPlus','Char').SetString( ByteChar ( #43 ));
24489: 'AsciiMinus','Char').SetString( ByteChar ( #45 ));
24490: 'AsciiEqualSign','Char').SetString( ByteChar ( #61 ));
24491: 'AsciiSingleQuote','Char').SetString( ByteChar ( #39 ));
24492: 'AsciiDoubleQuote','Char').SetString( ByteChar ( #34 ));
24493: 'AsciiDigit0','Char').SetString( ByteChar ( #48 ));
24494: 'AsciiDigit9','Char').SetString( ByteChar ( #57 ));
24495: 'AsciiUpperA','Char').SetString( ByteChar ( #65 ));
24496: 'AsciiUpperZ','Char').SetString( ByteChar ( #90 ));
24497: 'AsciiLowerA','Char').SetString( ByteChar ( #97 ));
24498: 'AsciiLowerZ','Char').SetString( ByteChar ( #122 ));
24499: 'WideNULL','Char').SetString( WideChar ( #0 ));
24500: 'WideSOH','Char').SetString( WideChar ( #1 ));
24501: 'WideSTX','Char').SetString( WideChar ( #2 ));
24502: 'WideETX','Char').SetString( WideChar ( #3 ));
24503: 'WideEOT','Char').SetString( WideChar ( #4 ));
24504: 'WideENQ','Char').SetString( WideChar ( #5 ));
24505: 'WideACK','Char').SetString( WideChar ( #6 ));
24506: 'WideBEL','Char').SetString( WideChar ( #7 ));
24507: 'WideBS','Char').SetString( WideChar ( #8 ));
24508: 'WideHT','Char').SetString( WideChar ( #9 ));
24509: 'WideLF','Char').SetString( WideChar ( #10 ));
24510: 'WideVT','Char').SetString( WideChar ( #11 ));
24511: 'WideFF','Char').SetString( WideChar ( #12 ));
24512: 'WideCR','Char').SetString( WideChar ( #13 ));
24513: 'WideSO','Char').SetString( WideChar ( #14 ));
24514: 'WideSI','Char').SetString( WideChar ( #15 ));
24515: 'WideDLE','Char').SetString( WideChar ( #16 ));
24516: 'WideDC1','Char').SetString( WideChar ( #17 ));
24517: 'WideDC2','Char').SetString( WideChar ( #18 ));
24518: 'WideDC3','Char').SetString( WideChar ( #19 ));
24519: 'WideDC4','Char').SetString( WideChar ( #20 ));
24520: 'WideNAK','Char').SetString( WideChar ( #21 ));
24521: 'WideSYN','Char').SetString( WideChar ( #22 ));
24522: 'WideETB','Char').SetString( WideChar ( #23 ));
24523: 'WideCAN','Char').SetString( WideChar ( #24 ));
24524: 'WideEM','Char').SetString( WideChar ( #25 ));
24525: 'WideEOF','Char').SetString( WideChar ( #26 ));
24526: 'WideESC','Char').SetString( WideChar ( #27 ));
24527: 'WideFS','Char').SetString( WideChar ( #28 ));
24528: 'WideGS','Char').SetString( WideChar ( #29 ));
24529: 'WideRS','Char').SetString( WideChar ( #30 ));
24530: 'WideUS','Char').SetString( WideChar ( #31 ));
24531: 'WideSP','Char').SetString( WideChar ( #32 ));
24532: 'WideDEL','Char').SetString( WideChar ( #127 ));
24533: 'WideXON','char').SetString( WideDC1);
24534: 'WideXOFF','char').SetString( WideDC3);
24535: 'WideCRLF','String').SetString( WideString ( #13#10 ));
24536:  Func IsAsciiCharB( const C : Char) :Bool');
24537:  Func IsAsciiCharW( const C : WideChar) :Bool');
24538:  Func IsAsciiChar( const C : Char) :Bool');
24539:  //Func IsAsciiBufB( const Buf: Pointer; const Len: NativeInt):Bool');
24540:  //Func IsAsciiBufW( const Buf: Pointer; const Len  NativeInt):Bool');
24541:  Func IsAsciiStringB( const S :Str) :Bool');
24542:  Func IsAsciiStringU( const S : UnicodeString) :Bool');
24543:  Func IsAsciiString( const S :Str) :Bool');
24544:  Func AsciiHexCharValueB( const C : Char) : Integer');
24545:  Func AsciiHexCharValueW( const C : WideChar) : Integer');
24546:  Func AsciiIsHexCharB( const C : Char) :Bool');
24547:  Func AsciiIsHexCharW( const C : WideChar) :Bool');
24548:  Func AsciiDecimalCharValueB( const C : Char) : Integer');
24549:  Func AsciiDecimalCharValueW( const C : WideChar) : Integer');
24550:  Func AsciiIsDecimalCharB( const C : Char) :Bool');
24551:  Func AsciiIsDecimalCharW( const C : WideChar) :Bool');
24552:  Func AsciiOctalCharValueB( const C : Char) : Integer');
24553:  Func AsciiOctalCharValueW( const C : WideChar) : Integer');
24554:  Func AsciiIsOctalCharB( const C : Char) :Bool');
24555:  Func AsciiIsOctalCharW( const C : WideChar) :Bool');
24556:  Func CharCompareNoAsciiCaseB( const A, B : Char) : Integer');
24557:  Func CharCompareNoAsciiCaseW( const A, B : WideChar) : Integer');
24558:  Func CharCompareNoAsciiCase( const A, B : Char) : Integer');
24559:  Func CharEqualNoAsciiCaseB( const A, B : Char) :Bool');
24560:  Func CharEqualNoAsciiCaseW( const A, B : WideChar) :Bool');
24561:  Func CharEqualNoAsciiCase( const A, B : Char) :Bool');
24562:  //Func StrPCompareNoAsciiCaseB( const A, B: PByteChar; const Len: NativeInt):Integer');
24563:  //Func StrPCompareNoAsciiCaseW( const A, B: PWideChar; const Len: NativeInt):Integer');
24564:  Func StrPCompareNoAsciiCase( const A, B : PChar; const Len : NativeInt) : Integer');
24565:  Func StrCompareNoAsciiCaseA( const A, B : Ansistr) : Integer');
24566:  Func StrCompareNoAsciiCaseB( const A, B :Str) : Integer');
24567:  Func StrCompareNoAsciiCaseU( const A, B : UnicodeString) : Integer');
24568:  Func StrCompareNoAsciiCase( const A, B :Str) : Integer');
24569:  Func AsciiLowCaseB( const C : Char) : Char');
24570:  Func AsciiLowCaseW( const C : WideChar) : WideChar');
24571:  Func AsciiLowCase( const C : Char) : Char');
24572:  Func AsciiUpCaseB( const C : Char) : Char ');
```

```
24573:  Func AsciiUpCaseW( const C : WideChar) : WideChar');
24574:  Func AsciiUpCase( const C : Char) : Char');
24575:  Proc AsciiConvertUpperB( var S :Str);
24576:  Proc AsciiConvertUpperU( var S : UnicodeString);
24577:  Proc AsciiConvertUpper( var S :Str);
24578:  Proc AsciiConvertLowerB( var S :Str);
24579:  Proc AsciiConvertLowerU( var S : UnicodeString);
24580:  Proc AsciiConvertLower( var S :Str);
24581:  Func AsciiUpperCaseB( const A :Str) :Str');
24582:  Func AsciiUpperCaseU( const A : UnicodeString) : UnicodeString');
24583:  Func AsciiUpperCase( const A :Str) :Str');
24584:  Func AsciiLowerCaseB( const A :Str) :Str');
24585:  Func AsciiLowerCaseU( const A : UnicodeString) : UnicodeString');
24586:  Func AsciiLowerCase( const A :Str) :Str');
24587:  Proc AsciiConvertFirstUpB( var S : RawByteString);
24588:  Proc AsciiConvertFirstUpU( var S : UnicodeString);
24589:  Proc AsciiConvertFirstUp( var S :Str);
24590:  Func AsciiFirstUpB( const S : RawByteString) : RawByteString');
24591:  Func AsciiFirstUpU( const S : UnicodeString) : UnicodeString');
24592:  Func AsciiFirstUp( const S :Str) :Str');
24593:  Proc AsciiConvertArrayUpper( var S : TStringArray);
24594:  Proc AsciiConvertArrayLower( var S : TStringArray);
24595:  Proc TestAscii');
24596: end;
24597:
24598: Proc SIRegister_flcStringPatternMatcher(CL: TPSPascalCompiler);
24599: begin
24600:  TMatchPatternGreed', '( mpgLazy, mpgGreedy, mpgGreedyNoBacktrack);
24601:  //CL.AddDelphiFunction('Func StrZMatchPatternA(M, S: PAnsiChar; const G : TMatchPatternGreed): Integer');
24602:  //CL.AddDelphiFunction('Func StrZMatchPatternW(M, S: PWideChar; const G : TMatchPatternGreed): Integer');
24603:  Func StrEqualPatternA(const M,S: RawByteString; const G : TMatchPatternGreed) : Bool;
24604:  Func StrEqualPatternU(const M,S: UnicodeString; const G : TMatchPatternGreed) : Bool;
24605:  Func StrEqualPattern( const M, S :Str; const G : TMatchPatternGreed) :Bool');
24606:  Func StrPosPatternA(const F,S:RawByteString; var Len:Int;const StartIndex:Int;const
       G:TMatchPatternGreed):Int;
24607:  Func StrPosPatternU(const F,S:UnicodeString;var Len:Int;const StartIndex:Int const
       G:TMatchPatternGreed):Int;
24608:  Func MatchFileMaskB(const Mask,Key:RawByteString;const AsciiCaseSensitive:Bool):Bool);
24609:   TMatchQuantifier','( mqOnce, mqAny, mqLeastOnce, mqOptional );
24610:   TMatchQuantSeqOptions', '(moDeterministic, moNonGreedy );
24611:  //Func MatchQuantSeqB(var MatchPos:Integer;const MatchSeq:array of ByteCharSet;const Quant:array of
       TMatchQuantifier;const S:RawByteString;const MatchOptions TMatchQuantSeqOptions;const
       StartIndex:Integer;const StopIndex:Integer):Bool');
24612:  Proc TestPatternmatcher');
24613:  Func CharMatch(const A, B: Char; const AsciiCaseSensitive:Bool):Bool;
24614: end;
24615:
24616: https://github.com/fundamentalslib/fundamentals5/blob/master/Source/Utils/flcStrings.pas
24617: https://github.com/fundamentalslib/fundamentals5/blob/master/Source/Utils/flcUtils.pas
24618: https://raw.githubusercontent.com/fundamentalslib/fundamentals5/master/Source/Utils/flcUnicodeChar.pas
24619: Unicode Transformation Format(UTF) encoding-UTF-7, UTF-8, UTF-16, and UTF-32 - each codepoint is encoded
       as a sequence of codeunits. The size of each codeunit is determined by the encoding - 7 bits for UTF-7,8
       bits for UTF-8,16 bits for UTF-16, and 32 bits for UTF-32 (hence their names).
24620: In Delphi 2009 and later, String is an alias for UnicodeString, and Char is an alias for WideChar.
       WideChar is 16 bits. A UnicodeString holds a UTF-16 encoded string (in earlier versions of Delphi, the
       equivalent string type was WideString), and each WideChar is a UTF-16 codeunit. In UTF-16, a codepoint can
       be encoded using either 1 or 2 codeunits. 1 codeunit can encode codepoint values in the Basic Multilingual
       Plane(BMP) range - $0000 to $FFFF, inclusive. Higher codepoints require 2 codeunits, which is also known
       as a surrogate pair.
24621: D2007 supports Unicode, just not to the extent that D2009+ does. Unicode in D2007 is handled using
       WideString and the few RTL support functions that do exist.
24622: URL contains percent-encoded UTF-8 byte octets. Simply convert sequences into their binary representation
       and then use UTF8Decode() to decode the UTF-8 data to a WideString.For example:
24623:
24624: Func HexToBits(C: Char): Byte;
24625: begin
24626:   case C of
24627:     '0'..'9': Result:= Byte(Ord(C) - Ord('0'));
24628:     'a'..'f': Result:= Byte(10 + (Ord(C) - Ord('a')));
24629:     'A'..'F': Result:= Byte(10 + (Ord(C) - Ord('A')));
24630:   else
24631:     raise Exception.Create('Invalid encoding detected');
24632:   end;
24633: end;
24634:
24635: https://stackoverflow.com/questions/14107717/decode-utf-8-encoded-cyrillic-with-delphi-2007
24636: (* === compile-time registration functions === *)
24637: (*----------------------------------------------------------------------------*)
24638: Proc SIRegister_flcUnicodeChar(CL: TPSPascalCompiler);
24639: begin
24640:  WideSingleQuote','String').SetString( WideChar ( '''' ));
24641:  WideDoubleQuote','String').SetString( WideChar ( '"' ));
24642:  WideNoBreakSpace','Char').SetString( WideChar ( #$00A0 ));
24643:  'WideLineSeparator','Char').SetString( WideChar ( #$2028 ));
24644:  'WideParagraphSeparator','Char').SetString( WideChar ( #$2029 ));
24645:  WideBOM_MSB_First','Char').SetString( WideChar ( #$FFFE ));
24646:  WideBOM_LSB_First','Char').SetString( WideChar ( #$FEFF ));
24647:  WideObjectReplacement','Char).SetString( WideChar ( #$FFFC ));
24648:  WideCharReplacement','Char).SetString( WideChar ( #$FFFD ));
```

```
24649:  WideInvalid','Char).SetString( WideChar ( #$FFFF ));
24650:  WideCopyrightSign','Char).SetString( WideChar ( #$00A9 ));
24651:  WideRegisteredSign','Char).SetString( WideChar ( #$00AE ));
24652:  WideHighSurrogateFirst','Char).SetString( WideChar ( #$D800 ));
24653:  WideHighSurrogateLast','Char).SetString( WideChar ( #$DB7F ));
24654:  WideLowSurrogateFirst','Char).SetString( WideChar ( #$DC00 ));
24655:  WideLowSurrogateLast','Char).SetString( WideChar ( #$DFFF ));
24656:  WidePrivateHighSurrogateFirst',Char').SetString( WideChar ( #$DB80 ));
24657:  WidePrivateHighSurrogateLast',Char').SetString( WideChar ( #$DBFF ));
24658:  flcWideNULL','Char).SetString( WideChar ( #0 ));
24659:  flcWideBS','Char).SetString( WideChar ( #8 ));
24660:  flcWideHT','Char).SetString( WideChar ( #9 ));
24661:  'flcWideLF','Char').SetString( WideChar ( #10 ));
24662:  'flcWideVT','Char').SetString( WideChar ( #11 ));
24663:  'flcWideFF','Char').SetString( WideChar ( #12 ));
24664:  'flcWideCR','Char').SetString( WideChar ( #13 ));
24665:  'flcWideEOF','Char').SetString( WideChar ( #26 ));
24666:  'flcWideSP','Char').SetString( WideChar ( #32 ));
24667:  'UCS4_STRING_TERMINATOR','LongWord').SetUInt( $9C);
24668:  'UCS4_LF','LongWord').SetUInt( $0A);
24669:  'UCS4_CR','LongWord').SetUInt( $0D);
24670: Func UnicodeIsAsciiChar( const Ch : WideChar) :Bool');
24671: CL.AddTypeS('TUnicodeSpaceWidth','(uswNone,uswZero,uswAdjustableEmOneOver'
24672:  +'Four,uswEmOneOverTwo,uswEmOneOverThree,uswEmOneOverFour,uswEmFourOverEighteen,uswEmOneOverFive,
     uswEmOneOverSix,uswEmOne,uswWidthDigit,uswWidthPeriod,uswWidthIdeographic);
24673: Func UnicodeSpaceWidth( const Ch : WideChar) : TUnicodeSpaceWidth');
24674: Func UnicodeIsNoBreakSpace( const Ch : WideChar) :Bool');
24675: Func UnicodeIsSpaceFixedWidth( const Ch : WideChar) :Bool');
24676: Func UnicodeIsSpaceAdjustableWidth( const Ch : WideChar) :Bool');
24677: Func UnicodeIsSpace( const Ch : WideChar) :Bool');
24678: Func UnicodeIsLineBreak( const Ch : WideChar) :Bool');
24679: Func UnicodeIsWhiteSpace( const Ch : WideChar) :Bool');
24680: Func UnicodeIsControl( const Ch : WideChar) :Bool');
24681: Func UnicodeIsControlOrWhiteSpace( const Ch : WideChar) :Bool');
24682: Func UnicodeIsIgnorable( const Ch : UCS4Char) :Bool');
24683: Func UnicodeIsDash( const Ch : WideChar) :Bool');
24684: Func UnicodeIsHyphen( const Ch : WideChar) :Bool');
24685: Func UnicodeIsDashOrHyphen( const Ch : WideChar) :Bool');
24686: Func UnicodeIsFullStop0( const Ch : UCS4Char) :Bool;
24687: Func UnicodeIsFullStop1( const Ch : WideChar) :Bool;
24688: Func UnicodeIsComma( const Ch : WideChar) :Bool');
24689: Func UnicodeIsExclamationMark( const Ch : WideChar) :Bool');
24690: Func UnicodeIsQuestionMark( const Ch : WideChar) :Bool');
24691: Func UnicodeIsLeftParenthesis( const Ch : WideChar) :Bool');
24692: Func UnicodeIsLeftBracket( const Ch : WideChar) :Bool');
24693: Func UnicodeGetRightParenthesis( const LeftParenthesis : WideChar) : WideChar');
24694: Func UnicodeGetRightBracket( const LeftBracket : WideChar) : WideChar');
24695: Func UnicodeIsSingularQuotationMark( const Ch : WideChar) :Bool');
24696: Func UnicodeIsOpeningQuotationMark( const Ch : WideChar) :Bool');
24697: Func UnicodeIsClosingQuotationMark( const Ch : WideChar) :Bool');
24698: Func UnicodeGetClosingQuotationMark( const OpeningQuote : WideChar) : WideChar');
24699: Func UnicodeGetOpeningQuotationMark( const ClosingQuote : WideChar) : WideChar');
24700: Func UnicodeIsPunctuation( const Ch : WideChar) :Bool');
24701: Func UnicodeIsDecimalDigit2( const Ch : UCS4Char) :Bool;
24702: Func UnicodeIsDecimalDigit3( const Ch : WideChar) :Bool;
24703: Func UnicodeIsAsciiDecimalDigit( const Ch : WideChar) :Bool');
24704: Func UnicodeDecimalDigitValue4( const Ch : UCS4Char) : Integer;
24705: Func UnicodeDecimalDigitValue5( const Ch : WideChar) : Integer;
24706: Func UnicodeFractionCharacterValue( const Ch : WideChar; var A, B :Integer):Bool');
24707: Func UnicodeRomanNumeralValue( const Ch : WideChar) : Integer');
24708: Func UnicodeIsAsciiHexDigit( const Ch : WideChar) :Bool');
24709: Func UnicodeIsHexDigit6( const Ch : UCS4Char) :Bool;
24710: Func UnicodeIsHexDigit7( const Ch : WideChar) :Bool;
24711: Func UnicodeHexDigitValue8( const Ch : UCS4Char) : Integer;
24712: Func UnicodeHexDigitValue9( const Ch : WideChar) : Integer;
24713: Func UnicodeIsUpperCase( const Ch : WideChar) :Bool');
24714: Func UnicodeIsLowerCase( const Ch : WideChar) :Bool');
24715: Func UnicodeIsTitleCase( const Ch : WideChar) :Bool');
24716: Func UnicodeIsLetter( const Ch : WideChar) :Bool');
24717: Func UnicodeIsAlphabetic( const Ch : WideChar) :Bool');
24718: Func UnicodeUpCase( const Ch : WideChar) : WideChar');
24719: Func UnicodeLowCase( const Ch : WideChar) : WideChar');
24720: Func UnicodeCharIsEqualNoCase( const A, B : WideChar) :Bool');
24721: Func UnicodeGetCombiningClass( const Ch : WideChar) : Byte');
24722: Func UnicodeLocateFoldingUpperCase( const Ch : WideChar) : UnicodeString');
24723: Func UnicodeLocateFoldingTitleCase( const Ch : WideChar) : UnicodeString');
24724: Func UnicodeUpCaseFoldingU( const Ch : WideChar) : UnicodeString');
24725: Func UnicodeLowCaseFoldingU( const Ch : WideChar) : UnicodeString');
24726: Func UnicodeTitleCaseFoldingU( const Ch : WideChar) : UnicodeString');
24727: Func UnicodeGetCharacterDecompositionU10( const Ch : WideChar) : UnicodeString;
24728: Func UnicodeGetCharacterDecompositionU11( const Ch : UCS4Char) : UnicodeString;
24729: Func UnicodeUpperCaseFoldingU( const S : UnicodeString) : UnicodeString');
24730: Func UnicodeLowerCaseFoldingU( const S : UnicodeString) : UnicodeString');
24731: Proc TestUnicodeChar');
24732: Func HexToBits(C: Char): Byte;
24733: Func Utf8ToStr(const Source :Str) :Str;); end;
24734: unit PJResFile_Routines;
24735: Func IsIntResource( const ResID : PChar) :Bool');
24736: Func IsEqualResID2( const R1, R2 : PChar) :Bool');
```

```
24737: Func ResIDToStr( const ResID : PChar) :Str');
24738:
24739: 15 CLF_Fundamentals Testroutines 47520
24740: ------------------------------------
24741:    1 TestMathClass;
24742:    2 TestStatisticClass;
24743:    3 TestBitClass;
24744:    4 TestCharset;
24745:    5 TestTimerClass
24746:    6 TestRationalClass
24747:    7 TestComplexClass
24748:    8 TestMatrixClass;
24749:    9 TestStringBuilderClass
24750:    10 TestASCII;
24751:    11 TestASCIIRoutines;
24752:    12 TestPatternmatcher;
24753:    13 TestUnicodeChar;
24754:    14 unit uPSI_AfUtils;
24755:    15 unit uPSI_PsAPI;
24756:    16 TestVectorClass;
24757:    17 TestVectorClassExtended;
24758:    18 TestKMeans;
24759:    19 TestKMeans2;
24760:    20 TestTNNetVolume;
24761:    21 TestConvolutionAPI();
24762:    22 TestDataParallelism(NN: TNNet);
24763:    //TestTNNetVolume;
24764:
24765: ****************************************************************
24766: Release Notes maXbox 4.7.5.20 Jan 2021 mX47
24767: ****************************************************************
24768: Add 25 Units + 4 Tutorials
24769: 1277 unit uPSI_SystemsDiagram.pas  Dendron
24770: 1278 unit uPSI_qsFoundation.pas    Dendron
24771: 1279 uPSI_JclStringLists2          JCL
24772: 1280 uPSI_cInternetUtils2          FLC
24773: 1281 uPSI_cWindows.pas            FLC
24774: 1282 uPSI_flcSysUtils.pas  +TBytes utils
24775: 1283 unit uPSI_RotImg.pas          DA
24776: 1284 uPSI_SimpleImageLoader.pas    LAZ
24777: 1285 uPSI_HSLUtils.pas             LAZ
24778: 1286 uPSI_GraphicsMathLibrary.pas  EF
24779: 1287 unit uPSI_umodels.pas         DMath
24780: 1288 uPSI_flcStatistics.pas        FLC5
24781: 1289 uPSI_flcMaths.pas             FLC5
24782: 1290 uPSI_flcCharSet.pas
24783: 1291 uPSI_flcBits32.pas
24784: 1292 uPSI_flcTimers.pas
24785: 1293 uPSI_cBlaiseParserLexer.pas
24786: 1294 uPSI_flcRational.pas
24787: 1295 uPSI_flcComplex.pas
24788: 1296 unit uPSI_flcMatrix (uPSI_flcVectors.pas)
24789: 1297 unit uPSI_flcStringBuilder.pas
24790: 1298 unit PJResFile_Routines;
24791: 1299 uPSI_flcASCII.pas
24792: 1300 uPSI_flcStringPatternMatcher;
24793: 1301 unit uPSI_flcUnicodeChar.pas
24794: Totals of Func Calls: 33282
24795: SHA1: of 4.7.5.20 D82EAD01C58738887661428F94B207DB1D8FAEB5
24796: CRC32: 203C82F0 29.5 MB (31,012,768 bytes
24797: newunits: 01 RotImg.pas + uModel : TModel lib dmath.dll
24798:           02 SimpleImageLoader.pas
24799:           03 systemsdiagram.pas + fpc switch
24800:           04 qsfoundation.pas   NO FPC Vector operator
24801:           05 prediction.pas  SimulationEngine missing
24802:           06 HSLUtils.pas - color model
24803:           07 cInternetUtils.pas  - header information
24804:           08 cWindows.pas  - cstrings routines as flcSysUtils
24805:           09 flcSysUtils.pas  //2 functions with cwindows possible+ freqObj +TBytes utils
24806:           10 GraphicsMathLibrary.PAS  gml-profix
24807:           11 flcBits32.pas  //$IFDEF DEBUG} {$IFDEF TEST} Proc Test;
24808:           12 flcFloats.pas  No Floats instead: uPSI_cBlaiseParserLexer.pas
24809:           13 flcDecimal.pas  + TestClass
24810:           14 flcCharSet.pas //test include wrapper + -
24811:           15 flcComplex.pas     -Class
24812:           16 flcMaths.pas    //{$IFDEF MATHS_TEST}  Proc Test
24813:           17 flcMatrix.pas    - less TVectors
24814:           18 flcRational.pas    -Class
24815:           19 flcStatistics.pas  -Class
24816:           20 flcStringBuilder.pas - less Unicode
24817:           21 flcVectors.pas   No Vectors cause compatibility
24818:           22 flcTimers.pas     {$DEFINE TIMERS_TEST}
24819:
24820: ****************************************************************
24821: Release Notes maXbox 4.7.5.90 October 2021 mX475
24822: ****************************************************************
24823:
24824: Add 26 Units + 10 Tutorials
24825: 1403 unit uPSI_SemaphorGrids;
```

```
24826: 1404 unit uXmlDates2;
24827: 1405 unit uPSI_JclTimeZones;
24828: 1406 unit uPSI_XmlDocRssParser.pas
24829: 1407 unit uPSI_RssParser.pas
24830: 1408 uPSI_SimpleParserRSS.pas
24831: 1409 unit uPSI_SimpleRSSUtils;
24832: 1410 unit uPSI_RssModel;     _BlueHippo
24833: 1411 unit uPSI_StrUtil;      _FIBPlus
24834: 1412 unit uPSI_TAChartUtils; _TEE
24835: 1413 unit uPSI_PythonEngine.pas _P4D_Beta
24836: 1414 unit uPSI_VclPythonGUIInputOutput;
24837: 1415 unit uPSI_VarPyth;
24838: 1416 unit JclUsesUtils;
24839: 1417 unit uPSI_cParameters;
24840: 1418 unit uPSI_WDCCMisc; (uPSI_cFileTemplates);
24841: 1419 uPSI_WDCCOleVariantEnum.pas
24842: 1420 unit uPSI_WDCCWinInet.pas  _WDCC
24843: 1421 uPSI_PythonVersions.pas
24844: 1422 unit uPSI_PythonAction.pas
24845: 1423 uPSI_SingleList.pas
24846: 1424 unit uPSI_AdMeter.pas; Async Professional
24847: 1425 unit uPSI_neuralplanbuilder; CAI
24848: 1426 unit uPSI_neuralvolume.pas; CAI
24849: 1427 unit uPSI_neuralvolumev.pas; CAI
24850: 1428 unit uPSI_DoubleList4;  CapJack
24851: 1429 unit uPSI_ByteListClass;  CapJack
24852: 1430 unit uPSI_flcVectors4; FLC5
24853: 1431 unit uPSI_flcMatrix4;
24854: 1432 uPSI_CurlHttpCodes.pas
24855: 1433 unit uPSI_NeuralNetwork.pas; CAI
24856: 1434 unit unit uPSI_neuralfit; CAI
24857: 1435 unit uPSI_neuraldatasets; CAI
24858: 1436 unit uPSI_neuraldatasetsv.pas CAI
24859: 1437 uPSI_flcFloats.pas FLC5
24860:
24861: //Async Professional
24862:
24863: SIRegister_TFastStringStream(CL);
24864: Proc LoadJPEGResource(image1: TImage; aJpgImage:Str);
24865: Func CreateDOSProcessRedirected3(const CommandLine, InputFile, OutputFile, ErrMsg:Str):Bool;
24866: //Type TSysCharSet = set of Char;
24867: Func SysCharSetToStr(const C: TSysCharSet): Ansistr;
24868: Func StrToSysCharSet(const S: Ansistr): TSysCharSet;
24869: Proc MaskFPUExceptions( ExceptionsMasked :Bool; MatchPythonPrecision :Bool);
24870: Func GetOleVariantEnum( Collection : OLEVariant) : IGetOleVariantEnum');
24871: Func GetOleVariantArrEnum( Collection : OLEVariant) : IGetOleVariantEnum');
24872:
24873:
24874: Proc SIRegister_TSimpleParserRSS(CL: TPSPascalCompiler);
24875: begin
24876:   //with RegClassS(CL,'TSimpleParserBase', 'TSimpleParserRSS') do
24877:   with CL.AddClassN(CL.FindClass('TSimpleParserBase'),'TSimpleParserRSS') do begin
24878:     RegisterMethod('Proc Parse');
24879:     RegisterMethod('Proc Generate');
24880:   end;
24881: end;
24882:
24883:  SERIES_MARK_FORMATS: array [TSeriesMarksStyle] of String = (
24884:     '', '',
24885:     '%0:.9g', // smsValue
24886:     '%1:.2f%%', // smsPercent
24887:     '%2:s', // smsLabel
24888:     '%2:s %1:.2f%%', // smsLabelPercent
24889:     '%2:s %0:.9g', // smsLabelValue
24890:     '%2:s', // smsLegend: not sure what it means, left for Delphi compatibility
24891:     '%1:.2f%% of %3:g', // smsPercentTotal
24892:     '%1:.2f%% of %3:g', // smsLabelPercentTotal
24893:     '%4:.9g' // smsXValue
24894:   );
24895:
24896: Proc SIRegister_uXmlDates(CL: TPSPascalCompiler);
24897: begin
24898:   CL.AddTypeS('TXMLDateZeroOptions', '( dzoBlank, dzoZero, dzoNow );
24899:  Func GetTimeZoneOffset :Str');
24900:  Func GetTimestampWithTimeZone( const ADateTime : TDateTime) :Str');
24901:  Func GetXmlDate( const ADateTime : TDateTime; const DateZeroOption : TXMLDateZeroOptions) :Str');
24902:  Func ConvertToDelphiDateFromXml( const ADateTime :Str) : TDateTime');
24903: end;
24904:
24905: Proc SIRegister_JclTimeZones(CL: TPSPascalCompiler);
24906: begin
24907:   CL.AddTypeS('TJclTZIValueInfo', 'record Bias : Longint; StandardBias : Intege'
24908:    +'r; DaylightBias : Integer; StandardDate : TSystemTime; DaylightDate : TSystemTime; end');
24909:   //CL.AddTypeS('PJclTimeZoneRegInfo', '^TJclTimeZoneRegInfo // will not work');
24910:   TJclTimeZoneRegInfo', 'record DisplayDesc :Str; StandardName'
24911:    + :Str; DaylightName :Str; SortIndex : Integer; MapID :Str; TZI: TJclTZIValueInfo; end');
24912:   TJclTimeZoneCallBackFunc', 'Func ( const TimeZoneRec : TJclTimeZoneRegInfo) :Bool');
24913:   SIRegister_TJclTimeZoneInfo(CL);
24914:   SIRegister_TJclTimeZones(CL);
```

```
24915:   CL.AddClassN(CL.FindClass('TOBJECT'),'EDaylightSavingsNotSupported');
24916:   CL.AddClassN(CL.FindClass('TOBJECT'),'EAutoAdjustNotEnabled');
24917:  Func EnumTimeZones( CallBackFunc : TJclTimeZoneCallBackFunc) :Bool');
24918:  Func IsAutoAdjustEnabled :Bool');
24919:  Func CurrentTimeZoneSupportsDaylightSavings :Bool');
24920:  Func DateCurrentTimeZoneClocksChangeToStandard : TDateTime');
24921:  Func DateCurrentTimeZoneClocksChangeToDaylightSavings : TDateTime');
24922:  Func GetCurrentTimeZoneDescription :Str');
24923:  Func GetCurrentTimeZoneDaylightSavingsPeriod :Str');
24924:  Func GetCurrentTimeZoneGMTOffset :Str');
24925:  Func GetCurrentTimeZoneUTCBias : Integer');
24926:  Func GetWMIScheduledJobUTCTime( Time : TDateTime) :Str');
24927:  Func UTCNow : TDateTime');
24928: end;
24929:
24930: type IRSSParser = interface
24931:     ['{263159EC-F94D-4BA6-9D34-5D93277D4DDC}']
24932:     Func ParseRSSFeed(XML:Str): TRSSFeed;
24933:   end;
24934:
24935: Proc SIRegister_TXmlDocRssParser(CL: TPSPascalCompiler);
24936: begin
24937:   //with RegClassS(CL,'TInterfacedObject', 'TXmlDocRssParser') do
24938:   with CL.AddClassN(CL.FindClass('TInterfacedObject'),'TXmlDocRssParser') do begin
24939:     RegisterMethod('Func ParseRSSDate( DateStr :Str) : TDateTime');
24940:     RegisterMethod('Func ParseRSSFeed( XML :Str) : TRSSFeed');
24941:   end;
24942: end;
24943:
24944: (*----------------------------------------------------------------------------*)
24945: Proc SIRegister_RssParser(CL: TPSPascalCompiler);
24946: begin
24947:  Func ParseRSSDate( DateStr :Str) : TDateTime');
24948:  Func ParseRSSFeed( XML :Str) : TRSSFeed');
24949: end;
24950:
24951: unit SimpleRSSUtils;
24952: type TXMLTypeRSS = (xtRDFrss, xtRSSrss, xtAtomrss, xtiTunesrss);
24953: type TContentTypeRSS = (ctTextrss, ctHTMLrss, ctXHTMLrss);
24954: type TEncodingTypeRSS = (etBase64rss,etEscapedrss,etXMLrss);
24955:
24956: Func StringToLanguage(Language:Str): TLanguages;
24957: Func LanguageToString(Language: TLanguages):str;
24958: Func DecodeStringRSS(Encoding: TEncodingTypeRSS; Data:Str):Str;
24959: Proc GetSkipHours(RootNode: IXMLNode; var aSimpleRSS: TSimpleRSS);
24960: Proc GetSkipDays(RootNode: IXMLNode; var aSimpleRSS: TSimpleRSS);
24961: Proc SIRegister_SimpleRSSUtils(CL: TPSPascalCompiler);
24962:  Func StringToLanguage( Language :Str) : TLanguages');
24963:  Func LanguageToString( Language : TLanguages) :Str');
24964:  Func DecodeStringRSS( Encoding : TEncodingTypeRSS; Data :Str) :Str');
24965:  Proc GetSkipHours( RootNode : IXMLNode; var aSimpleRSS : TSimpleRSS);
24966:  Proc GetSkipDays( RootNode : IXMLNode; var aSimpleRSS : TSimpleRSS);
24967:  Func GetPageByURL(URL:Str):Str;
24968: end;
24969:
24970: (*----------------------------------------------------------------------------*)
24971: Proc SIRegister_StrUtil(CL: TPSPascalCompiler);
24972: begin
24973:   CL.AddTypeS('_TPosition', 'record X : Longint; Y : Longint; end');
24974:   CL.AddTypeS('TFastStringStream', 'TStringStream');
24975:   SIRegister_TFastStringStream(CL);
24976:   CL.AddTypeS('_TCharSet', 'set of AnsiChar');
24977:  Func _Position( AX, AY : Integer) : _TPosition');
24978:  Func _MakeStr( C : Char; N : Integer) :Str');
24979:  Func _StrAsFloat( S :Str) : double');
24980:  Func _ToClientDateFmt( D :Str; caseFmt : byte) :Str');
24981:  Func _ExtractWord( Num : integer; const Str :Str; const WordDelims : TCharSet):Str');
24982:  Func _WordCount( const S :Str; const WordDelims : TCharSet) : Integer');
24983:  Func _AnsiUpperCaseA( const s : Ansistr) : Ansistr');
24984:  Func _CompareStrWA( const S1 :Str; const S2 : Ansistr) : Integer');
24985:  Func _AnsiCompareTextAA( const S1, S2 : Ansistr) : Integer');
24986:  Func _WildStringCompare( FirstString, SecondString :Str) :Bool');
24987:  Func _MaskCompare( const aString, Mask :Str) :Bool');
24988:  Func _SQLMaskCompare( const aString, Mask :Str) :Bool;
24989:  Func _SQLMaskCompareAW( const aString : Ansistr; const Mask :Str) :Bool');
24990:  Func _SQLMaskCompareAA( const aString, Mask : Ansistr) :Bool');
24991:  Func _SQLMaskCompare1( const aString, Mask : Widestring) :Bool;
24992:  Func _TrimCLRF( const s :Str) :Str');
24993:  Func _ReplaceStr( const S, Srch, Replace :Str) :Str');
24994:  Func _ReplaceStrInSubstr( const Source String; const Srch, Replace:str;BeginPos, EndPos:integer):str');
24995:  Func _ReplaceCIStr( const S, Srch, Replace:Str; Ansi :Bool):str');
24996:  Func _ReplaceStrCIInSubstr(const Source:str; const Srch,Replace:str; BeginPos,EndPos:integer):str');
24997:  Func _FastUpperCase( const S :Str) :Str');
24998:  Proc _DoUpperCase( var S :Str; FirstPos : integer; LastPos : integer);
24999:  Proc _DoAnsiUpperCase( var S : Ansistr);
25000:  Proc _DoAnsiUpperCase1( var S :Str);
25001:  Proc _DoWideUpperCase( var S : WideString);
25002:  Proc _DoUtf8Decode( const s : AnsiChar; StrLen : integer; var ws: WideString);
25003:  Proc _DoUtf8Decode1( const s : Ansistr; var ws : WideString);
```

```
25004:  Proc _DoUtf8Decode2( const s : variant; var ws : WideString);
25005:  Func _EquelStrings( const s, s1 :Str; CaseSensitive :Bool) :Bool');
25006:  Func _iifStr( Condition :Bool; const Str1, Str2 :Str) :Str');
25007:  Func _iifVariant( Condition :Bool; const Var1, Var2 : Variant) : Variant');
25008:  Func _StrOnMask( const StrIn, MaskIn, MaskOut :Str) :Str');
25009:  Func _StrIsInteger( const Str :Str) :Bool');
25010:  Func _FormatIdentifierValue( Dialect : Integer; const Value :Str) :Str');
25011:  Func _FormatIdentifier( Dialect : Integer; const Value :Str) :Str');
25012:  Func _NormalizeName( Dialect : Integer; const Name :Str) :Str');
25013:  Func _EasyFormatIdentifier( Dialect : Integer; const Value :Str; DoEasy :Bool) :Str');
25014:  Func _EquelNames( CI :Bool; const Value, Value1 :Str) :Bool');
25015:  Func _NeedQuote( const Name :Str) :Bool');
25016:  Func _EasyNeedQuote( const Name :Str) :Bool');
25017:  Func _PosCh( aCh : Char; const s :Str) : integer');
25018:  Func _PosCh1( aCh : Char; const s :Str; StartPos : integer) : integer;
25019:  Func _PosCh2( aCh : AnsiChar; const s : Ansistr; StartPos : integer) : integer;
25020:  Func _PosCI( const Substr, Str :Str) : integer');
25021:  Func _PosExt( const Substr, Str :Str; BegSub, EndSub : TCharSet) : integer');
25022:  Func _PosExtCI( const Substr, Str :Str; BegSub, EndSub : TCharSet; AnsiUpper :Bool) : integer');
25023:  Func _PosInSubstr( const substr, Str :Str; BeginPos, EndPos : integer) : integer');
25024:  Func _PosInRight( const substr, Str :Str; BeginPos : integer) : integer');
25025:  Func _PosInSubstrCI( const SearchStr, Str :Str; BeginPos, EndPos : integer) : integer');
25026:  Func _PosInSubstrExt(const SearchStr:str;Str:str;BeginPos,EndPos:integer; BegSub, EndSub:TCharSet):int');
25027:  Func _PosInSubstrCIExt(const SearchStr,Str:Str;BeginPos, EndPos : integer; BegSub,EndSub :TCharSet):int');
25028:  Func _FirstPos( Search : array of Char; const InString :Str) : integer;
25029:  Func _FirstPos1( Search : array of AnsiChar; const InString : Ansistr) : integer;
25030:  Func _LastChar( const Str :Str) : Char');
25031:  Func _QuotedStr( const S :Str) :Str');
25032:  Func _CutQuote( const S :Str) :Str');
25033:  Func _StringInArray( const s :Str; const a : array of String) :Bool');
25034:  Func _IsBlank( const Str :Str) :Bool');
25035:  Func _IsBeginPartStr11( const PartStr, TargetStr : Ansistr) :Bool;
25036:  Func _IsBeginPartStrWA( const PartStr :Str; const TargetStr : Ansistr) :Bool');
25037:  Func _IsBeginPartStrVarA( const PartStr : Variant; const TargetStr : Ansistr) :Bool');
25038:  Func _IsBeginPartStr12( const PartStr, TargetStr : Widestring) :Bool;
25039:  Proc _DoLowerCase13( var Str :Str);
25040:  Proc _DoLowerCase14( var Str : Ansistr);
25041:  Proc _DoTrim( var Str :Str);
25042:  Func _VarTrimRight( const str : Variant) : Variant');
25043:  Proc _DoTrimRight15( var Str :Str);
25044:  Proc _DoTrimRight16( var Str : Widestring);
25045:  Proc _DoTrimRight17( var Str : Ansistr);
25046:  Proc _DoCopy18( const Source :Str; var Dest :Str; Index, Count : Integer);
25047:  Proc _DoCopy19( const Source : Ansistr; var Dest : Ansistr; Index, Count : Integer);
25048:  Func _FastTrim( const S :Str) :Str');
25049:  Func _FastCopy20( const S :Str; Index : integer; Count : Integer) :Str;
25050:  Func _FastCopy21( const S : Ansistr; Index : integer; Count : Integer) : Ansistr;
25051:  Proc _SLDifference( ASL, BSL : TStringList; ResultSL : TStrings);
25052:  Func _EmptyStrings( SL : TStrings) :Bool');
25053:  Proc _DeleteEmptyStr( Src : TStrings);
25054:  Func _NonAnsiSortCompareStrings( SL : TStringList; Index1, Index2 : Integer) : Integer');
25055:  Func _FindInDiapazon(SL:TStrList;const S:str;const StartIndex,EndIndex:int;AnsiCompare:bool;var
       Index:Int):bool);
25056:  Func _NonAnsiIndexOf( SL : TStringList; const S :Str) : integer');
25057:  Proc _GetNameAndValue( const s :Str; var Name, Value : Ansistr);
25058:  Func _StrToDateFmt( const ADate, Fmt :Str) : TDateTime');
25059:  Func _DateToSQLStr( const ADate : TDateTime) :Str');
25060:  Func _ValueFromStr( const Str :Str) :Str');
25061:  Func _WideUpperCase( const S : WideString) : WideString');
25062:  Func _Q_StrLen( P : AnsiChar) :Card');
25063:  Func _IsOldParamName( const ParamName :Str) :Bool');
25064:  Func _IsNewParamName( const ParamName :Str) :Bool');
25065:  Func _IsMasParamName( const ParamName :Str) :Bool');
25066:  Func _GUIDAsString( const AGUID : TGUID) : Ansistr');
25067:  Proc _GUIDAsStringToPChar( const AGUID : TGUID; Dest : AnsiChar);
25068:  Func _StringAsGUID( const AStr : Ansistr) : TGUID');
25069:  Func _CompareStrAndGuid( GUID : TGUID; const Str : Ansistr) : integer');
25070:  Func _CompareStrAndGuidP( GUID : TGUID; const Str : Ansistr) : integer');
25071:  Func _IsEqualGUIDs( const guid1, guid2 : TGUID) :Bool');
25072:  Func _IsNumericStr( const Str :Str) :Bool');
25073:  Func _IsEmptyStr( const Str :Str) :Bool');
25074:  Func _CompareStrTrimmed( const Str1, Str2 : PChar; Len : integer) : integer');
25075:  Func _CharInSet( C : AnsiChar; const CharSet : TSysCharSet) :Bool');
25076:  Func DecodeBase58(const Input:Str): TByteArray;
25077:  Func StreamToByteArray(Stream: TStream): TBytes;
25078:  Func InterpolateRGB(AColor1, AColor2: Integer; ACoeff: Double): Integer;
25079:  Proc SetPropDefaults(AObject: TPersistent; APropNames: array of String);
25080:  //Hashed := HashSHA256(HashSHA256(Copy(Decoded, 0, 21)));
25081:  Proc ValidateBitcoinAddress(const Address:Str; DHashSHA256: TByteArray);
25082: end;
25083:
25084: //type TPyObject = PPyObject;  this is all a part of prerelease to get a decent bufferpointer layout!
25085: //type PPyObject = TPyObject;
25086: The engine has a property, PythonEngine.InitScript:
25087: PythonEngine.InitScript.Add('import sys');
25088: PythonEngine.InitScript.Add('sys.path.insert(0, 'path/to/script');
25089:
25090: (*----------------------------------------------------------------------------*)
25091: Proc SIRegister_PythonEngine(CL: TPSPascalCompiler);
```

```
25092: begin
25093:   CL.AddTypeS('TPythonVersionProp', 'record DllName:Str; RegVersion:Str; APIVersion: Integer; end');
25094:  'PYT_METHOD_BUFFER_INCREASE','LongInt').SetInt( 10);
25095:  'PYT_MEMBER_BUFFER_INCREASE','LongInt').SetInt( 10);
25096:  'PYT_GETSET_BUFFER_INCREASE','LongInt').SetInt( 10);
25097:  'METH_VARARGS','LongWord').SetUInt( $0001);
25098:  'METH_KEYWORDS','LongWord').SetUInt( $0002);
25099:  'CO_OPTIMIZED','LongWord').SetUInt( $0001);
25100:  'CO_NEWLOCALS','LongWord').SetUInt( $0002);
25101:  'CO_VARARGS','LongWord').SetUInt( $0004);
25102:  'CO_VARKEYWORDS','LongWord').SetUInt( $0008);
25103:  Py_LT','LongInt').SetInt( 0);
25104:  Py_LE','LongInt').SetInt( 1);
25105:  Py_EQ','LongInt').SetInt( 2);
25106:  Py_NE','LongInt').SetInt( 3);
25107:  Py_GT','LongInt').SetInt( 4);
25108:  Py_GE','LongInt').SetInt( 5);
25109:   CL.AddTypeS('TRichComparisonOpcode', '( pyLT, pyLE, pyEQ, pyNE, pyGT, pyGE );
25110:   C_Long', 'Integer');  C_ULong', 'Cardinal');
25111:   C_Long', 'NativeInt'); C_ULong', 'NativeUInt');
25112:   TPFlag', '( tpfHeapType, tpfBaseType, tpfReady, tpfReadying, tpf'
25113:    +'HaveGC, tpVectorCall, tpMethodDescriptor, tpHaveVersionTag, tpValidVersion'
25114:    +'Tag, tpIsAbstract, tpLongSubclass, tpListSubClass, tpTupleSubclass, tpByte'
25115:    +'sSubclass, tpBaseExcSubclass, tpTypeSubclass );
25116:   CL.AddTypeS('TPFlags', 'set of TPFlag');
25117:   CL.AddConstantN('TPFLAGS_DEFAULT','LongInt').Value.ts32 := ord(tpfBaseType) or ord(tpHaveVersionTag);
25118:  'single_input','LongInt').SetInt( 256);
25119:  'file_input','LongInt').SetInt( 257);
25120:  'eval_input','LongInt').SetInt( 258);
25121:  'PyUnicode_WCHAR_KIND','LongInt').SetInt( 0);
25122:  'PyUnicode_1BYTE_KIND','LongInt').SetInt( 1);
25123:  'PyUnicode_2BYTE_KIND','LongInt').SetInt( 2);
25124:  'PyUnicode_4BYTE_KIND','LongInt').SetInt( 4);
25125:   T_SHORT','LongInt').SetInt( 0);
25126:  'T_INT','LongInt').SetInt( 1);
25127:  'T_LONG','LongInt').SetInt( 2);
25128:  'T_FLOAT','LongInt').SetInt( 3);
25129:  'T_DOUBLE','LongInt').SetInt( 4);
25130:  'T_STRING','LongInt').SetInt( 5);
25131:  'T_OBJECT','LongInt').SetInt( 6);
25132:  'T_CHAR','LongInt').SetInt( 7);
25133:  'T_BYTE','LongInt').SetInt( 8);
25134:  'T_UBYTE','LongInt').SetInt( 9);
25135:  'T_USHORT','LongInt').SetInt( 10);
25136:  'T_UINT','LongInt').SetInt( 11);
25137:  'T_ULONG','LongInt').SetInt( 12);
25138:  'T_STRING_INPLACE','LongInt').SetInt( 13);
25139:  'T_OBJECT_EX','LongInt').SetInt( 16);
25140:  'PY_READONLY','LongInt').SetInt( 1);
25141:  //CL.AddConstantN('RO','').SetString( READONLY);
25142:  CL.AddConstantN('READ_RESTRICTED','LongInt').SetInt( 2);
25143:  CL.AddConstantN('PY_WRITE_RESTRICTED','LongInt').SetInt( 4);
25144:   CL.AddTypeS('TPyMemberType', '( mtShort, mtInt, mtLong, mtFloat, mtDouble, mt'
25145:    +'String, mtObject, mtChar, mtByte, mtUByte, mtUShort, mtUInt, mtULong, mtStringInplace, mtObjectEx );
25146:   CL.AddTypeS('TPyMemberFlag', '( mfDefault, mfReadOnly, mfReadRestricted, mfWriteRestricted, mfRestricted
);
25147:    CL.AddTypeS('TBasicService', '( pbsGetAttr, pbsSetAttr, pbsRepr, pbsCompa'
25148:    +'re, pbsHash, pbsStr, pbsGetAttrO, pbsSetAttrO, pbsCall, pbsTraverse, pbsClear, pbs'
25149:    +'RichCompare, pbsIter, pbsIterNext );
25150:   CL.AddTypeS('TBasicServices', 'set of TBasicService);
25151:   CL.AddTypeS('TNumberService', '( pnsAdd, pnsSubtract, pnsMultiply, pnsRem'
25152:    +'ainder, pnsDivmod, pnsPower, pnsNegative, pnsPositive, pnsAbsolute, pnsInvert, pn'
25153:    +'sLShift, pnsRShift, pnsAnd, pnsXor, pnsOr, pnsInt, pnsFloat, pnsFloorDivide, pnsTr'
25154:    +'ueDivide, pnsMatrixMultiply, pnsBool );
25155:    CL.AddTypeS('TNumberServices', 'set of TNumberService');
25156:   CL.AddTypeS('TInplaceNumberService', '( pnsInplaceAdd, pnsInplaceSubtra'
25157:    +'ct, pnsInplaceMultiply, pnsInplaceRemainder, pnsInplacePower, pnsInplaceLShift'
25158:    +', pnsInplaceRShift, pnsInplaceAnd, pnsInplaceXor, pnsInplaceOr, pnsInplaceFloor'
25159:    +'Divide, pnsInplaceTrueDivide, pnsInplaceMatrixMultiply );
25160:    CL.AddTypeS('TInplaceNumberServices', 'set of TInplaceNumberService');
25161:  CL.AddConstantN('PY_CR','Char').SetString( #13);
25162:  'PY_LF','Char').SetString( #10);
25163:  'PY_TAB','Char').SetString( #09);
25164:   //TypeS('PP_frozen', '^P_frozen // will not work');
25165:   //TypeS('P_frozen', '^_frozen // will not work');
25166:   //TypeS('PPyObject', '^PyObject // will not work');
25167:   //TypeS('PPPyObject', '^PPyObject // will not work');
25168:   //TypeS('PPPPyObject', '^PPPyObject // will not work');
25169:   //TypeS('PPyTypeObject', '^PyTypeObject // will not work');
25170:   //TypeS('PPySliceObject', '^PySliceObject // will not work');
25171:   //TypeS('binaryfunc', 'integer// will not work');
25172:   CL.AddTypeS('Py_complex', 'record real : double; imag : double; end');
25173:   CL.AddTypeS('PyMethodDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25174:    +'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_method : PPyMethodDef; end');
25175:   CL.AddTypeS('PPyMemberDescrObject', '^PyMemberDescrObject // will not work');
25176:   CL.AddTypeS('PyMemberDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25177:    +'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_member : PPyMemberDef; end');
25178:   //CL.AddTypeS('PPyGetSetDescrObject', '^PyGetSetDescrObject // will not work');
25179:   CL.AddTypeS('PyGetSetDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
```

```
25180:    +'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_getset : PPyGetSetDef; end');
25181:   //CL.AddTypeS('PPyWrapperDescrObject', '^PyWrapperDescrObject // will not work');
25182:   CL.AddTypeS('PyWrapperDescrObject', 'record ob_refcnt : NativeInt; ob_type : '
25183:    +'PPyTypeObject; d_type: PPyTypeObject;d_name: PPyObject; d_base: pwrapperbase; d_wrapped:Pointer;
     end');
25184:   //CL.AddTypeS('PPyModuleDef_Base', '^PyModuleDef_Base // will not work');
25185:   CL.AddTypeS('PyModuleDef_Base', 'record ob_refcnt : NativeInt; ob_type : PPyT'
25186:    +'ypeObject; m_index : NativeInt; m_copy : PPyObject; end');
25187:   //CL.AddTypeS('PPyModuleDef', '^PyModuleDef // will not work');
25188:   CL.AddTypeS('PyModuleDef', 'record m_base : PyModuleDef_Base; m_name : PAnsiC'
25189:    +'har; m_doc : PAnsiChar; m_size : NativeInt; m_methods : PPyMethodDef; m_re'
25190:    +'load : inquiry; m_traverse : traverseproc; m_clear : inquiry; m_free : inquiry; end');
25191:   CL.AddTypeS('PyTypeObject', 'record ob_refcnt : NativeInt; ob_type : PPyTypeO'
25192:    +'bject; ob_size : NativeInt; tp_name : PAnsiChar; tp_basicsize : NativeInt;'
25193:    +' tp_itemsize : NativeInt; tp_dealloc : pydestructor; tp_vectorcall_offset '
25194:    +': NativeInt; tp_getattr : getattrfunc; tp_setattr : setattrfunc; tp_as_asy'
25195:    +'nc : Pointer; tp_repr : reprfunc; tp_as_number : PPyNumberMethods; tp_as_s'
25196:    +'equence : PPySequenceMethods; tp_as_mapping : PPyMappingMethods; tp_hash :'
25197:    +' hashfunc; tp_call : ternaryfunc; tp_str : reprfunc; tp_getattro : getattr'
25198:    +'ofunc; tp_setattro : setattrofunc; tp_as_buffer : Pointer; tp_flags : C_UL'
25199:    +'ong; tp_doc : PAnsiChar; tp_traverse : traverseproc; tp_clear : inquiry; t'
25200:    +'p_richcompare : richcmpfunc; tp_weaklistoffset : NativeInt; tp_iter : geti'
25201:    +'terfunc; tp_iternext : iternextfunc; tp_methods : PPyMethodDef; tp_members'
25202:    +' : PPyMemberDef; tp_getset : PPyGetSetDef; tp_base : PPyTypeObject; tp_dic'
25203:    +'t : PPyObject; tp_descr_get : descrgetfunc; tp_descr_set : descrsetfunc; t'
25204:    +'p_dictoffset : NativeInt; tp_init : initproc; tp_alloc : allocfunc; tp_new'
25205:    +' : newfunc; tp_free : pydestructor; tp_is_gc : inquiry; tp_bases : PPyObje'
25206:    +'ct; tp_mro : PPyObject; tp_cache : PPyObject; tp_subclasses : PPyObject; t'
25207:    +'p_weaklist : PPyObject; tp_del : PyDestructor; tp_version_tag :Card; '
25208:    +'tp_finalize : PyDestructor; tp_vectorcall : Pointer; tp_xxx1 : NativeInt; '
25209:    +'tp_xxx2 : NativeInt; tp_xxx3 : NativeInt; tp_xxx4 : NativeInt; tp_xxx5 : N'
25210:    +'ativeInt; tp_xxx6 : NativeInt; tp_xxx7 : NativeInt; tp_xxx8 : NativeInt; t'
25211:    +'p_xxx9 : NativeInt; tp_xxx10 : NativeInt; end');
25212:   CL.AddTypeS('PPyInterpreterState', '__Pointer');
25213:   CL.AddTypeS('PPyThreadState', '__Pointer');
25214:   //CL.AddTypeS('PNode', '^node // will not work');
25215:   CL.AddTypeS('node', 'record n_type : smallint; n_str : PAnsiChar; n_lineno : '
25216:    +'integer; n_col_offset : integer; n_nchildren : integer; n_child : PNode; end');
25217:   //CL.AddTypeS('PPyCompilerFlags', '^PyCompilerFlags // will not work');
25218:   CL.AddTypeS('PyCompilerFlags', 'record flags : integer; cf_feature_version:integer; end');
25219:  CL.AddConstantN('_PyDateTime_DATE_DATASIZE','LongInt').SetInt( 4);
25220:  ('_PyDateTime_TIME_DATASIZE','LongInt').SetInt( 6);
25221:  ('_PyDateTime_DATETIME_DATASIZE','LongInt').SetInt( 10);
25222:   CL.AddTypeS('PyGILState_STATE', '( PyGILState_LOCKED, PyGILState_UNLOCKED );
25223:   CL.AddClassN(CL.FindClass('TOBJECT'),'EDLLLoadError');
25224:   SIRegister_EDLLImportError(CL);
25225:   SIRegister_EPythonError(CL);
25226:    TOBJECT'),'EPyExecError');
25227:   'TOBJECT'),'EPyException');
25228:   'TOBJECT'),'EPyStandardError');
25229:   'TOBJECT'),'EPyArithmeticError');
25230:   'TOBJECT'),'EPyLookupError');
25231:   'TOBJECT'),'EPyAssertionError');
25232:   'TOBJECT'),'EPyAttributeError');
25233:   'TOBJECT'),'EPyEOFError');
25234:   'TOBJECT'),'EPyFloatingPointError');
25235:   'TOBJECT'),'EPyEnvironmentError');
25236:   'TOBJECT'),'EPyIOError');
25237:   'TOBJECT'),'EPyOSError');
25238:   'TOBJECT'),'EPyImportError');
25239:   'TOBJECT'),'EPyIndexError');
25240:   'TOBJECT'),'EPyKeyError');
25241:   'TOBJECT'),'EPyKeyboardInterrupt');
25242:   'TOBJECT'),'EPyMemoryError');
25243:   'TOBJECT'),'EPyNameError');
25244:   'TOBJECT'),'EPyOverflowError');
25245:   'TOBJECT'),'EPyRuntimeError');
25246:   'TOBJECT'),'EPyNotImplementedError');
25247:   PySyntaxError(CL);
25248:   (TOBJECT'),'EPyIndentationError');
25249:   (TOBJECT'),'EPyTabError');
25250:   (TOBJECT'),'EPySystemError');
25251:   (TOBJECT'),'EPySystemExit');
25252:   (TOBJECT'),'EPyTypeError');
25253:   (TOBJECT'),'EPyUnboundLocalError');
25254:   (TOBJECT'),'EPyValueError');
25255:   (TOBJECT'),'EPyUnicodeError');
25256:   (TOBJECT'),'UnicodeEncodeError');
25257:   (TOBJECT'),'UnicodeDecodeError');
25258:   (TOBJECT'),'UnicodeTranslateError');
25259:   (TOBJECT'),'EPyZeroDivisionError');
25260:   (TOBJECT'),'EPyStopIteration');
25261:   (TOBJECT'),'EPyWarning');
25262:   (TOBJECT'),'EPyUserWarning');
25263:   (TOBJECT'),'EPyDeprecationWarning');
25264:   (TOBJECT'),'PendingDeprecationWarning');
25265:   (TOBJECT'),'FutureWarning');
25266:   (TOBJECT'),'EPySyntaxWarning');
25267:   (TOBJECT'),'EPyRuntimeWarning');
```

```
25268:   (TOBJECT'),'EPyReferenceError');
25269:   (TOBJECT'),'EPyWindowsError');
25270:   CL.AddConstantN('kMaxLines','LongInt').SetInt( 1000);
25271:   CL.AddConstantN('kMaxLineLength','LongInt').SetInt( 256);
25272:   CL.AddTypeS('TSendDataEvent', 'Proc ( Sender : TObject; const Data : Ansistr);
25273:   TReceiveDataEvent', 'Proc ( Sender : TObject; var Data : Ansistr);
25274:   TSendUniDataEvent', 'Proc ( Sender : TObject; const Data : UnicodeString);
25275:   TReceiveUniDataEvent', 'Proc ( Sender : TObject; var Data : UnicodeString);
25276:   IOChar', 'WideChar');
25277:   IOString', 'UnicodeString');
25278:   TIOStringList', 'TStringList');
25279:   SIRegister_TPythonInputOutput(CL);
25280:   SIRegister_TDynamicDll(CL);
25281:   SIRegister_TPythonInterface(CL);
25282:   CL.AddTypeS('TDatetimeConversionMode', '( dcmToTuple, dcmToDatetime );
25283:  //CL.AddConstantN('DEFAULT_DATETIME_CONVERSION_MODE','').SetString( dcmToTuple);
25284:   CL.AddClassN(CL.FindClass('TOBJECT'),'TEngineClient');
25285:   CL.AddTypeS('TPathInitializationEvent', 'Proc ( Sender : TObject; var Path :Str);
25286:   //CL.AddTypeS('TSysPathInitEvent', 'Proc ( Sender : TObject; PathList : PPyObject);
25287:   CL.AddTypeS('TPythonFlag','(pfDebug,pfInteractive,pfNoSite,pfOptimize,pfVerbose,pfFrozenFlag
25288:                              ,pfIgnoreEnvironmentFlag);
25289:   CL.AddTypeS('TPythonFlags', 'set of TPythonFlag');
25290:   SIRegister_TTracebackItem(CL);
25291:   SIRegister_TPythonTraceback(CL);
25292:   SIRegister_TPythonEngine(CL);
25293:   SIRegister_TEngineClient(CL);
25294:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMethodsContainer');
25295:   CL.AddClassN(CL.FindClass('TOBJECT'),'TEventDefs');
25296:   SIRegister_TEventDef(CL);
25297:   SIRegister_TEventDefs(CL);
25298:   SIRegister_TMethodsContainer(CL);
25299:   SIRegister_TMembersContainer(CL);
25300:   SIRegister_TGetSetContainer(CL);
25301:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPythonModule');
25302:   CL.AddClassN(CL.FindClass('TOBJECT'),'TErrors');
25303:   CL.AddTypeS('TErrorType', '( etString, etClass );
25304:   SIRegister_TParentClassError(CL);
25305:   SIRegister_TError(CL);
25306:   SIRegister_TErrors(CL);
25307:   SIRegister_TPythonModule(CL);
25308:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPythonType');
25309:   SIRegister_TPyObject(CL);
25310:   //CL.AddTypeS('TPyObjectClass', 'class of TPyObject');
25311:   CL.AddTypeS('TMappingService', '( msLength, msSubscript, msAssSubscript );
25312:   CL.AddTypeS('TMappingServices', 'set of TMappingService');
25313:   //CL.AddTypeS('TJclEmailReadOptions', 'set of TJclEmailReadOption');
25314:   SIRegister_TTypeServices(CL);
25315:   SIRegister_TPythonType(CL);
25316:   CL.AddTypeS('TGetDataEvent', 'Proc ( Sender : TObject; var Data : Variant);
25317:   CL.AddTypeS('TSetDataEvent', 'Proc ( Sender : TObject; Data : Variant);
25318:   //CL.AddTypeS('TExtGetDataEvent', 'Proc ( Sender : TObject; var Data : PPyObject);
25319:   //CL.AddTypeS('TExtSetDataEvent', 'Proc ( Sender : TObject; Data : PPyObject);
25320:   SIRegister_TPythonDelphiVar(CL);
25321:   SIRegister_TPyVar(CL);
25322:   CL.AddTypeS('TThreadExecMode', '( emNewState, emNewInterpreter );
25323:   SIRegister_TPythonThread(CL);
25324:  {('Func pyio_write( self, args : PPyObject) : PPyObject');
25325:  C'Func pyio_read( self, args : PPyObject) : PPyObject');
25326:  C'Func pyio_SetDelayWrites( self, args : PPyObject) : PPyObject');
25327:  C'Func pyio_SetMaxLines( self, args : PPyObject) : PPyObject');
25328:  C'Func pyio_GetTypesStats( self, args : PPyObject) : PPyObject');  }
25329:  Func GetPythonEngine : TPythonEngine');
25330:  Func PythonOK :Bool');
25331:  //Func PythonToDelphi( obj : PPyObject) : TPyObject');
25332:  //Func IsDelphiObject( obj : PPyObject) :Bool');
25333: // Proc PyObjectDestructor( pSelf : PPyObject);
25334:  //Proc FreeSubtypeInst( ob : PPyObject);
25335:  //Func PyType_HasFeature( AType : PPyTypeObject; AFlag : Integer) :Bool');
25336:  Func SysVersionFromDLLName( const DLLFileName :Str) :Str');
25337:  Proc PythonVersionFromDLLName( LibName :Str; out MajorVersion, MinorVersion : integer);
25338:  Func IsPythonVersionRegistered(PythonVersion:str; out InstallPath:tring;out AllUserInstall:Bool) Bool');
25339:  Proc MaskFPUExceptions( ExceptionsMasked :Bool; MatchPythonPrecision :Bool);
25340:  Func CleanString13( const s : Ansistr; AppendLF :Bool) : Ansistr;
25341:  Func CleanString14( const s : UnicodeString; AppendLF :Bool) : UnicodeString;
25342:  Proc setgPythonEngine(mypyengine: TPythonEngine);
25343:  Func getgPythonEngine: TPythonEngine;
25344: end;
25345:
25346: Proc SIRegister_VclPythonGUIInputOutput(CL: TPSPascalCompiler);
25347: begin
25348:  CL.AddConstantN('WM_WriteOutput','LongInt').SetInt( WM_USER + 1);
25349:   SIRegister_TPythonGUIInputOutput(CL);
25350:  CL.AddDelphiFunction('Proc Register');
25351: end;
25352:
25353: (*----------------------------------------------------------------------------*)
25354: Proc SIRegister_TDynamicDll(CL: TPSPascalCompiler);
25355: begin
25356:   //with RegClassS(CL,'TComponent', 'TDynamicDll') do
```

```
25357:   with CL.AddClassN(CL.FindClass('TComponent'),'TDynamicDll') do begin
25358:     Constructor Create( AOwner : TComponent);
25359:     Proc Free;
25360:     Proc OpenDll( const aDllName :Str);
25361:     Func IsHandleValid :Bool');
25362:     Proc LoadDll');
25363:     Proc UnloadDll');
25364:     Proc Quit');
25365:     RAutoLoad', 'Boolean', iptrw);
25366:     RAutoUnload', 'Boolean', iptrw);
25367:     RDllName', 'string', iptrw);
25368:     RDllPath', 'string', iptrw);
25369:     RAPIVersion', 'Integer', iptrw);
25370:     RRegVersion', 'string', iptrw);
25371:     RFatalAbort', 'Boolean', iptrw);
25372:     RFatalMsgDlg', 'Boolean', iptrw);
25373:     RUseLastKnownVersion', 'Boolean', iptrw);
25374:     ROnAfterLoad', 'TNotifyEvent', iptrw);
25375:     ROnBeforeLoad', 'TNotifyEvent', iptrw);
25376:     ROnBeforeUnload', 'TNotifyEvent', iptrw);
25377:   end;
25378: end;
25379:
25380: (*----------------------------------------------------------------------------*)
25381: Proc SIRegister_TPythonEngine(CL: TPSPascalCompiler);
25382: begin
25383:   //with RegClassS(CL,'TPythonInterface', 'TPythonEngine') do
25384:   with CL.AddClassN(CL.FindClass('TPythonInterface'),'TPythonEngine') do begin
25385:     Constructor Create( AOwner : TComponent);
25386:     Proc Free;
25387:     Proc SetPythonHome( const PythonHome : UnicodeString);
25388:     Proc SetProgramName( const ProgramName : UnicodeString);
25389:     Func IsType( ob : PPyObject; obt : PPyTypeObject) :Bool');
25390:     Func Run_CommandAsString(const command:Ansistr; mode:Integer):str');
25391:     Func Run_CommandAsObject(const command : Ansistr; mode : Integer) : PPyObject');
25392:     Func Run_CommandAsObjectWithDict(const command:Ansistr;mode:Integer;locals,globals:PPyObject):
      PPyObject');
25393:     Func EncodeString1( const str : UnicodeString) : Ansistr;;
25394:     Func EncodeString2( const str : Ansistr) : Ansistr;
25395:     Func EncodeWindowsFilePath( const str :Str) : Ansistr');
25396:     Proc Initialize2;
25397:     Proc DoOpenDll2(const aDllName :Str);
25398:     Proc SetProgramArgs2;
25399:     Proc ExecString( const command : Ansistr);
25400:     Proc ExecStr( const command : Ansistr);       //alias
25401:     Proc ExecString3( const command : Ansistr);   //alias
25402:     Proc ExecStrings4( strings : TStrings);
25403:     Proc ExecStrings( strings : TStrings);     //alias
25404:     Func EvalString5( const command : Ansistr) : PPyObject;
25405:     Func EvalStringAsStr( const command : Ansistr) :Str');
25406:     Func EvalStr( const command : Ansistr) :Str');
25407:     Func EvalStrings6( strings : TStrings) : PPyObject;
25408:     Proc ExecString7(const command:Ansistr; locals,globals:PPyObject);
25409:     Proc ExecStrings8( strings : TStrings; locals, globals : PPyObject);
25410:     Func EvalString9(const command:Ansistr; locals, globals : PPyObject): PPyObject;
25411:     Func EvalStrings10( strings : TStrings; locals, globals : PPyObject): PPyObject;
25412:     Func EvalStringsAsStr( strings : TStrings) :Str');
25413:     Func EvalPyFunction( pyfunc, pyargs : PPyObject) : Variant');
25414:     Func EvalFunction(pyfunc:PPyObject;const args:array of const): Variant');
25415:     Func EvalFunctionNoArgs( pyfunc : PPyObject) : Variant');
25416:     Func CheckEvalSyntax( const str : Ansistr) :Bool');
25417:     Func CheckExecSyntax( const str : Ansistr) :Bool');
25418:     Func CheckSyntax( const str : Ansistr; mode : Integer) :Bool');
25419:     Proc RaiseError');
25420:     Func PyObjectAsString( obj : PPyObject) :Str');
25421:     Proc DoRedirectIO');
25422:     Proc AddClient( client : TEngineClient);
25423:     Proc RemoveClient( client : TEngineClient);
25424:     Func FindClient( const aName :Str) : TEngineClient');
25425:     Func TypeByName( const aTypeName : Ansistr) : PPyTypeObject');
25426:     Func ModuleByName( const aModuleName : Ansistr) : PPyObject');
25427:     Func MethodsByName( const aMethodsContainer :Str) : PPyMethodDef');
25428:     Func VariantAsPyObject( const V : Variant) : PPyObject');
25429:     Func PyObjectAsVariant( obj : PPyObject) : Variant');
25430:     Func VarRecAsPyObject( const v : TVarRec) : PPyObject');
25431:     Func MakePyTuple( const objects : array of PPyObject) : PPyObject');
25432:     Func MakePyList( const objects : array of PPyObject) : PPyObject');
25433:     Func ArrayToPyTuple( const items : array of const) : PPyObject');
25434:     Func ArrayToPyList( const items : array of const) : PPyObject');
25435:     Func ArrayToPyDict( const items : array of const) : PPyObject');
25436:     Func StringsToPyList( strings : TStrings) : PPyObject');
25437:     Func StringsToPyTuple( strings : TStrings) : PPyObject');
25438:     Proc PyListToStrings( list : PPyObject; strings : TStrings);
25439:     Proc PyTupleToStrings( tuple : PPyObject; strings : TStrings);
25440:     Func ReturnNone : PPyObject');
25441:     Func ReturnTrue : PPyObject');
25442:     Func ReturnFalse : PPyObject');
25443:     Func FindModule( const ModuleName : Ansistr) : PPyObject');
25444:     Func FindFunction( const ModuleName, FuncName : Ansistr) : PPyObject');
```

```
25445:     Func SetToList( data : Pointer; size : Integer) : PPyObject');
25446:     Proc ListToSet( List : PPyObject; data : Pointer; size : Integer);
25447:     Proc CheckError( ACatchStopEx :Bool);
25448:     Func GetMainModule : PPyObject');
25449:     Func PyTimeStruct_Check( obj : PPyObject) :Bool');
25450:     Func PyDate_Check( obj : PPyObject) :Bool');
25451:     Func PyDate_CheckExact( obj : PPyObject) :Bool');
25452:     Func PyDateTime_Check( obj : PPyObject) :Bool');
25453:     Func PyDateTime_CheckExact( obj : PPyObject) :Bool');
25454:     Func PyTime_Check( obj : PPyObject) :Bool');
25455:     Func PyTime_CheckExact( obj : PPyObject) :Bool');
25456:     Func PyDelta_Check( obj : PPyObject) :Bool');
25457:     Func PyDelta_CheckExact( obj : PPyObject) :Bool');
25458:     Func PyTZInfo_Check( obj : PPyObject) :Bool');
25459:     Func PyTZInfo_CheckExact( obj : PPyObject) :Bool');
25460:     Func PyUnicodeFromString11( const AString : UnicodeString) : PPyObject;
25461:     Func PyUnicodeFromString12( const AString : Ansistr) : PPyObject;
25462:     Func PyUnicodeAsString( obj : PPyObject) : UnicodeString');
25463:     Func PyUnicodeAsUTF8String( obj : PPyObject) : RawByteString');
25464:     Func PyBytesAsAnsistr( obj : PPyObject) : Ansistr');
25465:     ClientCount', 'Integer', iptr);
25466:     Clients', 'TEngineClient Integer', iptr);
25467:     ExecModule', 'Ansistr', iptrw);
25468:     ThreadState', 'PPyThreadState', iptr);
25469:     Traceback', 'TPythonTraceback', iptr);
25470:     LocalVars', 'PPyObject', iptrw);
25471:     GlobalVars', 'PPyObject', iptrw);
25472:     IOPythonModule', 'TObject', iptr);
25473:     PythonHome', 'UnicodeString', iptrw);
25474:     ProgramName', 'UnicodeString', iptrw);
25475:     AutoFinalize', 'Boolean', iptrw);
25476:     VenvPythonExe', 'string', iptrw);
25477:     DatetimeConversionMode', 'TDatetimeConversionMode', iptrw);
25478:     InitScript', 'TStrings', iptrw);
25479:     InitThreads', 'Boolean', iptrw);
25480:     IO', 'TPythonInputOutput', iptrw);
25481:     PyFlags', 'TPythonFlags', iptrw);
25482:     RedirectIO', 'Boolean', iptrw);
25483:     UseWindowsConsole', 'Boolean', iptrw);
25484:     OnAfterInit', 'TNotifyEvent', iptrw);
25485:     OnPathInitialization', 'TPathInitializationEvent', iptrw);
25486:     OnSysPathInit', 'TSysPathInitEvent', iptrw);
25487:   end;
25488: end;
25489:
25490: Proc SIRegister_TPythonModule(CL: TPSPascalCompiler);
25491: begin
25492:   //with RegClassS(CL,'TMethodsContainer', 'TPythonModule') do
25493:   with CL.AddClassN(CL.FindClass('TMethodsContainer'),'TPythonModule') do  begin
25494:     Constructor Create( AOwner : TComponent);
25495:     Proc MakeModule');
25496:     Proc DefineDocString');
25497:     Proc Initialize');
25498:     Proc InitializeForNewInterpreter');
25499:     Proc AddClient( client : TEngineClient);
25500:     Func ErrorByName( const AName : Ansistr) : TError');
25501:     Proc RaiseError( const error, msg : Ansistr);
25502:     Proc RaiseErrorFmt(const error,format:Ansistr;const Args:array of const);
25503:     Proc RaiseErrorObj(const error, msg : Ansistr; obj : PPyObject);
25504:     Proc BuildErrors');
25505:     Proc SetVar( const varName : Ansistr; value : PPyObject);
25506:     Func GetVar( const varName : Ansistr) : PPyObject');
25507:     Proc DeleteVar( const varName : Ansistr);
25508:     Proc ClearVars');
25509:     Proc SetVarFromVariant(const varName:Ansistr;const value:Variant);
25510:     Func GetVarAsVariant( const varName : Ansistr) : Variant');
25511:     Module', 'PPyObject', iptr);
25512:     Clients', 'TEngineClient Integer', iptr);
25513:     ClientCount', 'Integer', iptr);
25514:     DocString', 'TStringList', iptrw);
25515:     ModuleName', 'Ansistr', iptrw);
25516:     Errors', 'TErrors', iptrw);
25517:     OnAfterInitialization', 'TNotifyEvent', iptrw);
25518:   end;
25519: end;
25520:
25521: Example:
25522: Proc initDemodll;
25523: begin   //mode: eval_input, file_input
25524:   try
25525:     gEngine:= TPythonEngine.Create(nil);
25526:     gEngine.AutoFinalize:= False;
25527:     gEngine.LoadDll;
25528:     gModule:= TPythonModule.Create(nil);
25529:     gModule.Engine:= gEngine;
25530:     gModule.ModuleName:= 'pydemodll';
25531:     //gModule.AddMethod( 'add', @Add, 'add(a,b) -> a+b' );
25532:    //  The engine has a property, PythonEngine.InitScript:
25533:     gEngine.InitScript.Add('import sys');
```

```
25534:     gEngine.InitScript.Add('sys.path.insert(0, "path/to/script");
25535:     //gModule.Initialize;
25536:     writeln('PythonOK: in init '+botostr(PythonOK));
25537:     //gEngine.EvalStringAsStr(pycmd);
25538:   except
25539:      gEngine.raiseError;
25540:      writeln(ExceptionToString(ExceptionType, ExceptionParam));
25541:      gEngine.Free;
25542:      gModule.Free;
25543:   end;
25544: end;
25545:
25546: const LB = CR+LF;
25547: const PyModule =
25548:          'def printData(data): '+#13#10+
25549:          '  return data+data+"/n"';
25550:
25551: const PYCMD = 'print("this is box")'+LB+
25552:               'import sys'+LB+
25553:               'f=open(r"C:\maXbox\maxbox3\maxbox3\maXbox3\examples\pytest2.txt","w")'+LB+
25554:               'f.write("Hello PyWorld_mX4, \n")'+LB+
25555:               'f.write("This data will be written on the file.")'+LB+
25556:               'f.close()';
25557:
25558: Proc PYLaz_P4D_Demo;
25559: //https://wiki.freepascal.org/Python4Delphi
25560: var eng : TPythonEngine;
25561:     Out1: TPythonGUIInputOutput;
25562: begin
25563:   eng:= TPythonEngine.Create(Nil);
25564:   Out1:= TPythonGUIInputOutput.create(nil)
25565:   Out1.output:= pyMemo; //debugout.output;  //memo2;
25566:   Out1.RawOutput:= False;
25567:   Out1.UnicodeIO:= False;
25568:   Out1.maxlines:= 20;
25569:   out1.displaystring('this string')
25570:   //eng.IO:= Out1;
25571:   Out1.writeline('draw the line');
25572:   eng.LoadDll;
25573:   eng.IO:= Out1;
25574:     if eng.IsHandleValid then begin
25575:       writeln('DLLhandle: '+botostr(eng.IsHandleValid))
25576:       WriteLn('evens: '+ eng.EvalStringAsStr('[x*2 for x in range(10)]'));
25577:       WriteLn('gauss: '+ eng.EvalStr('sum([x for x in range(101)])'));   //alias
25578:       writeln('syncheck '+
25579:           botostr(eng.CheckEvalSyntax('print("powers:",[x**2 for x in range(10)])')));
25580:       eng.ExecString('print("powers:",[x**2 for x in range(10)]);
25581:       eng.ExecStr(PYCMD);
25582:       writeln('ExecSynCheck2 '+
25583:            botostr(eng.CheckExecSyntax(filetostring(PYSCRIPT))));
25584:       eng.ExecString(filetostring(PYSCRIPT));
25585:     end
25586:   else writeln('invalid library handle! '+Getlasterrortext);
25587:   writeln('PythonOK '+botostr(PythonOK));
25588:   out1.free;
25589:   //pyImport(PyModule);
25590:   eng.free;
25591: end;
25592:
25593: //python template
25594: with TPythonEngine.Create(Nil) do begin
25595:   pythonhome:= 'C:\Users\max\AppData\Local\Programs\Python\Python36-32\';
25596:   try
25597:     loadDLL;
25598:     Println('Decimal: '+
25599:           EvalStr('__import__("decimal").Decimal(0.1)'));
25600:   except
25601:     raiseError;
25602:   finally
25603:     free;
25604:   end;
25605:
25606:  // This is a new method in VarPyth, but I'll put it here just in case....
25607: function VarPyToStrings(const AValue : Variant; const AStrings: TStrings): Integer;
25608: begin
25609:   Assert(Assigned(AStrings));
25610:   if VarIsPythonList(AValue) then
25611:     GetPythonEngine.PyListToStrings(
25612:       ExtractPythonObjectFrom(AValue), AStrings)
25613:   else
25614:     raise Exception.Create('Python List expected: ' + _type(AValue));
25615:   Result := AStrings.Count;
25616: end;
25617:
25618: procedure TForm16.Button1Click(Sender: TObject);
25619: begin
25620:   with NLTK1 do begin
25621:     nltk.download('punkt');
25622:     nltk.download('averaged_perceptron_tagger');
```

```
25623:     nltk.download('maxent_ne_chunker');
25624:     nltk.download('words');
25625:     nltk.download('treebank');
25626:
25627:     var tokens := nltk.word_tokenize(memo1.lines.Text);
25628:     VarPyToStrings(tokens, lbTokens.Items);
25629:
25630: Proc SIRegister_VarPyth(CL: TPSPascalCompiler);
25631: begin
25632:  CL.AddTypeS('TSequenceType', '( pystTuple, pystList );
25633:  Func VarPythonCreate0( AObject : TPyObject) : Variant;
25634:  Func VarPythonCreate1( const AValue : Variant) : Variant;
25635:  Func VarPythonCreate2( const AValues: array of const; ASequenceType : TSequenceType) : Variant;
25636:  Func VarPythonEval( const APythonExpression : Ansistr) : Variant');
25637:  Func VarPython : TVarType');
25638:  Func VarIsPython( const AValue : Variant) :Bool');
25639:  Func VarAsPython( const AValue : Variant) : Variant');
25640:  Func ExtractPythonObjectFrom( const AValue : Variant) : TPyObject');
25641:  Func VarIsSame( const A, B : Variant) :Bool');
25642:  Func VarIsSameType( const A, B : Variant) :Bool');
25643:  Func VarIsPythonSequence( const AValue : Variant) :Bool');
25644:  Func VarIsPythonMapping( const AValue : Variant) :Bool');
25645:  Func VarIsPythonNumber( const AValue : Variant) :Bool');
25646:  Func VarIsPythonString( const AValue : Variant) :Bool');
25647:  Func VarIsPythonInteger( const AValue : Variant) :Bool');
25648:  Func VarIsPythonFloat( const AValue : Variant) :Bool');
25649:  Func VarIsPythonTuple( const AValue : Variant) :Bool');
25650:  Func VarIsPythonList( const AValue : Variant) :Bool');
25651:  Func VarIsPythonDict( const AValue : Variant) :Bool');
25652:  Func VarIsPythonClass( const AValue : Variant) :Bool');
25653:  Func VarIsPythonMethod( const AValue : Variant) :Bool');
25654:  Func VarIsPythonFunction( const AValue : Variant) :Bool');
25655:  Func VarIsPythonModule( const AValue : Variant) :Bool');
25656:  Func VarIsPythonCallable( const AValue : Variant) :Bool');
25657:  Func VarIsPythonIterator( const AValue : Variant) :Bool');
25658:  Func VarIsPythonUnicode( const AValue : Variant) :Bool');
25659:  Func VarIsPythonDateTime( const AValue : Variant) :Bool');
25660:  Func VarIsPythonDate( const AValue : Variant) :Bool');
25661:  Func VarIsPythonTime( const AValue : Variant) :Bool');
25662:  Func VarIsPythonDateTimeDelta( const AValue : Variant) :Bool');
25663:  Func VarIsPythonTZInfo( const AValue : Variant) :Bool');
25664:  Func VarIsBool( const AValue : Variant) :Bool');
25665:  Func VarIsEnum( const AValue : Variant) :Bool');
25666:  Func VarIsInstanceOf( const AInstance, AClass : Variant) :Bool');
25667:  Func VarIsSubclassOf( const ADerived, AClass : Variant) :Bool');
25668:  Func VarIsSubtypeOf( const ADerived, AType : Variant) :Bool');
25669:  Func VarIsNone( const AValue : Variant) :Bool');
25670:  Func VarIsTrue( const AValue : Variant) :Bool');
25671:  Func VarModuleHasObject( const AModule : Variant; aObj: Ansistr):Bool');
25672:  Func NewPythonList( const ASize : Integer) : Variant');
25673:  Func NewPythonTuple( const ASize : Integer) : Variant');
25674:  Func NewPythonDict : Variant');
25675:  Func VarPythonAsString( AValue : Variant) :Str');
25676:  Func VarPythonToVariant( AValue : Variant) : Variant');
25677:  Func PyNone : Variant');
25678:  Func pyEllipsis : Variant');
25679:  Func pyMainModule : Variant');
25680:  Func pyBuiltinModule : Variant');
25681:  Func pySysModule : Variant');
25682:  Func ypDatetimeModule : Variant');
25683:  Func PyImport( const AModule : Ansistr) : Variant');
25684:  Func pylen( const AValue : Variant) : NativeInt');
25685:  Func py_type( const AValue : Variant) : Variant');
25686:  Func pyiter( const AValue : Variant) : Variant');
25687:   CL.AddTypeS(TVarPyEnumerator', 'record FIterator : Variant; FCurrent : Variant; end');
25688:   CL.AddTypeS(TVarPyEnumerateHelper', 'record FIterable : Variant; end');
25689:  Func VarPyIterate( const AValue : Variant) : TVarPyEnumerateHelper');
25690: end;
25691:
25692: Proc SIRegister_WDCCMisc(CL: TPSPascalCompiler);  //WMI Delphi Code Creator
25693: begin   //https://github.com/RRUZ/wmi-delphi-code-creator/tree/master/Units/Misc
25694:  TProcLog', 'Proc ( const Log :Str);
25695:  Proc wdc_CaptureConsoleOutput( const lpCommandLine :Str; OutPutList : TStrings);
25696:  Proc wdc_MsgWarning( const Msg :Str);
25697:  Proc wdc_MsgInformation( const Msg :Str);
25698:  Func wdc_MsgQuestion( const Msg :Str) :Bool');
25699:  Func wdc_GetFileVersion( const FileName :Str) :Str');
25700:  Func wdc_GetFileDescription( const FileName :Str) :Str');
25701:  Func wdc_GetTempDirectory :Str');
25702:  Func wdc_GetWindowsDirectory :Str');
25703:  Func wdc_GetSpecialFolder( const CSIDL : integer) :Str');
25704:  Func wdc_IsWow64 :Bool');
25705:  Func wdc_CopyDir( const fromDir, toDir :Str) :Bool');
25706:  Proc wdc_SetGridColumnWidths( DbGrid : TDBGrid);
25707:  Func wdc_Ping( const Address :Str; Retries, BufferSize : Word; Log : TStrings) :Bool');
25708:  Proc wdc_ScaleImage32( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double);
25709:  Proc wdc_ExtractIconFile( Icon : TIcon; const FileName :Str; IconType :Card);
25710:  Proc wdc_ExtractBitmapFile( Bmp : TBitmap; const FileName :Str; IconType :Card);
25711:  Proc wdc_ExtractBitmapFile32( Bmp : TBitmap; const FileName :Str; IconType :Card);
```

```
25712:  Proc wdc_CheckForUpdates( Silent :Bool);
25713: end;
25714:
25715: (*----------------------------------------------------------------------------*)
25716: Proc SIRegister_WDCCOleVariantEnum(CL: TPSPascalCompiler);
25717: begin
25718:   SIRegister_IOleVariantEnum(CL);
25719:   SIRegister_IGetOleVariantEnum(CL);
25720:   SIRegister_TOleVariantEnum(CL);
25721:   SIRegister_TOleVariantArrayEnum(CL);
25722:  Func GetOleVariantEnum( Collection : OLEVariant) : IGetOleVariantEnum');
25723:  Func GetOleVariantArrEnum( Collection : OLEVariant) : IGetOleVariantEnum');
25724: end;
25725:
25726: (*----------------------------------------------------------------------------*)
25727: Proc SIRegister_WDCCWinInet(CL: TPSPascalCompiler);
25728: begin
25729:  CL.AddConstantN('WM_UWININET_THREAD_FINISHED','LongInt').SetInt( WM_USER + 669);
25730:  CL.AddConstantN('WM_UWININET_THREAD_CANCELLED','LongInt').SetInt( WM_USER + 670);
25731:  CL.AddTypeS('TuWinInetProcCallBack', 'Proc ( BytesRead : Integer);
25732:   SIRegister_TWinINetGetThread(CL);
25733:  Func wdc_GetRemoteFileSize( const Url :Str) : Integer');
25734:  Proc wdc_WinInet_HttpGet( const Url :Str; Stream : TStream; CallBack : TuWinInetProcCallBack);
25735:  Func wdc_WinInet_HttpGet2( const Url :Str; CallBack : TuWinInetProcCallBack) :Str;
25736:  Func wdc_GetWinInetError( ErrorCode :Card) :Str');
25737: end;
25738:
25739: Proc SIRegister_PythonVersions(CL: TPSPascalCompiler);
25740: begin
25741:   SIRegister_TPythonVersion(CL);
25742:   CL.AddTypeS('TPythonVersions', 'array of TPythonVersion');
25743:  Func CompareVersions( A, B :Str) : Integer');
25744:  Func IsEXEx64( const EXEName :Str) :Bool');
25745:  Func Isx64( const FileName :Str) :Bool');
25746:  Func GetRegisteredPythonVersion( SysVersion :Str; out PythonVersion : TPythonVersion) :Bool');
25747:  Func GetRegisteredPythonVersions : TPythonVersions');
25748:  Func GetLatestRegisteredPythonVersion( out PythonVersion : TPythonVersion) :Bool');
25749:  Func PythonVersionFromPath(const Path:str;out PythonVersion:TPythonVersion;AcceptVirtualEnvs:Bool):Bool);
25750: end;
25751:
25752: Example: var pyv: TPythonVersion;  pyvs:TPythonVersions;
25753:    pyvs:= GetRegisteredPythonVersions;
25754:    writeln(objtostr(pyv));
25755:    for it:= 1 to length(pyvs)-1 do begin
25756:     writeln(pyvs[it].dllname)
25757:     writeln(pyvs[it].installpath)
25758:     writeln(pyvs[it].PythonExecutable )
25759:     writeln(pyvs[it].Version )
25760:    end;
25761:
25762: Proc SIRegister_TPythonAction(CL: TPSPascalCompiler);
25763: begin
25764:   //with RegClassS(CL,'TAction', 'TPythonAction') do
25765:   with CL.AddClassN(CL.FindClass('TAction'),'TPythonAction') do begin
25766:     'fRegisteredMethods', 'TList', iptrw);
25767:     'fPythonModule', 'TPythonModule', iptrw);
25768:     'fClearname', 'string', iptrw);
25769:     'fRegistername', 'string', iptrw);
25770:     'fUnregistername', 'string', iptrw);
25771:     Proc ClearMethods');
25772:     Constructor Create( AOwner : TComponent);
25773:     Func HandlesTarget( Target : TObject) :Bool');
25774:     Func Execute :Bool');
25775:     Proc UpdateTarget( Target : TObject);
25776:     Proc InitializeAction');
25777:     RegisterProperty('RegisteredMethods', 'TList', iptr);
25778:     RegisterProperty('PythonModule', 'TPythonModule', iptrw);
25779:   end;
25780: end;
25781:
25782: //V4.7.5.90
25783: procedure SIRegister_TSingleListClass(CL: TPSPascalCompiler);
25784: begin
25785:   TSingleListSortCompare
25786:    = function (Item1, Item2: Single): Integer;
25787:   TSingleDescriptor
25788:    = function (Index:Integer;Item : Single) : string;
25789:
25790:   //with RegClassS(CL,'TObject', 'TSingleListClass') do
25791:   with CL.AddClassN(CL.FindClass('TObject'),'TSingleListClass') do begin
25792:     Constructor Create');
25793:     Procedure Free;
25794:     Function Add( Item : Single) : Integer');
25795:     Procedure Clear');
25796:     Procedure SaveToStream( const S : TStream);
25797:     Procedure LoadFromStream( const S : TStream; const KeepCurrentSortType : Boolean);
25798:     Procedure SaveToFile( FileName : string);
25799:     Procedure LoadFromFile( FileName : string; const KeepCurrentSortType : Boolean);
25800:     Procedure Delete( Index : Integer);
```

```
25801:     Function ErrMsg( const Msg : string; Data : Integer) : string');
25802:     Procedure Exchange( Index1, Index2 : Integer);
25803:     Function Expand : TSingleListClass');
25804:     Function First : Single');
25805:     Function IndexOf( Value : Single) : Integer');
25806:     Procedure Insert( Index : Integer; Item : Single);
25807:     Function Last : Single');
25808:     Procedure Move( CurIndex, NewIndex : Integer);
25809:     Function Remove( Item : Single) : Integer');
25810:     Procedure Pack( NilValue : Single);
25811:     Procedure Sort( Compare : TSingleListSortCompare);
25812:     Procedure SortUp');
25813:     Procedure SortDown');
25814:     Procedure ShowList( StringList: TStrings; Descriptor: TSingleDescriptor; ClearIt:Boolean);
25815:     Function Minimum : Single');
25816:     Function Maximum : Single');
25817:     Function Range : Single');
25818:     Function Sum : Extended');
25819:     Function SumSqr : Extended');
25820:     Function Average : Extended');
25821:     Procedure CopyFrom( List : TSingleListClass; const KeepCurrentSortType : Boolean);
25822:     Procedure CopyTo( List : TSingleListClass; const KeepDestSortType : Boolean);
25823:     Procedure Push( Value : Single);
25824:     Function LifoPop( DefValue : Single) : Single');
25825:     Function FifoPop( DefValue : Single) : Single');
25826:     RegisterProperty('List', 'PSinglePtrList', iptr);
25827:     RegisterProperty('Capacity', 'Integer', iptrw);
25828:     RegisterProperty('Count', 'Integer', iptrw);
25829:     RegisterProperty('Items', 'Single Integer', iptrw);
25830:     SetDefaultPropery('Items');
25831:     RegisterProperty('SortType', 'TSingleSortOption', iptrw);
25832:   end;
25833: end;
25834:
25835: procedure SIRegister_DoubleList4(CL: TPSPascalCompiler);
25836: begin
25837:  CL.AddConstantN('SDoubleListVoidError','String').SetString( 'Invalid method call (empty list)!');
25838:  CL.AddConstantN('SDoubleListSortError','String').SetString( 'Invalid method call (sorted list)!');
25839:    //= array[0..MaxDoubleListSize - 1] of Double;
25840:   //  CL.AddTypeS('TDoublePtrList', 'array[0..2147417850] of Single');
25841:  CL.AddTypeS('TDoublePtrList', 'array of Single');
25842:     { TDoubleListSortCompare
25843:    = function (Item1, Item2: Double): Integer;
25844:     TDoubleDescriptor = function (Index:Integer;Item : Double) : string;  }
25845:      CL.AddTypeS('TDoubleListSortCompare', 'function (Item1,Item2: Double):Integer;
25846:    //TSingleDescriptor //=// function (Index:Integer;Item : Single) : string;
25847:     CL.AddTypeS('TDoubleDescriptor', 'function (Index:Integer;Item : Double) : string;
25848:     CL.AddTypeS('TDoubleSortOption', '( DoubleSortNone, DoubleSortUpWithDup, Doub'
25849:    +'leSortUpNoDup, DoubleSortDownWithDup, DoubleSortDownNoDup );
25850:   SIRegister_TDoubleList(CL);
25851:  CL.AddDelphiFunction('Function DefDescDouble( Index : Integer; Item : Double) : string');
25852: end;
25853:
25854: procedure SIRegister_ByteListClass(CL: TPSPascalCompiler);
25855: begin
25856:  CL.AddConstantN('SByteListVoidError','String').SetString( 'Invalid method call (empty list)!');
25857:  CL.AddConstantN('SByteListSortError','String').SetString( 'Invalid method call (sorted list)!');
25858:   CL.AddTypeS('TByteSortOption', '( ByteSortNone, ByteSortUpWithDup, ByteSortUp'
25859:    +'NoDup, ByteSortDownWithDup, ByteSortDownNoDup );
25860:     CL.AddTypeS('TBytePtrList', 'array of Byte');
25861:     TByteListSortCompare', 'function (Item1, Item2: Byte): Integer;
25862:     TByteDescriptor', 'function (Index:Integer;Item : Byte) : string;
25863:   SIRegister_TByteListClass(CL);
25864:  CL.AddDelphiFunction('Function DefDescByte( Index : Integer; Item : Byte): string');
25865: end;
25866:
25867: procedure SIRegister_neuralplanbuilder(CL: TPSPascalCompiler);
25868: begin
25869:  'NeuralMaxStates2','LongInt').SetInt( 400);
25870:  'NeuralMaxPlans2','LongInt').SetInt( 100);
25871:   CL.AddTypeS('TNeuralState2', 'array of byte');
25872:   CL.AddTypeS('TProcPred2', Function ( var ST : array of byte; Action : byte): boolean');
25873:   SIRegister_TActionStateList(CL);
25874:   Procedure TVisitedStatesCopy( var A, B : TActionStateList);
25875:   SIRegister_TPlan(CL);
25876:   SIRegister_TCompositePlan(CL);
25877: end;
25878:
25879: procedure SIRegister_neuralvolume(CL: TPSPascalCompiler);
25880: begin
25881:  'csMinAvxSize','LongInt').SetInt( 16);
25882:  'csEncodeRGB','LongInt').SetInt( 0);
25883:  'csEncodeHSV','LongInt').SetInt( 1);
25884:  'csEncodeHSL','LongInt').SetInt( 2);
25885:  'csEncodeLAB','LongInt').SetInt( 3);
25886:  'csEncodeGray','LongInt').SetInt( 4);
25887:   CL.AddTypeS('TNeuralFloat', 'Single');
25888:   //CL.AddTypeS('TNeuralFloatPtr', '^TNeuralFloat // will not work');
25889:   //CL.AddTypeS('TNeuralFloatArrPtr', '^TNeuralFloatArr // will not work');
```

```
25890:   CL.AddTypeS('TNeuralIntegerArray', 'array of integer');
25891:   //CL.AddTypeS('T', 'TNeuralFloat');
25892:   CL.AddTypeS('PtrInt', 'Integer');
25893:   SIRegister_TNNetList(CL);
25894:   SIRegister_TVolume(CL);
25895:   SIRegister_TNNetVolume(CL);
25896:   SIRegister_TNNetVolumePair(CL);
25897:   SIRegister_TMObject(CL);
25898:   SIRegister_TNNetVolumeList(CL);
25899:   SIRegister_TNNetVolumePairList(CL);
25900:   SIRegister_TNNetKMeans(CL);
25901:   SIRegister_TNNetStringList(CL);
25902:   SIRegister_TStringListInt(CL);
25903:   SIRegister_TNNetDictionary(CL);
25904:  Function CreateTokenizedStringList86( str : string; c : char) : TStringList;
25905:  Function CreateTokenizedStringList87( c : char) : TStringList;
25906:  Function HiperbolicTangent( x : TNeuralFloat) : TNeuralFloat');
25907:  Function HiperbolicTangentDerivative( x : TNeuralFloat) : TNeuralFloat');
25908:  Function RectifiedLinearUnit( x : TNeuralFloat) : TNeuralFloat');
25909:  Function RectifiedLinearUnitDerivative( x : TNeuralFloat) : TNeuralFloat');
25910:  Function RectifiedLinearUnitLeaky( x : TNeuralFloat) : TNeuralFloat');
25911:  Function RectifiedLinearUnitLeakyDerivative( x : TNeuralFloat) : TNeuralFloat');
25912:  Function ReLULeakyBound( x : TNeuralFloat) : TNeuralFloat');
25913:  Function ReLULeakyBoundDerivative( x : TNeuralFloat) : TNeuralFloat');
25914:  Function Sigmoid( x : TNeuralFloat) : TNeuralFloat');
25915:  Function SigmoidDerivative( x : TNeuralFloat) : TNeuralFloat');
25916:  Function Identity( x : TNeuralFloat) : TNeuralFloat');
25917:  Function IdentityDerivative( x : TNeuralFloat) : TNeuralFloat');
25918:  Function SoftmaxDerivative( x : TNeuralFloat) : TNeuralFloat');
25919:  Function DiffAct( x : TNeuralFloat) : TNeuralFloat');
25920:  Function DiffActDerivative( x : TNeuralFloat) : TNeuralFloat');
25921:  Function NeuronForceMinMax88( x, pMin, pMax : TNeuralFloat) : TNeuralFloat;
25922:  Function NeuronForceMinMax89( x, pMin, pMax : integer) : integer;
25923:  Function NeuronForceRange( x, range : TNeuralFloat) : TNeuralFloat');
25924:  Function NeuronForceMinRange( x, range : TNeuralFloat) : TNeuralFloat');
25925:  Procedure rgb2hsv( r, g, b : TNeuralFloat; var h, s, v : TNeuralFloat);
25926:  Procedure hsv2rgb( h, s, v : TNeuralFloat; var r, g, b : TNeuralFloat);
25927:  Function hue2rgb( p, q, t : TNeuralFloat) : TNeuralFloat');
25928:  Procedure rgb2hsl( r, g, b : TNeuralFloat; var h, s, l : TNeuralFloat);
25929:  Procedure hsl2rgb( h, s, l : TNeuralFloat; var r, g, b : TNeuralFloat);
25930:  Procedure lab2rgb( l, a, b : TNeuralFloat; var r, g, bb : TNeuralFloat);
25931:  Procedure rgb2lab( r, g, b : TNeuralFloat; var l, a, bb : TNeuralFloat);
25932:  Function RoundAsByte( x : TNeuralFloat) : byte');
25933:  Function CompareStringListIntegerAsc( List : TStringList; Index1, Index2 : Integer) : Integer');
25934:  Function CompareStringListIntegerDesc( List : TStringList; Index1, Index2 : Integer) : Integer');
25935:  Function CompareNNetVolumeListAsc( const Item1, Item2 : TNNetVolume) : Integer');
25936:  Function CompareNNetVolumeListDesc( const Item1, Item2 : TNNetVolume) : Integer');
25937:  Function NeuralFloatToStr( V : TNeuralFloat) : string');
25938:  Function NeuralStrToFloat( V : String) : TNeuralFloat');
25939:  Procedure TestTNNetVolume( );    //messagedialog!
25940:  Procedure TestKMeans2( );         //unattendend
25941: end;
25942:
25943:  (*----------------------------------------------------------------------------*)
25944: procedure SIRegister_neuralvolumev(CL: TPSPascalCompiler);
25945: begin
25946:  Procedure SaveHandleToBitmap( OutputFileName : string; hWnd : HWND);
25947:  Procedure LoadVolumeIntoTImage( V : TNNetVolume; Image : TImage; color_encoding : integer);
25948:  Procedure LoadRGBVolumeIntoTImage( V : TNNetVolume; Image : TImage);
25949:  Procedure LoadPictureIntoVolume( LocalPicture : TPicture; Vol : TNNetVolume);
25950:  Procedure LoadBitmapIntoVolume( LocalBitmap : TBitmap; Vol : TNNetVolume);
25951:  Procedure LoadImageFromFileIntoVolume( ImageFileName : string; V : TNNetVolume);
25952: end;
25953:
25954: function GetSmallestIdxInRange(StartPos, Len: integer): integer;
25955: function GetValueCount(Value: T): integer;
25956: GetParentProcessName2
25957: writeln('GetParentProcessName2: '+GetProcessName(GetParentProcessID(getprocessID)));
25958: Function DefDescByte( Index : Integer; Item : Byte) : string');
25959: function NumStringParts(SourceStr,Delimiter:String):Integer');
25960: function GetStringPart(SourceStr,Delimiter:String;Num:Integer):string');
25961: Function InverseCompareResult( const C : TCompareResult) : TCompareResult');
25962:
25963: function GetWindowProcessID(const Wnd: Windows.HWND): Windows.DWORD;
25964: var
25965:   GetWindowThreadProcessId: function(Wnd: Windows.HWND; // API fn prototype
25966:     lpdwProcessId: Windows.PDWORD): Windows.DWORD; stdcall;
25967: begin
25968:   // We load the API function explicitly to make the routine compatible with as
25969:   // many compilers as possible: the definition of GetWindowThreadProcessId in
25970:   // Delphi and FreePascal Windows units varies across compilers
25971:   GetWindowThreadProcessId := GetProcAddress(
25972:     GetModuleHandle('user32.dll'), 'GetWindowThreadProcessId' );
25973:   if Assigned(GetWindowThreadProcessId) and Windows.IsWindow(Wnd) then
25974:     GetWindowThreadProcessId(Wnd, @Result)
25975:   else
25976:     Result := Windows.DWORD(-1);
25977: end;
25978:
```

```
25979:
25980: Procedure TestVectorClassExtended  - type  TInt64Array = TInt64ArrayClass;
25981:
25982: procedure SIRegister_TVectorClass(CL: TPSPascalCompiler);
25983: begin
25984:   //with RegClassS(CL,'TVectorBaseArray', 'TVectorClass') do
25985:   //with CL.AddClassN(CL.FindClass('TVectorBaseArray'),'TVectorClass') do
25986:   with CL.AddClassN(CL.FindClass('TExtendedArray'),'TVectorClass') do begin
25987:     Function CreateInstance : AType');
25988:     Procedure Add( const V : MFloat);
25989:     Procedure Add1( const V : PMFloat; const Count : Integer);
25990:     Procedure Add2( const V : PInt64; const Count : Integer);
25991:     Procedure Add3( const V : MFloatArray);
25992:     Procedure Add4( const V : Int64Array);
25993:     Procedure Add5( const V : TVectorBaseArray);
25994:     Procedure Add6( const V : TInt64Array);
25995:     Procedure Add7( const V : TObject);
25996:     Procedure Subtract8( const V : MFloat);
25997:     Procedure Subtract9( const V : PMFloat; const Count : Integer);
25998:     Procedure Subtract10( const V : PInt64; const Count : Integer);
25999:     Procedure Subtract11( const V : MFloatArray);
26000:     Procedure Subtract12( const V : Int64Array);
26001:     Procedure Subtract13( const V : TVectorBaseArray);
26002:     Procedure Subtract14( const V : TInt64Array);
26003:     Procedure Subtract15( const V : TObject);
26004:     Procedure Multiply16( const V : MFloat);
26005:     Procedure Multiply17( const V : PMFloat; const Count : Integer);
26006:     Procedure Multiply18( const V : PInt64; const Count : Integer);
26007:     Procedure Multiply19( const V : MFloatArray);
26008:     Procedure Multiply20( const V : Int64Array);
26009:     Procedure Multiply21( const V : TVectorBaseArray);
26010:     Procedure Multiply22( const V : TInt64Array);
26011:     Procedure Multiply23( const V : TObject);
26012:     Function DotProduct24( const V : PMFloat; const Count : Integer): MFloat;
26013:     Function DotProduct25( const V : PInt64; const Count : Integer) : MFloat;
26014:     Function DotProduct26( const V : MFloatArray) : MFloat;
26015:     Function DotProduct27( const V : Int64Array) : MFloat;
26016:     Function DotProduct28( const V : TVectorBaseArray) : MFloat;
26017:     Function DotProduct29( const V : TInt64Array) : MFloat;
26018:     Function DotProduct30( const V : TObject) : MFloat;
26019:     Function Norm : MFloat');
26020:     Function Min : MFloat');
26021:     Function Max : MFloat');
26022:     Function Range( var Min, Max : MFloat) : MFloat');
26023:     Function IsZero( const CompareDelta : MFloat) : Boolean');
26024:     Function HasZero( const CompareDelta : MFloat) : Boolean');
26025:     Function HasNegative : Boolean');
26026:     Procedure Normalize');
26027:     Procedure Negate');
26028:     Procedure ValuesInvert');
26029:     Procedure ValuesSqr');
26030:     Procedure ValuesSqrt');
26031:     Function Sum : MFloat');
26032:     Function SumOfSquares : MFloat');
26033:     Procedure SumAndSquares( out Sum, SumOfSquares : MFloat);
26034:     Procedure SumAndCubes( out Sum, SumOfSquares, SumOfCubes : MFloat);
26035:     Procedure SumAndQuads( out Sum, SumOfSquares, SumOfCubes, SumOfQuads : MFloat);
26036:     Function WeightedSum( const Weights : TVector) : MFloat');
26037:     Function Mean : MFloat');
26038:     Function HarmonicMean : MFloat');
26039:     Function GeometricMean : MFloat');
26040:     Function Median : MFloat');
26041:     Function Mode : MFloat');
26042:     Function Variance : MFloat');
26043:     Function StdDev( var Mean : MFloat) : MFloat');
26044:     Function PopulationVariance : MFloat');
26045:     Function PopulationStdDev : MFloat');
26046:     Function M1 : MFloat');
26047:     Function M2 : MFloat');
26048:     Function M3 : MFloat');
26049:     Function M4 : MFloat');
26050:     Function Skew : MFloat');
26051:     Function Kurtosis : MFloat');
26052:     Function Product : MFloat');
26053:     Function Angle( const V : TVector) : MFloat');
26054:   end;
26055: end;
26056:
26057: (*----------------------------------------------------------------------------*)
26058: procedure SIRegister_TInt64ArrayClass(CL: TPSPascalCompiler);  BigDataClass
26059: begin
26060:   //with RegClassS(CL,'AInt64Array', 'TInt64ArrayClass') do
26061:   with CL.AddClassN(CL.FindClass('AInt64Array'),'TInt64ArrayClass') do begin
26062:     Constructor Create( const V : Int64Array);
26063:     Procedure ExchangeItems( const Idx1, Idx2 : Integer);
26064:     Function DuplicateRange( const LoIdx, HiIdx : Integer) : AArray');
26065:     Procedure Delete( const Idx : Integer; const ACount : Integer);
26066:     Procedure Insert( const Idx : Integer; const ACount : Integer);
26067:     Procedure Assign1( const V : Int64Array);
```

```
26068:     Procedure Assign2( const V : array of Int64);
26069:     Function AppendItem( const Value : Int64) : Integer');
26070:     ('Data', 'Int64Array', iptrw);
26071:     RegisterProperty('Count', 'Integer', iptrw);
26072:   end;
26073: end;
26074:
26075: procedure SIRegister_TExtendedArray(CL: TPSPascalCompiler);
26076: begin
26077:   //with RegClassS(CL,'AExtendedArray', 'TExtendedArray') do
26078:   with CL.AddClassN(CL.FindClass('AExtendedArray'),'TExtendedArray') do begin
26079:     Constructor Create( const V : ExtendedArray);
26080:     //RegisterMethod('Procedure Free;
26081:     Procedure Assign( const Source : TObject);
26082:     Procedure ExchangeItems( const Idx1, Idx2 : Integer);
26083:     Function DuplicateRange( const LoIdx, HiIdx : Integer) : AArray');
26084:     Procedure Delete( const Idx : Integer; const ACount : Integer);
26085:     Procedure Insert( const Idx : Integer; const ACount : Integer);
26086:     Procedure Assign2( const V : ExtendedArray);
26087:     //RegisterMethod('Procedure Assign( const V : array of Extended);
26088:     Function AppendItem( const Value : Extended) : Integer');
26089:     RegisterProperty('Data', 'ExtendedArray', iptrw);
26090:     RegisterProperty('Count', 'Integer', iptrw);
26091:   end;
26092: end;
26093:
26094: (*----------------------------------------------------------------------------*)
26095: procedure SIRegister_neuraldatasets(CL: TPSPascalCompiler);
26096: begin
26097:   TTinyImageChannel','array [0..31] of array[0..31] of byte; ');
26098:   TTinyImageChannel1D','array [0..32 * 32 - 1] of byte; ');
26099:   TMNistImage','array [0..27] of array[0..27] of byte; ');
26100:   //TTinyImageChannel1D = packed array [0..32 * 32 - 1] of byte;
26101:   //TMNistImage = packed array [0..27, 0..27] of byte;
26102:   TTinyImage', 'record bLabel : byte; R : TTinyImageChannel; G : T'
26103:    +'TinyImageChannel; B : TTinyImageChannel; end');
26104:   TCifar100Image', 'record bCoarseLabel : byte; bFineLabel : byte;'
26105:    +' R : TTinyImageChannel; G : TTinyImageChannel; B : TTinyImageChannel; end');
26106:   TTinySingleChannelImage', 'record bLabel : byte; Grey : TTinyImageChannel; end');
26107:   TTinySingleChannelImage1D', 'record bLabel : byte; Grey : TTinyImageChannel1D; end');
26108:   //CL.AddTypeS('TTinySingleChannelImagePtr', '^TTinySingleChannelImage // will not work');
26109:   //CL.AddTypeS('TTinySingleChannelImage1DPtr', '^TTinySingleChannelImage1D // will not work');
26110:   SIRegister_TFileNameList(CL);
26111:   SIRegister_TClassesAndElements(CL);
26112:  Procedure CreateVolumesFromImagesFromFolder(out ImgTrainingVolumes,ImgValidationVolumes,
       ImgTestVolumes:TNNetVolumeList;FolderName,pImageSubFolder:string;color_encoding:integer;TrainingProp,
       ValidationProp,TestProp:single;'+'NewSizeX:integer;NewSizeY:integer);
26113:  Procedure CreateFileNameListsFromImagesFromFolder( out TrainingFileNames, ValidationFileNames,
       TestFileNames: TFileNameList; FolderName, pImageSubFolder : string; TrainingProp, ValidationProp, TestProp
       : single);
26114:  //CL.AddDelphiFunction('Procedure LoadImageIntoVolume( M : TFPMemoryImage; Vol : TNNetVolume);
26115:  //CL.AddDelphiFunction('Procedure LoadVolumeIntoImage( Vol : TNNetVolume; M : TFPMemoryImage);
26116:  Function LoadImageFromFileIntoVolume( ImageFileName : string; V : TNNetVolume) : boolean');
26117:  Function SaveImageFromVolumeIntoFile( V : TNNetVolume; ImageFileName : string) : boolean');
26118:  Procedure ConfusionWriteCSVHeader( var CSVConfusion : TextFile; Labels : array of string);
26119:  Procedure ConfusionWriteCSV( var CSVConfusion : TextFile; Vol : TNNetVolume; Digits : integer);
26120:  Procedure LoadTinyImageIntoNNetVolume1( var TI : TTinyImage; Vol : TNNetVolume);
26121:  Procedure LoadTinyImageIntoNNetVolume2( var TI : TCifar100Image; Vol : TNNetVolume);
26122:  Procedure LoadTinyImageIntoNNetVolume3( var TI : TMNistImage; Vol : TNNetVolume);
26123:  Procedure LoadNNetVolumeIntoTinyImage4( Vol : TNNetVolume; var TI : TTinyImage);
26124:  Procedure LoadNNetVolumeIntoTinyImage5( Vol : TNNetVolume; var TI : TCifar100Image);
26125:  Procedure LoadTinySingleChannelIntoNNetVolume( var SC : TTinySingleChannelImage; Vol : TNNetVolume);
26126:  Procedure TinyImageCreateGrey( var TI : TTinyImage; var TIGrey : TTinySingleChannelImage);
26127:  Procedure TinyImageHE( var TI, TIHE : TTinySingleChannelImage);
26128:  Procedure TinyImageVE( var TI, TIVE : TTinySingleChannelImage);
26129:  Procedure TinyImageRemoveZeroGradient( var TI : TTinySingleChannelImage; distance : byte);
26130:  Procedure TinyImageHVE( var TI, TIHE : TTinySingleChannelImage);
26131:  Function TinyImageTo1D( var TI : TTinySingleChannelImage) : TTinySingleChannelImage1D');
26132:  Procedure CreateCifar10Volumes(out ImgTrainingVolumes,ImgValidationVolumes,
       ImgTestVolumes:TNNetVolumeList;color_encoding:byte);
26133:  Procedure CreateCifar100Volumes(out ImgTrainingVolumes,ImgValidationVolumes,
       ImgTestVolumes:TNNetVolumeList;color_encoding:byte;Verbose :bool;
26134:  Procedure CreateMNISTVolumes( out ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes :
       TNNetVolumeList; TrainFileName, TestFileName : string; Verbose : boolean; IsFashion : boolean);
26135:  Procedure loadCifar10Dataset6(ImgVolumes:TNNetVolumeList;idx:integer;base_pos:
       integer;color_encoding:byte););
26136:  Procedure
       loadCifar10Dataset7(ImgVolumes:TNNetVolumeList;fileName:string;base_pos:integer;color_encoding:byte);
26137:  Procedure loadCifar100Dataset(ImgVolumes TNNetVolumeList;fileName: string;
       color_encoding:byte;Verbose:bool);
26138:  Procedure
       loadMNISTDataset(ImgVolumes:TNNetVolumeList;fileName:str;Verbose:bool;IsFashion:bool;MaxLabel:int);
26139:  Function CheckCIFARFile( ) : boolean');
26140:  Function CheckCIFAR100File( ) : boolean');
26141:  Function CheckMNISTFile( fileName : string; IsFasion : boolean) : boolean');
26142:  Procedure TestBatch(NN TNNet;ImgVolumes:TNNetVolumeList;SampleSize:integer;out Rate,Loss,
       ErrorSum:TNeuralFloat)
26143:  Procedure TranslateCifar10VolumesToMachineAnimal( VolumeList : TNNetVolumeList);
26144:  Function SwapEndian( I : integer) : integer');
```

```
26145: end;
26146:
26147: (*----------------------------------------------------------------------------*)
26148: procedure SIRegister_neuralnetworkCAI(CL: TPSPascalCompiler);
26149: begin
26150:  CL.AddConstantN('csMaxInterleavedSize','integer').SetInt( 95);
26151:  CL.AddTypeS('TNeuralFloatDynArr', 'array of TNeuralFloat');
26152:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNet');
26153:   SIRegister_TNNetNeuron(CL);
26154:   SIRegister_TNNetNeuronList(CL);
26155:  CL.AddConstantN('csNNetMaxParameterIdx','LongInt').SetInt( 7);
26156:   SIRegister_TNNetLayer(CL);
26157:   //CL.AddTypeS('TNNetLayerClass', 'class of TNNetLayer');
26158:   SIRegister_TNNetLayerConcatedWeights(CL);
26159:   SIRegister_TNNetLayerList(CL);
26160:   SIRegister_TNNetInputBase(CL);
26161:   SIRegister_TNNetInput(CL);
26162:   SIRegister_TNNetIdentity(CL);
26163:   SIRegister_TNNetPad(CL);
26164:   SIRegister_TNNetIdentityWithoutL2(CL);
26165:   SIRegister_TNNetIdentityWithoutBackprop(CL);
26166:   //CL.AddTypeS('TNNetActivationFunctionClass', 'class of TNNetIdentity');
26167:   SIRegister_TNNetReLUBase(CL);
26168:   SIRegister_TNNetDigital(CL);
26169:   SIRegister_TNNetReLU(CL);
26170:   SIRegister_TNNetReLUL(CL);
26171:   SIRegister_TNNetSELU(CL);
26172:   SIRegister_TNNetSwish(CL);
26173:   SIRegister_TNNetReLUSqrt(CL);
26174:   SIRegister_TNNetPower(CL);
26175:   SIRegister_TNNetLeakyReLU(CL);
26176:   SIRegister_TNNetVeryLeakyReLU(CL);
26177:   SIRegister_TNNetSigmoid(CL);
26178:   SIRegister_TNNetHyperbolicTangent(CL);
26179:   SIRegister_TNNetMulLearning(CL);
26180:   SIRegister_TNNetMulByConstant(CL);
26181:   SIRegister_TNNetNegate(CL);
26182:   SIRegister_TNNetAddAndDiv(CL);
26183:   SIRegister_TNNetAddNoiseBase(CL);
26184:   SIRegister_TNNetDropout(CL);
26185:   SIRegister_TNNetRandomMulAdd(CL);
26186:   SIRegister_TNNetChannelRandomMulAdd(CL);
26187:   SIRegister_TNNetLayerMaxNormalization(CL);
26188:   SIRegister_TNNetLayerStdNormalization(CL);
26189:   SIRegister_TNNetMovingStdNormalization(CL);
26190:   SIRegister_TNNetChannelTransformBase(CL);
26191:   SIRegister_TNNetChannelShiftBase(CL);
26192:   SIRegister_TNNetChannelBias(CL);
26193:   SIRegister_TNNetChannelMul(CL);
26194:   SIRegister_TNNetChannelMulByLayer(CL);
26195:   SIRegister_TNNetCellMulByCell(CL);
26196:   SIRegister_TNNetCellBias(CL);
26197:   SIRegister_TNNetCellMul(CL);
26198:   SIRegister_TNNetChannelZeroCenter(CL);
26199:   SIRegister_TNNetChannelStdNormalization(CL);
26200:   SIRegister_TNNetLocalResponseNorm2D(CL);
26201:   SIRegister_TNNetInterleaveChannels(CL);
26202:   SIRegister_TNNetLocalResponseNormDepth(CL);
26203:   SIRegister_TNNetReshape(CL);
26204:   SIRegister_TNNetConcatBase(CL);
26205:   SIRegister_TNNetConcat(CL);
26206:   SIRegister_TNNetDeepConcat(CL);
26207:   SIRegister_TNNetSum(CL);
26208:   SIRegister_TNNetSplitChannels(CL);
26209:   SIRegister_TNNetSplitChannelEvery(CL);
26210:   SIRegister_TNNetFullConnect(CL);
26211:   //CL.AddTypeS('TNNetFullConnectClass', 'class of TNNetFullConnect');
26212:   SIRegister_TNNetFullConnectLinear(CL);
26213:   SIRegister_TNNetFullConnectSigmoid(CL);
26214:   SIRegister_TNNetFullConnectReLU(CL);
26215:   SIRegister_TNNetFullConnectDiff(CL);
26216:   SIRegister_TNNetSoftMax(CL);
26217:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetLayerFullConnect');
26218:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetLayerFullConnectReLU');
26219:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetLayerSoftMax');
26220:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetDense');
26221:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetDenseReLU');
26222:   SIRegister_TNNetConvolutionAbstract(CL);
26223:   SIRegister_TNNetDepthwiseConv(CL);
26224:   SIRegister_TNNetDepthwiseConvLinear(CL);
26225:   SIRegister_TNNetDepthwiseConvReLU(CL);
26226:   SIRegister_TNNetConvolutionBase(CL);
26227:   //CL.AddTypeS('TNNetConvolutionClass', 'class of TNNetConvolutionBase');
26228:   SIRegister_TNNetGroupedConvolutionLinear(CL);
26229:   SIRegister_TNNetGroupedConvolutionReLU(CL);
26230:   SIRegister_TNNetGroupedPointwiseConvLinear(CL);
26231:   //CL.AddTypeS('TNNetGroupedPointwiseConvClass', 'class of TNNetGroupedPointwiseConvLinear');
26232:   SIRegister_TNNetGroupedPointwiseConvReLU(CL);
26233:   SIRegister_TNNetConvolution(CL);
```

```
26234:   SIRegister_TNNetConvolutionSharedWeights(CL);
26235:   SIRegister_TNNetConvolutionLinear(CL);
26236:   SIRegister_TNNetConvolutionReLU(CL);
26237:   SIRegister_TNNetPointwiseConv(CL);
26238:   SIRegister_TNNetPointwiseConvLinear(CL);
26239:   SIRegister_TNNetPointwiseConvReLU(CL);
26240:   SIRegister_TNNetDeconvolution(CL);
26241:   SIRegister_TNNetDeconvolutionReLU(CL);
26242:   SIRegister_TNNetLocalConnect(CL);
26243:   SIRegister_TNNetLocalProduct(CL);
26244:   SIRegister_TNNetDeLocalConnect(CL);
26245:   SIRegister_TNNetLocalConnectReLU(CL);
26246:   SIRegister_TNNetDeLocalConnectReLU(CL);
26247:   SIRegister_TNNetPoolBase(CL);
26248:   SIRegister_TNNetMaxPool(CL);
26249:   SIRegister_TNNetMaxPoolPortable(CL);
26250:   SIRegister_TNNetMinPool(CL);
26251:   SIRegister_TNNetMaxChannel(CL);
26252:   SIRegister_TNNetMinChannel(CL);
26253:   SIRegister_TNNetAvgPool(CL);
26254:   SIRegister_TNNetAvgChannel(CL);
26255:   SIRegister_TNNetDeMaxPool(CL);
26256:   SIRegister_TNNetUpsample(CL);
26257:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetDeAvgPool');
26258:   SIRegister_TNNet(CL);
26259:   SIRegister_THistoricalNets(CL);
26260:   SIRegister_TNNetDataParallelism(CL);
26261:   SIRegister_TNNetByteProcessing(CL);
26262:   SIRegister_TNNetForByteProcessing(CL);
26263:   SIRegister_TBytePredictionViaNNet(CL);
26264:   SIRegister_TEasyBytePredictionViaNNet(CL);
26265:  Procedure CompareComputing( NN1, NN2 : TNNet);
26266:  Procedure CompareNNStructure( NN, NN2 : TNNet);
26267:  Procedure TestConvolutionAPI( );
26268:  Procedure TestDataParallelism( NN : TNNet);
26269:  //CL.AddDelphiFunction('Procedure
       TestConvolutionOpenCL(platform_id:cl_platform_id;device_id:cl_device_id);
26270:  //CL.AddDelphiFunction('Procedure
       TestFullConnectOpenCL(platform_id:cl_platform_id;device_id:cl_device_id);
26271:  Procedure RebuildPatternOnPreviousPatterns( Calculated : TNNetVolume; LocalWeight : TNNetVolume;
       PrevLayer : TNNetNeuronList; PrevStride : integer; ReLU : boolean; Threshold : TNeuralFloat);
26272:  Procedure RebuildNeuronListOnPreviousPatterns( CalculatedLayer : TNNetNeuronList; CurrentLayer, PrevLayer
       : TNNetNeuronList; PrevStride:integer;ReLU:boolean;Threshold:TNeuralFloat); InputLayer :=
26273:  function NN.AddDenseNetBlockCAI70(InnerConvNum,iConvNeuronCount,0, netconv,ChkUseSeparableConv.Checked,
       ChkMovingNorm.Checked,nil,nil, 0,1,1.0,1,1,1);
26274: end;
26275:
26276: (*----------------------------------------------------------------------------*)
26277: procedure SIRegister_neuralfit(CL: TPSPascalCompiler);
26278: begin
26279:   TCustomLearningRateScheduleObjFn', 'Function ( Epoch : integer) : single');
26280:   TMultiThreadProcItem', 'TObject');
26281:   SIRegister_TNeuralFitBase(CL);
26282:   SIRegister_TNeuralFitWithImageBase(CL);
26283:   TNNetDataAugmentationFn', 'Procedure ( pInput : TNNetVolume; ThreadId : integer);
26284:   TNNetLossFn', 'Function ( ExpectedOutput, FoundOutput : TNNetVolume; ThreadId : integer) :
       TNeuralFloat');
26285:   TNNetGetPairFn', 'Function ( Idx : integer; ThreadId : integer) : TNNetVolumePair');
26286:   TNNetGet2VolumesProc', 'Procedure ( Idx : integer; ThreadId : integer; pInput, pOutput : TNNetVolume);
26287:   SIRegister_TNeuralDataLoadingFit(CL);
26288:   SIRegister_TNeuralFit(CL);
26289:   SIRegister_TNeuralImageLoadingFit(CL);
26290:   SIRegister_TNeuralImageFit(CL);
26291:  Function MonopolarCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26292:  Function BipolarCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26293:  Function BipolarCompare99( A, B : TNNetVolume; ThreadId : integer) : boolean');
26294:  Function ClassCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26295: end;
26296: V 47620
26297:   FFit.FitLoading2(FDiscriminator, 64*10, 500, 500, 64, 35000,
26298:                @GetDiscriminatorTrainingPair, nil, nil); // This
26299:   FFit.FitLoading22(FDiscriminator, 64*10, 500, 500, 64, 35000,
26300:                @GetDiscriminatorTrainingProc, nil, nil);  //}
26301:
26302: procedure SIRegister_neuralgeneric(CL: TPSPascalCompiler);
26303: begin
26304:   SIRegister_TIncDec(CL);
26305:   SIRegister_TRandom(CL);
26306:  Function MaxSingle2( x, y : single) : single');
26307:  Function GetMaxDivisor( x, acceptableMax : integer) : integer');
26308:  Function GetMaxAcceptableCommonDivisor( a, b : integer; max_acceptable : integer) : integer');
26309: end;
26310:
26311:  (*----------------------------------------------------------------------------*)
26312: procedure SIRegister_neuraldatasetsv(CL: TPSPascalCompiler);
26313: begin
26314:   TImageDynArr', 'array of TImage');
26315:   TLabelDynArr', 'array of TLabel');
26316:  Procedure LoadTinyImageIntoTImage( var TI : TTinyImage; var Image : TImage);
```

```
26317:  Procedure LoadTISingleChannelIntoImage( var TI : TTinySingleChannelImage; var Image : TImage);
26318:  Procedure ShowNeurons(pNeuronList:TNNetNeuronList;var pImage:TImageDynArr;startImage,filterSize,
        color_encoding:int;ScalePerImage:bool;
26319:  Procedure CreateAscentImages(GrBoxNeurons: TGroupBox; var pImage: TImageDynArr; var pLabelX, pLabelY:
        TLabelDynArr; ImagesNum : integer; inputSize, displaySize, imagesPerRow : integer);
26320:  Procedure CreateNeuronImages(GrBoxNeurons: TGroupBox; var pImage: TImageDynArr; var pLabelX, pLabelY:
        TLabelDynArr; pNeuronList : TNNetNeuronList; filterSize, imagesPerRow, NeuronNum : integer);
26321:  Procedure FreeNeuronImages( var pImage : TImageDynArr; var pLabelX, pLabelY : TLabelDynArr);
26322:  Procedure LoadNNLayersIntoCombo( NN : TNNet; Combo : TComboBox);
26323: end;
26324:
26325: (*----------------------------------------------------------------------------*)
26326: procedure SIRegister_flcFloats(CL: TPSPascalCompiler);
26327: begin
26328:  Function flcDoubleMin( const A, B : Double) : Double');
26329:  Function flcDoubleMax( const A, B : Double) : Double');
26330:  Function flcExtendedMin( const A, B : Extended) : Extended');
26331:  Function flcExtendedMax( const A, B : Extended) : Extended');
26332:  Function flcFloatMin( const A, B : Float) : Float');
26333:  Function flcFloatMax( const A, B : Float) : Float');
26334:  Function flcFloatClip( const Value : Float; const Low, High : Float) : Float');
26335:  Function flcInSingleRange( const A : Float) : Boolean');
26336:  Function flcInDoubleRange( const A : Float) : Boolean');
26337:  Function flcInCurrencyRange0( const A : Float) : Boolean;
26338:  Function flcInCurrencyRange1( const A : Int64) : Boolean;
26339:  Function flcExtendedExponentBase2( const A : Extended; var Exponent : Integer) : Boolean');
26340:  Function flcExtendedExponentBase10( const A : Extended; var Exponent : Integer): Boolean');
26341:  Function flcExtendedIsInfinity( const A : Extended) : Boolean');
26342:  Function flcExtendedIsNaN( const A : Extended) : Boolean');
26343:  CL.AddConstantN('flcSingleCompareDelta','Extended').setExtended( 1.0E-34);
26344:  flcDoubleCompareDelta','Extended').setExtended( 1.0E-280);
26345:  ExtendedCompareDelta','').SetString( DoubleCompareDelta);
26346:  flcExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
26347:  DefaultCompareDelta','').SetString( SingleCompareDelta);
26348:  Function flcFloatZero( const A : Float; const CompareDelta : Float) : Boolean');
26349:  Function flcFloatOne( const A : Float; const CompareDelta : Float) : Boolean');
26350:  Function flcFloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean');
26351:  Function flcFloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult');
26352:  flcSingleCompareEpsilon','Extended).setExtended( 1.0E-5);
26353:  flcDoubleCompareEpsilon','Extended).setExtended( 1.0E-13);
26354:  flcExtendedCompareEpsilon','Extended).setExtended( 1.0E-17);
26355:  flcDefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
26356:  Function flcExtendedApproxEqual( const A, B : Extended; const CompareEpsilon : Double): Boolean');
26357:  Function flcExtendedApproxCompare( const A, B : Extended; const CompareEpsilon Double): TCompareResult');
26358:  Function flcDoubleApproxEqual( const A, B : Double; const CompareEpsilon : Double): Boolean');
26359:  Function flcDoubleApproxCompare(const A,B: Double; const CompareEpsilon: Double): TCompareResult');
26360:  Function flcFloatApproxEqual( const A, B : Float; const CompareEpsilon : Float): Boolean');
26361:  Function flcFloatApproxCompare( const A, B : Float; const CompareEpsilon : Float): TCompareResult');
26362:  Function flcFloatToStringA(const A : Float) : AnsiString');
26363:  Function flcFloatToStringB(const A : Float) : RawByteString');
26364:  Function flcFloatToStringU(const A : Float) : UnicodeString');
26365:  Function flcFloatToString(const A : Float) : String');
26366:  Function flcTryStringToFloatB(const A : RawByteString; out B : Float) : Boolean');
26367:  Function flcTryStringToFloatU(const A : UnicodeString; out B : Float) : Boolean');
26368:  Function flcTryStringToFloat(const A : String; out B : Float) : Boolean');
26369:  Function flcStringToFloatB(const A : RawByteString) : Float');
26370:  Function flcStringToFloatU(const A : UnicodeString) : Float');
26371:  Function flcStringToFloat(const A : String) : Float');
26372:  Function flcStringToFloatDefB(const A : RawByteString; const Default : Float): Float');
26373:  Function flcStringToFloatDefU(const A : UnicodeString; const Default : Float): Float');
26374:  Function flcStringToFloatDef(const A : String; const Default : Float) : Float');
26375:  Procedure FLCFloatStringTest');
26376:  Procedure FLCFloatTest');
26377: end;
26378:
26379: procedure SIRegister_neuralthread(CL: TPSPascalCompiler);
26380: begin
26381:   CL.AddTypeS('TNeuralProc', 'Procedure ( index, threadnum : integer);
26382:   SIRegister_TNeuralThread(CL);
26383:   SIRegister_TNeuralThreadList(CL);
26384:  Procedure NeuralThreadListCreate( pSize : integer);
26385:  Procedure NeuralThreadListFree( );
26386:  Function fNTL : TNeuralThreadList');
26387:  Procedure CreateNeuralThreadListIfRequired( );
26388:  Function NeuralDefaultThreadCount : integer');
26389:  Procedure NeuralInitCriticalSection( var pCritSec : TRTLCriticalSection);
26390:  Procedure NeuralDoneCriticalSection( var pCritSec : TRTLCriticalSection);
26391:  Procedure InitCriticalSection( var pCritSec : TRTLCriticalSection);
26392:  Procedure DoneCriticalSection( var pCritSec : TRTLCriticalSection);
26393:  Function neuralGetProcessId( ) : integer');
26394: end;
26395:
26396: procedure SIRegister_uSysTools(CL: TPSPascalCompiler);
26397: begin
26398:  Function
        usGetRegExFileList(APath:String;APattern:String;ASubDirs:Boolean;AResultList:TStrings;AAttributes:Integer;AWithPath:Bo
26399:  Procedure usFindFileInPaths(const AFileName:String;const ASearchPaths:String;AMinimumFileSize:Int64;const
        AResultList:TStrings);
26400:  Procedure usPrintWindow( Wnd : HWND; ATo : TBitmap);
```

```
26401:  Function usGetWindowByClassTree( const AClassTree: TStringList; AParent:HWND;
       AWithWildCards:Boolean):HWND');
26402:  Function usGetTempFile( AExtension : String) : String');
26403:  Function usDeleteFileToWasteBin( AFileName : string) : boolean');
26404:  Function usExpandEnvVars( AInputString : String) : String');
26405:  Function usFileSize( AFileName : String) : Int64');
26406:  Function usIsMultiprocessor : Boolean');
26407:  Function usProcessorCount : Cardinal');
26408:  Function usIsAdmin: Boolean');
26409:  Function usMultipleStringReplace(AString:Str;AOldPatterns,ANewPatterns:array of
       Str;AFlags:TReplaceFlags):Str;
26410:  Function usMakeFileName( ADesiredFName : String) : String');
26411:  Function usIsLike( AString, APattern : String) : Boolean');
26412:  Function usVarRecToVariant( AValue : TVarRec) : Variant');
26413:  Function usVariantToTypedVarRec( const Item : Variant; VarType : TVarType) : TVarRec');
26414:  Function usVariantToVarRec( const Item : Variant) : TVarRec');
26415:  Procedure usFinalizeVarRec( var Item : TVarRec);
26416:  Function usGetWindowText( wnd : HWND) : WideString');
26417:  Procedure usSetWindowText( wnd : HWND; txt : WideString);
26418:  Function usMultipleWideStringReplace(AString:WideString;AOldPatterns,ANewPatterns:array of WideString;
26419:                                        AFlags: TReplaceFlags):WideString');
26420:  Procedure usAcceptNumericOnly( var Key : Char; Comma : Boolean);
26421:  Function usSameGUID( AGUID1, AGUID2 : TGUID) : Boolean');
26422:  Function usIsBitSet( AValue : Integer; ABitIndex : Byte) : Boolean');
26423:  Function usClientToScreen( AWindow : HWND; var APoint : TPoint) : Boolean');
26424:  Function usGetDataObjectFromFileList( const Directory : string; Files : TStrings) : IDataObject');
26425:  Function usGetDataObjectFromFile( const Directory : string; AFile : String) : IDataObject');
26426:  Function usGetFileListFromDataObject( const DataObject : IDataObject; Files : TStrings) : Integer');
26427:  Function usKeyPressed( AKey : Smallint) : Boolean;
26428:  Function usKeyPressed1( AKey : Byte) : Boolean;
26429:  Function usKeyToogled( AKey : Smallint) : Boolean;
26430:  Function usKeyToogled1( AKey : Byte) : Boolean;
26431:  Function usGetShellFolder(CSIDL : integer) : string');
26432: end;
26433:
26434: function SerializeStream(const AStream : TStream) : String;
26435: procedure DeserializeToStream(const Data : String; const AStream : TStream);
26436: procedure ReplaceImageInImageList(FromImageList: TImageList; FromIndex: Integer;
26437:                                   ToImageList: TImageList; ToIndex: Integer);
26438: Function GetSIDStringFromUser( AUsername : AnsiString; AServerName : AnsiString) : String');
26439:
26440: TestConvolutionAPI();  procedure TestBackProp(); //> 1.5 GByte MemoryTest
26441: https://github.com/ariaghora/noe
26442: https://github.com/Fr0sT-Brutal/awesome-pascal#machine-learning
26443: https://github.com/TheUnknownOnes/theunknownones/tree/master/Libraries/PascalScriptAddons
26444:
26445: procedure SIRegister_URungeKutta4(CL: TPSPascalCompiler);
26446: begin
26447:  CL.AddConstantN('maxfuncs2','LongInt').SetInt( 10);
26448:   //CL.AddTypeS('float', 'extended');
26449:   CL.AddTypeS('TUserFunction2', 'Function ( T, X, XPrime : double) : double');
26450:   TUserCallBackFunction2', 'Function ( T, X, XPrime : double) : Boolean');
26451:   TNData2', 'record X : double; xPrime : double; end');
26452:   TNvector2', 'array[0..maxfuncs2] of TNData2;
26453:  // TFuncVect  = array[1..MaxFuncs] of TUserFunctionV;
26454:   // TNvector = array[0..maxfuncs] of TNData;
26455:   TUserFunctionV2', 'Function ( V : TNVector2) : double');
26456:   TFuncVect2', 'array[1..MaxFuncs2] of TUserFunctionV2;
26457:   TUserCallbackFunctionV2','Function (V:TNVector2) : boolean');
26458:  Procedure
       RungeKutta2ndOrderIC2(LowerLimit:double;UpperLimit:double;InitialValue:double;InitialDeriv:double;
       ReturnInterval:double;CalcInterval:double;var Error:byte;UserFunc:TUserFunctionV2;UserCallBack:
       TUserCallbackFunctionV2);
26459:  Procedure RungeKutta2ndOrderIC_System2(LowerLimit:double; UpperLimit: double; InitialValues : TNVector2;
       ReturnInterval: double; CalcInterval: double; var Error: byte; NumEquations: Integer; Vector: TFuncVect2;
       UserCallBack: TUserCallbackFunctionV2);
26460: end;
26461:
26462: procedure SIRegister_OverbyteIcsUtils(CL: TPSPascalCompiler);
26463: begin
26464:  CL.AddTypeS('TIcsDbcsLeadBytes', 'TSysCharset');
26465:  'MB_ERR_INVALID_CHARS','LongWord').SetUInt( $00000008);
26466:  'WC_ERR_INVALID_CHARS','LongWord').SetUInt( $80);
26467:  CP_UTF16','LongInt').SetInt( 1200);
26468:  CP_UTF16Be','LongInt').SetInt( 1201);
26469:  CP_UTF32','LongInt').SetInt( 12000);
26470:  CP_UTF32Be','LongInt').SetInt( 12001);
26471:   TOBJECT'),'EIcsStringConvertError');
26472:   TCharsetDetectResult', '(cdrAscii,cdrUtf8,cdrUnknown );
26473:   TIcsNormForm','(icsNormalizationOther,icsNormalizationC,icsNormalizationD,icsNormalizationKD );
26474:  TIcsSearchRecW','record Time:Integer;Size:Integer;Attr: '
26475:    Integer;Name:String;ExcludeAttr:Integer;FindHandle:THandle;FindData:TWin32FindData;end');
26476:   TUnicode_String','record Length:Word;MaximumLength:Word;Buffer:WideChar;end');
26477:   TThreadID', 'LongWord;
26478:  // CL.AddTypeS('PUnicode_String', '^TUnicode_String // will not work');
26479:   SIRegister_TIcsFileStreamW(CL);
26480:  CL.AddConstantN('ICONV_UNICODE','String').SetString( 'UTF-16LE');
26481:  Function IcsIconvNameFromCodePage( CodePage : LongWord) : AnsiString');
26482:  Function IcsIsValidAnsiCodePage( const CP : LongWord) : Boolean');
```

```
26483:  Procedure IcsCharLowerA( var ACh : AnsiChar);
26484:  Function IcsGetCurrentThreadID : TThreadID');
26485:  Function IcsGetFreeDiskSpace( const APath : String) : Int64');
26486:  Function IcsGetLocalTimeZoneBias : LongInt');
26487:  Function IcsDateTimeToUTC( dtDT : TDateTime): TDateTime');
26488:  Function IcsUTCToDateTime( dtDT : TDateTime): TDateTime');
26489:  Function IcsGetTickCount : LongWord');
26490:  //AddDelphiFunction('Function IcsWcToMb(CodePage:LongWord; Flags:Cardinal;
        WStr:PWideChar;WStrLen:Integer; MbStr:PAnsiChar;MbStrLen:Integer;DefaultChar:PAnsiChar;UsedDefaultChar:PLon
26491:  //('Function IcsMbToWc(CodePage:LongWord;Flags:Cardinal;MbStr:PAnsiChar;MbStrLen:Int;
        WStr:PWideChar;WStrLen:Int):Int');
26492:  Function IcsGetDefaultWindowsUnicodeChar( CodePage : LongWord) : WideChar');
26493:  Function IcsGetDefaultWindowsAnsiChar( CodePage : LongWord) : AnsiChar');
26494:  Procedure IcsGetAcp( var CodePage : LongWord);
26495:  Function IcsIsDBCSCodePage( CodePage : LongWord) : Boolean');
26496:  Function IcsIsDBCSLeadByte( Ch : AnsiChar; CodePage : LongWord) : Boolean');
26497:  Function IcsIsMBCSCodePage( CodePage : LongWord) : Boolean');
26498:  Function IcsIsSBCSCodePage( CodePage : LongWord) : Boolean');
26499:  Function IcsGetLeadBytes( CodePage : LongWord) : TIcsDbcsLeadBytes');
26500:  Function icsUnicodeToUsAscii4( const Str: UnicodeString; FailCh : AnsiChar) : AnsiString;
26501:  Function icsUnicodeToUsAscii5( const Str: UnicodeString) : AnsiString;
26502:  Function icsUsAsciiToUnicode6( const Str: RawByteString; FailCh : AnsiChar) : UnicodeString;
26503:  Function icsUsAsciiToUnicode7( const Str: RawByteString) : UnicodeString;
26504:  Function icsUnicodeToAnsi8(const Str:WideChar;ACodePage LongWord; SetCodePage : Boolean) : RawByteString;
26505:  Function icsUnicodeToAnsi9(const Str:UnicodeString;ACodePage:LongWord;SetCodePage:Boolean):RawByteString;
26506:  Function icsUnicodeToAnsi10( const Str: UnicodeString) : RawByteString;
26507:  Function icsAnsiToUnicode11( const Str: AnsiChar; ACodePage : LongWord) : UnicodeString;
26508:  Function icsAnsiToUnicode12( const Str: RawByteString; ACodePage : LongWord) : UnicodeString;
26509:  CL.AddDelphiFunction('Function icsAnsiToUnicode13( const Str : RawByteString) : UnicodeString;
26510:  Function IcsBufferToUnicode14(const Buffer,BufferSize:Integer;BufferCodePage:LongWord;out FailedByteCount
        : Integer)UnicodeString;
26511:  Function IcsBufferToUnicode15(const Buffer,
        BufferSize:Integer;BufferCodePage:LongWord;RaiseFailedBytes:Bool): UnicodeString;
26512:  Function IcsGetWideCharCount(const Buffer,BufferSize:Integer;BufferCodePage:LongWord;out
        InvalidEndByteCount: Integer):Integer;
26513:  Function IcsGetWideChars(const Buffer,
        BufferSize:Integer;BufferCodePage:LongWord;Chars:WideChar;CharCount: Integer):Integer;
26514:  Function
        icsStreamWriteString16(AStream:TStream;Str:WideChar;cLen:Int;ACodePage:LongWord;WriteBOM:Bool):Int;
26515:  Function icsStreamWriteString17(AStream TStream;Str:WideChar;cLen :Integer;ACodePage : LongWord) :
        Integer;
26516:  Function icsStreamWriteString18(AStream:TStream;const Str:UnicodeString;ACodePage
        LongWord;WriteBOM:Bool):Int;
26517:  Function icsStreamWriteString19(AStream:TStream;const Str:UnicodeString;ACodePage : LongWord) : Integer;
26518:  Function icsStreamWriteString20( AStream : TStream; const Str : UnicodeString) : Integer;
26519:  Function icsIsUsAscii21( const Str : RawByteString) : Boolean;
26520:  Function icsIsUsAscii22( const Str : UnicodeString) : Boolean;
26521:  Procedure IcsAppendStr( var Dest : RawByteString; const Src : RawByteString);
26522:  Function icsatoi23( const Str : RawByteString) : Integer;
26523:  Function icsatoi24( const Str : UnicodeString) : Integer;
26524:  Function icsatoi6425( const Str : RawByteString) : Int64;
26525:  Function icsatoi6426( const Str : UnicodeString) : Int64;
26526:  Function IcsCalcTickDiff( const StartTick, EndTick : LongWord) : LongWord');
26527:  Function icsStringToUtf827( const Str : UnicodeString) : RawByteString;
26528:  Function icsStringToUtf828( const Str : RawByteString; ACodePage : LongWord) : RawByteString;
26529:  Function icsUtf8ToStringW( const Str : RawByteString) : UnicodeString');
26530:  Function icsUtf8ToStringA( const Str : RawByteString; ACodePage : LongWord) : AnsiString');
26531:  Function icsCheckUnicodeToAnsi( const Str : UnicodeString; ACodePage : LongWord) : Boolean');
26532:  Function icsIsUtf8TrailByte( const B : Byte) : Boolean');
26533:  Function icsIsUtf8LeadByte( const B : Byte) : Boolean');
26534:  Function IcsUtf8Size( const LeadByte : Byte) : Integer');
26535:  Function icsIsLeadChar( Ch : WideChar) : Boolean');
26536:  Function icsIsUtf8Valid29( const Str : RawByteString) : Boolean;
26537:  //Function IsUtf8Valid30( const Buf : Pointer; Len : Integer) : Boolean;
26538:  //Function CharsetDetect31( const Buf : Pointer; Len : Integer) : TCharsetDetectResult;
26539:  Function icsCharsetDetect32( const Str : RawByteString) : TCharsetDetectResult;
26540:  Function IcsCharNextUtf8( const Str : AnsiChar) : AnsiChar');
26541:  Function IcsCharPrevUtf8( const Start, Current : AnsiChar) : AnsiChar');
26542:  Function icsConvertCodepage(const
        Str:RawByteString;SrcCodePage:LongWord;DstCodePage:LongWord):RawByteString');
26543:  Function icshtoin33( Value : WideChar; Len : Integer) : Integer;
26544:  Function icshtoin34( Value : AnsiChar; Len : Integer) : Integer;
26545:  Function icshtoi235( value : WideChar) : Integer;
26546:  Function icshtoi236( value : AnsiChar) : Integer;
26547:  Function IcsBufferToHex37( const Buf, Size : Integer) : String;
26548:  Function IcsBufferToHex38( const Buf, Size : Integer; Separator : Char) : String;
26549:  Function icsIsXDigit39( Ch : WideChar) : Boolean;
26550:  Function icsIsXDigit40( Ch : AnsiChar) : Boolean;
26551:  Function icsXDigit41( Ch : WideChar) : Integer;
26552:  Function icsXDigit42( Ch : AnsiChar) : Integer;
26553:  Function icsIsCharInSysCharSet43( Ch : WideChar; const ASet : TSysCharSet):Boolean;
26554:  Function icsIsCharInSysCharSet44( Ch : AnsiChar; const ASet : TSysCharSet):Boolean;
26555:  Function icsIsDigit45( Ch : WideChar) : Boolean;
26556:  Function icsIsDigit46( Ch : AnsiChar) : Boolean;
26557:  Function icsIsSpace47( Ch : WideChar) : Boolean;
26558:  Function icsIsSpace48( Ch : AnsiChar) : Boolean;
26559:  Function icsIsCRLF49( Ch : WideChar) : Boolean;
26560:  Function icsIsCRLF50( Ch : AnsiChar) : Boolean;
26561:  Function icsIsSpaceOrCRLF51( Ch : WideChar) : Boolean;
```

```
26562:  Function icsIsSpaceOrCRLF52( Ch : AnsiChar) : Boolean;
26563:  Function icsXDigit2( S : PChar) : Integer');
26564:  Function icsstpblk53( PValue : WideChar) : WideChar;
26565:  Function icsstpblk54( PValue : AnsiChar) : AnsiChar;
26566:  Function IcsStrNextChar( const Str : AnsiChar; ACodePage : LongWord) : AnsiChar');
26567:  Function IcsStrPrevChar( const Start, Current : AnsiChar; ACodePage : LongWord) : AnsiChar');
26568:  Function IcsStrCharLength55( const Str: AnsiChar; ACodePage : LongWord) : Integer;
26569:  Function IcsNextCharIndex56( const S: RawByteString; Index : Integer; ACodePage: LongWord): Integer;
26570:  Function IcsGetBomBytes( ACodePage : LongWord) : TBytes');
26571:  Function IcsGetBufferCodepage57( Buf: AnsiChar; ByteCount : Integer) : LongWord;
26572:  Function IcsGetBufferCodepage58( Buf: AnsiChar; ByteCount : Integer; out BOMSize: Integer): LongWord;
26573:  Function IcsSwap16( Value : Word) : Word');
26574:  Procedure IcsSwap16Buf( Src, Dst : Word; WordCount : Integer);
26575:  Function IcsSwap32( Value : LongWord) : LongWord');
26576:  Procedure IcsSwap32Buf( Src, Dst : LongWord; LongWordCount : Integer);
26577:  Function IcsSwap64( Value : Int64) : Int64');
26578:  Procedure IcsSwap64Buf( Src, Dst : Int64; QuadWordCount : Integer);
26579:  Procedure IcsNameThreadForDebugging( AThreadName : AnsiString; AThreadID : TThreadID);
26580:  Function IcsNormalizeString( const S : UnicodeString; NormForm : TIcsNormForm) : UnicodeString');
26581:  Function IcsCryptGenRandom( var Buf, BufSize : Integer) : Boolean');
26582:  Function IcsRandomInt( const ARange : Integer) : Integer');
26583:  Function IcsFileUtcModified( const FileName : String) : TDateTime');
26584:  //Function IcsInterlockedCompareExchange(var Destination:Pointer;Exchange:Pointer;Comperand
        Pointer):Pointer');
26585:  Function IcsExtractFilePathW( const FileName : UnicodeString) : UnicodeString');
26586:  Function IcsExtractFileDirW( const FileName : UnicodeString) : UnicodeString');
26587:  Function IcsExtractFileDriveW( const FileName : UnicodeString) : UnicodeString');
26588:  Function IcsExtractFileNameW( const FileName : UnicodeString) : UnicodeString');
26589:  Function IcsExtractFileExtW( const FileName : UnicodeString) : UnicodeString');
26590:  Function IcsExpandFileNameW( const FileName : UnicodeString) : UnicodeString');
26591:  Function IcsExtractNameOnlyW( FileName : UnicodeString) : UnicodeString');
26592:  Function IcsChangeFileExtW( const FileName, Extension : UnicodeString) : UnicodeString');
26593:  Function IcsStrAllocW( Len : Cardinal) : WideChar');
26594:  Function IcsStrLenW( Str : WideChar) : Cardinal');
26595:  Function IcsAnsiCompareFileNameW59( const S1, S2 : UnicodeString) : Integer;
26596:  Function IcsAnsiCompareFileNameW60( const Utf8S1, Utf8S2 : UTF8String) : Integer;
26597:  Function IcsDirExistsW61( const FileName : WideChar) : Boolean;
26598:  Function IcsDirExistsW62( const FileName : UnicodeString) : Boolean;
26599:  Function IcsDirExistsW63( const Utf8FileName : UTF8String) : Boolean;
26600:  Function IcsFindFirstW64( const Path : UnicodeString; Attr : Integer; var F: TIcsSearchRecW):Integer;
26601:  Function IcsFindFirstW65( const Utf8Path : UTF8String; Attr : Integer;var F: TIcsSearchRecW):Integer;
26602:  Procedure IcsFindCloseW( var F : TIcsSearchRecW);
26603:  Function IcsFindNextW( var F : TIcsSearchRecW) : Integer');
26604:  Function IcsIncludeTrailingPathDelimiterW( const S : UnicodeString) : UnicodeString');
26605:  Function IcsExcludeTrailingPathDelimiterW( const S : UnicodeString) : UnicodeString');
26606:  Function IcsExtractLastDir66( const Path : RawByteString) : RawByteString;
26607:  Function IcsExtractLastDir67( const Path : UnicodeString) : UnicodeString;
26608:  Function IcsFileGetAttrW68( const FileName : UnicodeString) : Integer;
26609:  Function IcsFileGetAttrW69( const Utf8FileName : UTF8String) : Integer;
26610:  Function IcsFileSetAttrW70( const FileName : UnicodeString; Attr : Integer) : Integer;
26611:  Function IcsFileSetAttrW71( const Utf8FileName : UTF8String; Attr : Integer) : Integer;
26612:  Function IcsDeleteFileW72( const FileName : UnicodeString) : Boolean;
26613:  Function IcsDeleteFileW73( const Utf8FileName : UTF8String) : Boolean;
26614:  Function IcsRenameFileW74( const OldName, NewName : UnicodeString) : Boolean;
26615:  Function IcsRenameFileW75( const Utf8OldName, Utf8NewName : UTF8String) : Boolean;
26616:  Function IcsForceDirectoriesW76( Dir : UnicodeString) : Boolean;
26617:  Function IcsForceDirectoriesW77( Utf8Dir : UTF8String) : Boolean;
26618:  Function IcsCreateDirW78( const Dir : UnicodeString) : Boolean;
26619:  Function IcsCreateDirW79( const Utf8Dir : UTF8String) : Boolean;
26620:  Function IcsRemoveDirW80( const Dir : UnicodeString) : Boolean;
26621:  Function IcsRemoveDirW81( const Utf8Dir : UTF8String) : Boolean;
26622:  Function IcsFileAgeW82( const FileName : UnicodeString) : Integer;
26623:  Function IcsFileAgeW83( const Utf8FileName : UTF8String) : Integer;
26624:  Function IcsFileExistsW84( const FileName : UnicodeString) : Boolean;
26625:  Function IcsFileExistsW85( const Utf8FileName : UTF8String) : Boolean;
26626:  Function IcsAnsiLowerCaseW( const S : UnicodeString) : UnicodeString');
26627:  Function IcsAnsiUpperCaseW( const S : UnicodeString) : UnicodeString');
26628:  Function IcsMakeWord( L, H : Byte) : Word');
26629:  Function IcsMakeLong( L, H : Word) : Longint');
26630:  Function IcsHiWord( LW : LongWord) : Word');
26631:  Function IcsHiByte( W : Word) : Byte');
26632:  Function IcsLoByte( W : Word) : Byte');
26633:  Function IcsLoWord( LW : LongWord) : Word');
26634:  Procedure IcsCheckOSError( ALastError : Integer);
26635:  Function IcsIntToStrA( N : Integer) : AnsiString');
26636:  Function IcsIntToHexA( N : Integer; Digits : Byte) : AnsiString');
26637:  Function IcsTrim86( const Str : AnsiString) : AnsiString;
26638:  Function IcsTrim87( const Str : UnicodeString) : UnicodeString;
26639:  Function IcsLowerCase88( const S : AnsiString) : AnsiString;
26640:  Function IcsLowerCase89( const S : UnicodeString) : UnicodeString;
26641:  Function IcsUpperCase90( const S : AnsiString) : AnsiString;
26642:  Function IcsUpperCase91( const S : UnicodeString) : UnicodeString;
26643:  Function IcsUpperCaseA( const S : AnsiString) : AnsiString');
26644:  Function IcsLowerCaseA( const S : AnsiString) : AnsiString');
26645:  Function IcsCompareTextA( const S1, S2 : AnsiString) : Integer');
26646:  Function IcsTrimA( const Str : AnsiString) : AnsiString');
26647:  Function IcsSameTextA( const S1, S2 : AnsiString) : Boolean');
26648:  Function IcsCompareStr92( const S1, S2 : AnsiString) : Integer;
26649:  Function IcsCompareStr93( const S1, S2 : UnicodeString) : Integer;
```

```
26650:  Function IcsCompareText94( const S1, S2 : AnsiString) : Integer;
26651:  Function IcsCompareText95( const S1, S2 : UnicodeString) : Integer;
26652:  Function IcsStrLen96( const Str : AnsiChar) : Cardinal;
26653:  Function IcsStrLen97( const Str : WideChar) : Cardinal;
26654:  Function IcsStrPas98( const Str : AnsiChar) : AnsiString;
26655:  Function IcsStrPas99( const Str : WideChar) : string;
26656:  Function IcsStrCopy100( Dest : AnsiChar; const Source : AnsiChar) : AnsiChar;
26657:  Function IcsStrCopy101( Dest : WideChar; const Source : WideChar) : WideChar;
26658:  Function IcsStrPCopy102( Dest : PChar; const Source : string) : PChar;
26659:  Function IcsStrPCopy103( Dest : AnsiChar; const Source : AnsiString) : AnsiChar;
26660:  Function IcsStrPLCopy104( Dest : PChar; const Source : String; MaxLen : Cardinal) : PChar;
26661:  Function IcsStrPLCopy105( Dest : AnsiChar; const Source : AnsiString; MaxLen : Cardinal) : AnsiChar;
26662:  Function
       IcsStrCompOrdinalW(Str1:WideChar;Str1Length:Int;Str2:WideChar;Str2Length:Int;IgnoreCase:Bool):Int;
26663:  //Func
       RtlCompareUnicodeString(String1:PUNICODE_STRING;String2:PUNICODE_STRING;CaseInsensitive:BOOL):LongInt');
26664:  Function icsIsDebuggerPresent : BOOL');
26665:   SIRegister_TIcsIntegerList(CL);
26666:   SIRegister_TIcsCriticalSection(CL);
26667: end;
26668:
26669: procedure SIRegister_TCustomApplication(CL: TPSPascalCompiler);
26670: begin
26671:   //with RegClassS(CL,'TComponent', 'TCustomApplication') do
26672:   with CL.AddClassN(CL.FindClass('TComponent'),'TCustomApplication') do begin
26673:     Constructor Create( AOwner : TComponent);
26674:     Procedure HandleException( Sender : TObject);
26675:     Procedure Initialize');
26676:     Procedure Run');
26677:     Procedure ShowException( E : Exception);
26678:     Procedure Terminate');
26679:     Procedure Terminate1( AExitCode : Integer);
26680:     Function FindOptionIndex( const S : String; var Longopt : Boolean; StartAt : Integer) : Integer');
26681:     Function GetOptionValue( const S : String) : String');
26682:     Function GetOptionValue1( const C : Char; const S : String) : String');
26683:     Function GetOptionValues( const C : Char; const S : String) : TStringDynArray');
26684:     Function HasOption( const S : String) : Boolean');
26685:     Function HasOption1( const C : Char; const S : String) : Boolean');
26686:     Function CheckOptions(const ShortOptions:String;const Longopts:TStrings;Opts,
       NonOpts:TStrings;AllErrors: Bool):String;
26687:     Function CheckOptions1(const ShortOptions:String;const Longopts:array of string;Opts,NonOpts:TStrings;
       AllErrors:Bool):String;
26688:     Function CheckOptions2(const ShortOptions:String;const Longopts:TStrings; AllErrors:Boolean):String');
26689:     Function CheckOptions3(const ShortOptions:String;const LongOpts:array of
       string;AllErrors:Boolean):String;
26690:     Function CheckOptions4( const ShortOptions:String;const LongOpts:String; AllErrors:Boolean):String');
26691:     Function GetNonOptions(const ShortOptions: String; const Longopts:array of string) : TStringDynArray');
26692:     Procedure GetNonOptions1(const ShortOptions:String;const Longopts:array of string;NonOptions:TStrings);
26693:     Procedure GetEnvironmentList( List : TStrings; NamesOnly : Boolean);
26694:     Procedure GetEnvironmentList1( List : TStrings);
26695:     Procedure Log(EventType : TEventType; const Msg : String);
26696:     Procedure Log1(EventType : TEventType; const Fmt : String; const Args : array of string);
26697:     ExeName', 'string', iptr);
26698:     Terminated', 'Boolean', iptr);
26699:     Title', 'string', iptrw);
26700:     OnException', 'TExceptionEvent', iptrw);
26701:     ConsoleApplication', 'Boolean', iptr);
26702:     Location', 'String', iptr);
26703:     Params', 'String integer', iptr);
26704:     ParamCount', 'Integer', iptr);
26705:     EnvironmentVariable', 'String String', iptr);
26706:     OptionChar', 'Char', iptrw);
26707:     CaseSensitiveOptions', 'Boolean', iptrw);
26708:     StopOnException', 'Boolean', iptrw);
26709:     ExceptionExitCode', 'Longint', iptrw);
26710:     ExceptObject', 'Exception', iptrw);
26711:     ExceptObjectJS', 'JSValue', iptrw);
26712:     EventLogFilter', 'TEventLogTypes', iptrw);
26713:   end;
26714: end;
26715:
26716:
26717: ADD 4.7.6.10 III
26718: CL.AddDelphiFunction('function list_modules(aexename: string): string;
26719: CL.AddDelphiFunction('function list_units(aexename: string): string;
26720: check //ListView_SetItemState(H, I, Data, LVIS_SELECTED); commctrl
26721:
26722: procedure GetResourceHeader(var ResourceHeader: String; const FormName: String;
26723:                                                          const FileSize: Integer);
26724: GetResourceFormFile(List: TStrings; const FormName, FileName: String) : Boolean;
26725: function GetImageBase(const FileName: TFileName): DWORD;
26726: procedure ListViewSelectAll(ListView: TListView; Deselect: Boolean);
26727: function ListView_SetItemState(hwndLV: HWND; i: Integer; data, mask: UINT): Bool;
26728: function ListView_GetItemState(hwndLV: HWND; i, mask: Integer): Integer;
26729: function CreateOrGetOleObject(const ClassName: string): IDispatch;
26730: function FmtStrToInt(S: string): Integer;
26731: function MakeCellStrLV(const Text: String; Index: Integer; listview: TListview): String;
26732: https://docwiki.embarcadero.com/Libraries/Sydney/en/System.SysUtils.EmptyStr
26733: const emptystring= '';
```

```
26734:
26735: procedure SIRegister_TIdHashMessageDigest5(CL: TPSPascalCompiler);
26736: begin
26737:   //with RegClassS(CL,'TIdHashMessageDigest4', 'TIdHashMessageDigest5') do
26738:   with CL.AddClassN(CL.FindClass('TIdHashMessageDigest4'),'TIdHashMessageDigest5') do begin
26739:     Constructor Create;
26740:     Procedure Free;
26741:     Function HashValue(const ASrc: string): T4x4LongWordRecord');
26742:     Function HashValue1(AStream : TStream): T4x4LongWordRecord');
26743:     Function HashValueString(const ASrc: string): T4x4LongWordRecord');
26744:     Function HashValueStream(AStream : TStream): T4x4LongWordRecord');
26745:     Function AsHex( const AValue : T4x4LongWordRecord) : string');
26746:     function hashStreamAsHex(s: TStream): string;
26747:    end;
26748: end;
26749:
26750: procedure SIRegister_SeSHA256(CL: TPSPascalCompiler);
26751: begin
26752:  CL.AddConstantN('seHEADERSIZE','LongInt').SetInt( 80);
26753:  CL.AddConstantN('emptystring','string').setstring('');
26754:  CL.AddTypeS('T32se', 'array [0 .. 31] of byte;
26755:  CL.AddTypeS('THeaderse', 'array [0 .. seHEADERSIZE - 1] of byte;
26756:  //T32 = array [0 .. 31] of byte;
26757:   // To store the block header
26758:  // THeader = array [0 .. HEADERSIZE - 1] of byte;
26759:  // CL.AddTypeS('PSHA256HASH', '^TSHA256HASH // will not work');
26760:  Function CalcSHA256String( Msg : AnsiString) : TSHA256HASH;
26761:  Function CalcSHA256Stream( Stream : TStream) : TSHA256HASH;
26762:  Function seSHA256ToStr( Hash : TSHA256HASH) : String');
26763:  Function SHA256ToBinaryStr( Hash : TSHA256HASH) : AnsiString');
26764:  Function ReverseHash( Hash : string) : string');
26765:  Function GetMemoryStream : TMemoryStream');
26766:  Function T32ToString( const at32 : T32se) : string');
26767:  Function CalcHeaderSHA256( aHeader : THeaderse) : TSHA256HASH;
26768: end;
26769:
26770: procedure SIRegister_BlocksUnit(CL: TPSPascalCompiler);
26771: begin
26772:   CL.AddTypeS('TCrypto', '( tcBitcoin, tcPascalCoin );
26773:   CL.AddTypeS('TNetbc', '( tnMainNet, tnTestNet );
26774:   CL.AddTypeS('PByte', 'Byte');
26775:   SIRegister_TBlockFile(CL);
26776:   CL.AddTypeS('TBlockHeader', record versionNumber : UInt32; aPreviousBlockHas'
26777:    +h: T32se; aMerkleRoot: T32se; time: UInt32; DifficultyTarget : UInt32; nonce: UInt32; end');
26778:   SIRegister_TInputbc(CL);
26779:   SIRegister_TOutputbc(CL);
26780:   SIRegister_TInputsbc(CL);
26781:   SIRegister_TOutputsbc(CL);
26782:   SIRegister_TTransaction(CL);
26783:   SIRegister_TBlockTransactions(CL);
26784:   SIRegister_TBlockRecord(CL);
26785:   CL.AddTypeS(TStartFileBlockFoundNotify', 'Procedure ( const aBlockFiles: tstringlist);
26786:   TFoundFileBlockNotify', 'Procedure ( const aBlockFile : TBlockFi'
26787:    +le; const actualFileBlockNumber, totalBlockFiles : integer; var next : boolean);
26788:   TEndFilesBlockFoundNotify', 'Procedure ( const aBlockFiles : tstringlist);
26789:   TFoundBlockNotify','Procedure(const aBlock: TBlockRecord; var findnext: boolean);
26790:   TBlockProcessStepNotify', 'Procedure ( const aPos, asize : int64);
26791:   TEndProcessBlockFile', 'Procedure ( const aBlockFile : TBlockFile);
26792:   SIRegister_TBlocks(CL);
26793: end;
26794:
26795: procedure SIRegister_DelticsCommandLine(CL: TPSPascalCompiler);
26796: begin
26797:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCommandLine');
26798:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCommandLineSwitch');
26799:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidCommandLine');
26800:   //CL.AddTypeS('PSwitchDef', '^TSwitchDef // will not work');
26801:   CL.AddTypeS('TSwitchDef', 'record Name : String; Switch : String; Default : S'
26802:    +'tring; MinValues : Integer; MaxValues : Integer; Separator : Char; end');
26803:   SIRegister_TCommandLine(CL);
26804:   SIRegister_TCommandLineSwitch(CL);
26805:  Procedure CommandLineToArgs( const aString : String; const aArgs : TStrings);
26806: end;
26807:
26808: (*----------------------------------------------------------------------------*)
26809: procedure SIRegister_DelticsStrUtils(CL: TPSPascalCompiler);
26810: begin
26811:  CL.AddConstantN('LIKENESS_MATCH','Char').SetString( #255);
26812:   CL.AddTypeS('TANSIStringArray', 'array of ANSIString');
26813:   //CL.AddTypeS('TStringArray', 'array of String');
26814:   SIRegister_IStringList(CL);
26815:   SIRegister_IStringTool(CL);
26816:   SIRegister_TStringTool(CL);
26817:  Procedure dArrayInit( var aArray : TANSIStringArray; const aSize : Integer);
26818:  Procedure ArrayAdd1( var aArray : TANSIStringArray; const aValue : ANSIString);
26819:  Procedure ArrayAdd2( var aArray : TANSIStringArray; const aValue : array of ANSIString);
26820:  Procedure ArrayExtend( var aArray : TANSIStringArray; const aCount : Integer);
26821:  Function Likeness( const A : ANSIString; const B : ANSIString) : Integer');
26822:  Function LikenessEx(const A:ANSIString;const B:ANSIString; var strA:ANSIString;var
       strB:ANSIString):Integer;
```

```
26823:  Function LikenessEx2(const A:ANSIString;const B:ANSIString;var strA:ANSIString;var strB:ANSIString;var
        iSame :
26824:                        Int;var iTotal:Int):Int;
26825:  Function CamelCapsToWords( const aString : String) : String');
26826:  Function ReverseStr( const aString : String) : String;
26827:  Function ReverseStr2( const aString : ANSIString) : ANSIString;
26828:  Function dSplit( const aString : String; const aDelim : Char; const aSlices : TStrings) : Integer;
26829:  Function dSplit1( const aString : String; const aDelim : Char; var aSlices : TStringArray) : Integer;
26830:  Procedure dSplit2( const aString : ANSIString; const aDelim : ANSIChar; var aSlices : TANSIStringArray);
26831:  Function StrBeginsWith( const aString : String; const aBeginning : String) : Boolean');
26832:  Function StrBeginsWithText( const aString : String; const aBeginning : String) : Boolean');
26833:  Function dStrContains( const aString : String; const aSubString : String) : Boolean');
26834:  Function StrContainsText( const aString : String; const aSubString : String) : Boolean');
26835:  Function StrEndsWith( const aString : String; const aEnd : String) : Boolean');
26836:  Function StrEndsWithText( const aString : String; const aEnd : String) : Boolean');
26837:  Function StrLPad( const aString : String; const aWidth : Integer; const aTrunc : Boolean) : String');
26838:  Function StrRPad( const aString : String; const aWidth : Integer; const aTrunc : Boolean) : String');
26839:  Function StrDequote( const aString : String) : String');
26840:  Function StrPop( var aString : String; const aDelim : Char) : String');
26841:  Function StrPopQuoted( var aString : String; const aDelim : Char) : String');
26842:  Function dIsAlpha( const aChar : Char) : Boolean');
26843:  Function dIsDigit( const aChar : Char) : Boolean');
26844:  Function dIsLower( const aChar : Char) : Boolean');
26845:  Function dIsUpper( const aChar : Char) : Boolean');
26846:   //SIRegister_TManagedStringList(CL);
26847: function delticsExplode(const Separator, S: string; Limit: Integer = 0): TStringArray;
26848: function delticsImplode(const Glue: string; const Pieces: array of string): string;
26849: end;
26850:
26851: (*----------------------------------------------------------------------------*)
26852: procedure SIRegister_DelticsSysUtils(CL: TPSPascalCompiler);
26853: begin
26854:  CL.AddConstantN('delticsEmptyStr','String').SetString( '');
26855:  //CL.AddConstantN('NullGUID','TGUID').SetString( '{00000000-0000-0000-0000-000000000000}');
26856:   SIRegister_IAutoFree(CL);
26857:   //CL.AddTypeS('Exception', 'Exception');
26858:   SIRegister_ENotImplemented(CL);
26859:   CL.AddClassN(CL.FindClass('TOBJECT'),'EAccessDenied');
26860:   CL.AddTypeS('TGUIDFormat', '( gfDefault, gfNoBraces, gfNoHyphens, gfDigitsOnly );
26861:   CL.AddTypeS('TRoundingStrategy', '( rsDefault, rsAwayFromZero, rsTowardsZero );
26862:   CL.AddTypeS('TriBoolean', '( tbUnknown, tbTRUE, tbFALSE );
26863:  Procedure deltCloneList( const aSource : TStringList; const aDest : TStringList);
26864:  //CL.AddDelphiFunction('Function AutoFree( const aReference : PObject) : IUnknown;
26865:  //('Function AutoFree5( const aReferences : array of PObject) : IUnknown;
26866:  Procedure deltFreeAndNIL( var aObject: Tstrings);
26867:  //Procedure NILRefs( const aObjects : array of PObject);
26868:  Function deltIfThen7( aValue : Boolean; aTrue, aFalse : Boolean) : Boolean;
26869:  Function deltIfThen8( aValue : Boolean; aTrue, aFalse : TObject) : TObject;
26870:  Function deltIfThen9( aValue : Boolean; aTrue, aFalse : Integer) : Integer;
26871:  Function deltIfThen10( aValue : Boolean; aTrue, aFalse : ANSIString) : ANSIString;
26872:  Function deltIfThen11( aValue : Boolean; aTrue, aFalse : String) : String;
26873:  Function deltIfThenInt( aValue : Boolean; aTrue, aFalse : Integer) : Integer');
26874:  Function deltStringIndex( const aString : String; const aCases : array of String : Integer');
26875:  Function deltTextIndex( const aString : String; const aCases : array of String): Integer');
26876:  Function deltMin64( ValueA, ValueB : Int64) : Int64');
26877:  Function deltMin12( ValueA, ValueB : Cardinal) : Cardinal;
26878:  Function deltMin13( ValueA, ValueB : Integer) : Integer;
26879:  Function deltMax64( ValueA, ValueB : Int64) : Int64');
26880:  Function deltMax14( ValueA, ValueB : Cardinal) : Cardinal;
26881:  Function deltMax15( ValueA, ValueB : Integer) : Integer;
26882:  Procedure deltExchange( var A, B, aSize : LongWord);
26883:  Function deltGUIDToString( const aGUID : TGUID; const aFormat : TGUIDFormat) : String');
26884:  Function deltStringToGUID( const aString : String) : TGUID');
26885:  Function deltTryStringToGUID( const aString : String; var aGUID : TGUID) : Boolean');
26886:  Function deltIsNull( const aValue : TGUID) : Boolean');
26887:  Function deltNewGUID : TGUID');
26888:  Function deltSameGUID( const GUIDA, GUIDB : TGUID) : Boolean');
26889:  Procedure deltAddTrailingBackslash( var aString : String);
26890:  Procedure deltRemoveTrailingBackslash( var aString : String);
26891:  Function deltBinToHex( const aBuf : string; const aSize : Integer) : String');
26892:  Function deltReverseBytes16( const aValue : Word) : Word;
26893:  Function deltReverseBytes17( const aValue : LongWord) : LongWord;
26894:  Function deltReverseBytes18( const aValue : Int64) : Int64;
26895:  Function deltRound( const aValue : Extended; const aStrategy : TRoundingStrategy) : Integer');
26896:  //Procedure ForEachComponent(const aComponent:TComponent; const aProc:TComponentProc;const
        aRecursive:Bool; const aClass:TComponentClass);
26897:  Function deltExec( const aEXE : String; const aCommandLine : String) : Cardinal');
26898:  Procedure deltExecAndWait( const aEXE : String; const aCommandLine : String);
26899:  Function deltFindProcess( const aEXEName : String; var aProcess : TProcessEntry32): Boolean');
26900:  Function deltProcessExists( const aEXEName : String) : Boolean');
26901:  Function deltRegisterDLL( const aFileName : String; const aRegistrationProc:String):Boolean');
26902:  Function deltIsTRUE( const aTri : TriBoolean) : Boolean');
26903:  Function deltIsFALSE( const aTri : TriBoolean) : Boolean');
26904:  Function deltIsKnown( const aTri : TriBoolean) : Boolean');
26905: end;
26906:
26907: procedure SIRegister_U_Splines(CL: TPSPascalCompiler);
26908: begin
26909:  CL.AddConstantN('MaxNoVertices','LongInt').SetInt( 100);
```

```
26910:   CL.AddTypeS('TSplineVertex', 'record X : Single; Y : single; end');
26911:   CL.AddTypeS('TSplineRow', 'array of TSplineVertex');
26912:   SIRegister_TBSpline(CL);
26913:   SIRegister_TSplines(CL);
26914:  //CL.AddDelphiFunction('Procedure Register');
26915:  SIRegister_U_SpringMass2(CL);
26916:  CL.AddDelphiFunction('function getPython: string;
26917:  CL.AddDelphiFunction('function getPython2: string;
26918:  CL.AddDelphiFunction('function getPython3: string;
26919: end;
26920:
26921: procedure SIRegister_MARSCoreUtils(CL: TPSPascalCompiler);
26922: begin
26923:  Function CreateCompactGuidStr : string');
26924:  //CL.AddDelphiFunction('Function BooleanToTJSON( AValue : Boolean) : TJSONValue');
26925:  Function SmartConcat(const AArgs:array of string;const ADelimiter:string;const AAvoidDuplicateDelimiter:
       Boolean;const ATrim:Boolean; const ACaseInsensitive:Boolean):string');
26926:  Function EnsurePrefix2( const AString, APrefix : string; const AIgnoreCase : Boolean) : string');
26927:  Function EnsureSuffix( const AString, ASuffix : string; const AIgnoreCase : Boolean) : string');
26928:  Function StringArrayToString( const AArray : array of string; const ADelimiter : string) : string');
26929:  Procedure CopyStream2(ASourceStream,
       ADestStream:TStream;AOverWriteDest:Bool;AThenResetDestPosition:Boolean);
26930:  Function DateToISO8601( const ADate : TDateTime; AInputIsUTC : Boolean) : string');
26931:  Function ISO8601ToDate( const AISODate : string; AReturnUTC : Boolean) : TDateTime');
26932:  Function IsMask( const AString : string) : Boolean');
26933:  Function MatchesMask2( const AString, AMask : string) : Boolean');
26934:  Function StreamToBase64( const AStream : TStream) : string');
26935:  Procedure Base64ToStream( const ABase64 : string; const ADestStream : TStream);
26936:  Function StreamToBytes( const ASource : TStream) : TBytes');
26937:  //Function GetEncodingName( const AEncoding : TEncoding) : string');
26938:  function StringFallback(const AStrings: array of string; const ADefault: string): string;
26939:  function StripPrefix(const APrefix, AString: string): string;
26940: function StripSuffix(const ASuffix, AString: string): string;
26941: end;
26942:
26943: procedure SIRegister_U_CoasterB(CL: TPSPascalCompiler);
26944: begin
26945:   //CL.AddTypeS('float', 'single');
26946:   CL.AddTypeS(tcoasterRealPoint', 'record x : single; y : single; end');
26947:   TTrackPoint', 'record x : single; y : single; angleright : single; '
26948:    +sinangleright : single; cosangleright : single; quadrantright : integer; len'
26949:    +gthright : single; radiusright : single; end');
26950:    TTrackPointArray', 'array of TTrackPoint'); //carts', 'array of TCartLocRec'
26951:    TTrackRec', 'record x : single; y : single; index : integer; distT'
26952:    +oend : single; quadrant : integer; radius: single; angle: single; sinangle '
26953:    +: single; cosangle : single; end');
26954:   TCartLocRec', 'record p1 : tpoint; p2: tpoint; p3: tpoint; p4: tpoint; wr: integer; end');
26955:    TCartLocRecArray', 'array of TCartLocRec');
26956:   TLoopType', '( Insideloop, Outsideloop, none );
26957:   TTemplateMode', '( normal, makeline1, makeline2, makecircle1, makecircle2, movecircle );
26958:   CL.AddTypeS(ARRsoundstr', 'array [0..5] of string');
26959:    //type ARRstr = array [0..5] of string;
26960:   SIRegister_TcoasterLine(CL);
26961:   SIRegister_TcoasterCircle(CL);
26962:   SIRegister_TCoaster(CL);
26963: end;
26964:
26965: procedure SIRegister_TSpring(CL: TPSPascalCompiler);
26966: begin
26967:   //with RegClassS(CL,'tPanel', 'TSpring') do
26968:   with CL.AddClassN(CL.FindClass('tPanel'),'TSpring') do  begin
26969:     K', 'single', iptrw);  M', 'single', iptrw);
26970:     G', 'single', iptrw);  A', 'single', iptrw);
26971:     x', 'single', iptrw);  v', 'single', iptrw);
26972:     C', 'single', iptrw);  X0', 'single', iptrw);
26973:     V0', 'single', iptrw); XEnd', 'single', iptrw);
26974:     InitiallyConstrained', 'Boolean', iptrw);
26975:     Timeinc', 'single', iptrw);
26976:     Scale', 'single', iptrw);
26977:     Stype', 'integer', iptrw);
26978:     delay', 'integer', iptrw);
26979:     pts', 'Tspringpointarray', iptrw);
26980:     step', 'integer', iptrw);
26981:     wraps', 'integer', iptrw);
26982:     drawColor', 'TColor', iptrw);
26983:     Constructor create( Aowner : TPanel; r : Trect);
26984:     Procedure drawspring( Len : single);
26985:     Procedure erasespring');
26986:     Procedure redrawinitial');
26987:     Function Setup(newV0,newX0,newC,newK,newM,newG,newTimeInc:single;newconstraint:bool;
26988:                            Nbrloops,newStep, newdelay: integer;newcolor:TColor):bool');
26989:     Procedure animate(form1: TForm; memo2: Tmemo);
26990:     Procedure animate2(form1: TForm; memo2: Tmemo; detail: boolean);
26991:     //procedure animate2(form1: TForm; memo2: Tmemo; detail: boolean);
26992:     Procedure sizechanged( sender : TObject);
26993:     Function GetMaxAmp : single');
26994:   end;
26995: end;
26996:
```

```
26997: procedure SIRegister_clJsonParser(CL: TPSPascalCompiler);
26998: begin
26999:   SIRegister_EclJSONError(CL);
27000:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONString');
27001:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONPair');
27002:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONObject');
27003:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONArray');
27004:   SIRegister_TclJSONBase(CL);
27005:   SIRegister_TclJSONPair(CL);
27006:   SIRegister_TclJSONValue(CL);
27007:   SIRegister_TclJSONString(CL);
27008:   SIRegister_TclJSONBoolean(CL);
27009:   SIRegister_TclJSONArray(CL);
27010:   SIRegister_TclJSONObject(CL);
27011:  CL.AddConstantN('cUnexpectedDataEndCode','LongInt').SetInt( - 100);
27012:  'cUnexpectedDataSymbolCode','LongInt').SetInt( - 101);
27013:  'cInvalidControlSymbolCode','LongInt').SetInt( - 102);
27014:  'cInvalidUnicodeEscSequenceCode','LongInt').SetInt( - 103);
27015:  'cUnrecognizedEscSequenceCode','LongInt').SetInt( - 104);
27016:  'cUnexpectedDataTypeCode','LongInt').SetInt( - 106);
27017:  function GetConnectionKind2(var strKind: string): Boolean;
27018: end;
27019:
27020: procedure SIRegister_SynHighlighterPython(CL: TPSPascalCompiler);
27021: begin
27022:   CL.AddTypeS(TtkTokenKindpy', '( tkComment, tkIdentifier, tkKey, tkNull, tkNumb'
27023:    +er, tkSpace, tkString, tkSymbol, tkNonKeyword, tkTrippleQuotedString, tkSy'
27024:    +stemDefined, tkHex, tkOct, tkFloat, tkUnknown );
27025:   CL.AddTypeS('TRangeStatepy', '( rsANil,rsComment,rsUnKnown,rsMultilineString'
27026:    +', rsMultilineString2, rsMultilineString3 );
27027:   SIRegister_TSynPythonSyn(CL);
27028: end;
27029:
27030: procedure SIRegister_DudsCommonDelphi(CL: TPSPascalCompiler);
27031: begin
27032:  Function DelphiIDERunning : Boolean');
27033:  Function GetDelphiVersionRegistryKey( Version : Integer) : String');
27034:  Function GetDelphiRootDirectory( Version : Integer) : String');
27035:  Function IsDelphiVersionInstalled( Version : Integer) : Boolean');
27036:  Function GetDelphiEnvironmentVariables( Version : Integer; const EnvironmentVariables : TStrings) :
       Boolean');
27037:  Function GetDelphiLibraryPaths(Version:Int;const
       LibraryPaths:TStrings;EvaluateMacros:Bool;CheckPaths:Bool):Bool;
27038:  function StrConcat(ArrS:array of string;Space:string):string;
27039:  function StrDelSpc(s:string):string;
27040:  function StrSplit2(S:string;Ch:char):TStringDynArray;
27041:  procedure StrReplaceFirst(var S:string;olds,news:string);
27042:  procedure StrReplaceLast(var S:string;OldS,NewS:string);
27043: procedure ConvertV3ToS(X,Y,Z:extended;var r,a,b:extended);
27044: procedure ConvertSToV3(R,A,B:extended;var X,Y,Z:extended);
27045: procedure InterpolateColor(Color0,Color1:TColor;Count:integer;var TempColor:array of TColor);
27046: function TryStrToextended(const S: string; out Value: extended): Boolean;
27047: function isAVXSupported: Boolean;
27048: function GetEXEVersionData(const FileName: string): TEXEVersionData;
27049: function txtsearch(pat: string; text: string): integer;
27050: procedure GetRegisteredAntiviruses(ProgIDs: TStrings);
27051: type
27052:   TEXEVersionData = record
27053:     CompanyName,
27054:     FileDescription, FileVersion,
27055:     InternalName,
27056:     LegalCopyright, LegalTrademarks,
27057:     OriginalFileName, ProductName,
27058:     ProductVersion, Comments,
27059:     PrivateBuild, SpecialBuild: string;
27060:   end;
27061: type
27062: TNeuronClass = class(TObject)
27063:   private
27064:     FThreshold: Single;
27065:     FInputCount: Integer;
27066:     FInputs:TSingleDynArray;
27067:     FWeights:TSingleDynArray;
27068:     FAnalogOutput: Boolean;
27069:     function GetInput(Index: Integer): Single;
27070:     function GetOutput: Single;
27071:     function GetWeight(Index: Integer): Single;
27072:     procedure SetInput(Index: Integer; const Value: Single);
27073:     procedure SetWeight(Index: Integer; const Value: Single);
27074:     procedure SetInputCount(const Value: Integer);
27075:     function GetLoad: Single;
27076:   public
27077:     Constructor Create(aInputCount:Integer);
27078:     procedure RandomInitialize;
27079:     Property Load:Single read GetLoad;
27080:     property Output:Single read GetOutput;
27081:     property Threshold:Single read FThreshold write FThreshold;
27082:     property Inputs[Index: Integer]: Single read GetInput write SetInput;
27083:     property Weights[Index: Integer]: Single read GetWeight write SetWeight;
```

```
27084:     property InputCount:Integer read FInputCount write SetInputCount;
27085:     property AnalogOutput:Boolean read FAnalogOutput write FAnalogOutput;
27086: end;
27087: TPJConsoleApp:= class(TPJCustomConsoleApp)  Console Application Runner CAR
27088:   public
27089:     // Make all inherited protected properties public
27090:     property StdIn;
27091:     property StdOut;
27092:     property StdErr;
27093:     property CommandLine;
27094:     property CurrentDir;
27095:     property Visible;
27096:     property MaxExecTime;
27097:     property TimeSlice;
27098:     property KillTimedOutProcess;
27099:     property ProcessAttrs;
27100:     property ThreadAttrs;
27101:     property UseNewConsole;
27102:     property ConsoleTitle;
27103:     property ConsoleColors;
27104:     property ScreenBufferSize;
27105:     property WindowPosition;
27106:     property WindowSize;
27107:     property Environment;
27108:     property UnicodeEnvironment;
27109:     property Priority;
27110:     property TimeToLive;
27111:     property ElapsedTime;
27112:     property ProcessInfo;
27113:     property ExitCode;
27114:     property ErrorCode;
27115:     property ErrorMessage;
27116:     property OnStart;
27117:     property OnWork;
27118:     property OnComplete;
27119:   end;
27120:
27121: procedure SIRegister_uHTMLBuilder(CL: TPSPascalCompiler);
27122: begin
27123:   SIRegister_THTMLBase(CL);
27124:   SIRegister_THTMLCell(CL);
27125:   CL.AddTypeS(TAfterAddRow', 'Procedure (Table:THTMLBase; DataSet: TDataSet);
27126:   SIRegister_THTMLItem(CL);
27127:   SIRegister_THTMLParagraph(CL);
27128:   SIRegister_THTMLRow(CL);
27129:   SIRegister_THTMLTable(CL);
27130:   SIRegister_THTMLReport(CL);
27131:   SIRegister_THTMLBuild(CL);
27132: end;
27133: https://github.com/guitorres/htmlbuilder/blob/master/src/uHTMLBuilder.pas
27134:
27135: procedure SIRegister_RestJsonUtils(CL: TPSPascalCompiler);
27136: begin
27137:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidValue');
27138:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidValueForField');
27139:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidSyntax');
27140:   SIRegister_ENoSerializableClass(CL);
27141:   SIRegister_TJsonUtil(CL);
27142:  Function JavaToDelphiDateTime2( const dt : int64) : TDateTime');
27143:  Function DelphiToJavaDateTime2( const dt : TDateTime) : int64');
27144:  Function ISO8601DateToJavaDateTime( const str : String; var ms : Int64) : Boolean');
27145:  Function ISO8601DateToDelphiDateTime( const str : string; var dt : TDateTime) : Boolean');
27146:  Function DelphiDateTimeToISO8601Date2( dt : TDateTime) : string');
27147: end;
27148:
27149: procedure SIRegister_pxQRcode(CL: TPSPascalCompiler);
27150: begin
27151:  CL.AddConstantN('QR_ECLEVEL_L','LongInt').SetInt( 0);
27152:  ('QR_ECLEVEL_M','LongInt').SetInt( 1);
27153:  ('QR_ECLEVEL_Q','LongInt').SetInt( 2);
27154:  ('QR_ECLEVEL_H','LongInt').SetInt( 3);
27155:  Procedure CreateQRCodeBMP( const AText:string; const ABitmapStream:TStream; const ALevel:Byte;const
       ASize: Integer);
27156: end;
27157:
27158: procedure SIRegister_DelphiZXingQRCode(CL: TPSPascalCompiler);
27159: begin
27160:   TQRCodeEncoding','( qrcAuto,qrcNumeric,qrcAlphanumeric,qrcISO88591,qrcUTF8NoBOM,qrcUTF8BOM );
27161:   CL.AddTypeS(T2DBooleanArray', 'array of array of Boolean');
27162:   SIRegister_TDelphiZXingQRCode(CL);
27163: end;
27164:
27165: function DownloadArchive( sURL, sArchivoLocal: String ): boolean;
27166: function GetUrlContent(const Url: string): UTF8String;
27167: procedure FindFiles2List(StartDir,FileMask: string; recursively:boolean; var FilesList: TStringList);
27168: unit UtilsTimeCode;
27169: function TCtoFrames(S:String):Integer;
27170: function FramestoTC(Fi:Integer):String;
27171: function DFtoFrames(S:String):Integer;
```

```
27172: function FramestoDF(Fi:Integer):String;
27173: function ValidDropFrame(S:String):Boolean;
27174:
27175: unit uPSI_KLibUtils;
27176: procedure SIRegister_KLibUtils(CL: TPSPascalCompiler);
27177: begin
27178:  CL.AddTypeS('TResourceklib', 'record name : string; _type : string; end');
27179:  CL.AddTypeS('TAnonymousMethodklib',{reference to} 'procedure');
27180:  CL.AddTypeS('TCallBackklib', 'procedure(msg: string);
27181:  Procedure deleteFilesInDirklib( pathDir : string; const filesToKeep : array of string);
27182:  Procedure deleteFilesInDirWithStartingFileNameklib(dirName:string;
        startingFileName:string;fileType:string);
27183:  Function checkIfFileExistsAndEmptyklib( fileName : string) : boolean');
27184:  Procedure deleteFileIfExistsklib( fileName : string);
27185:  Function getTextFromFileklib( fileName : string) : string');
27186:  Function checkIfThereIsSpaceAvailableOnDriveklib( drive : char; requiredSpaceInBytes:int64): boolean');
27187:  Function getFreeSpaceOnDriveklib( drive : char) : int64');
27188:  Function getIndexOfDriveklib( drive : char) : integer');
27189:  Function getDriveExeklib : char');
27190:  Function getDirSizeklib( path : string) : int64');
27191:  Function getCombinedPathWithCurrentDirklib( pathToCombine : string) : string');
27192:  Function getDirExeklib : string');
27193:  Procedure createDirIfNotExistsklib( dirName : string);
27194:  Function checkIfIsLinuxSubDirklib( subDir : string; mainDir : string) : boolean');
27195:  Function getPathInLinuxStyleklib( path : string) : string');
27196:  Function checkIfIsSubDirklib(subDir : string; mainDir: string; trailingPathDelimiter:char):boolean');
27197:  Function getValidFullPathklib( fileName : string) : string');
27198:  Function checkMD5Fileklib( fileName : string; MD5 : string) : boolean');
27199:  Procedure unzipResourceklib( nameResource : string; destinationDir : string);
27200:  //Function getPNGResource( nameResource : string) : TPngImage');
27201:  Procedure getResourceAsEXEFileklib( nameResource : string; destinationFileName: string);
27202:  Procedure getResourceAsZIPFileklib( nameResource : string; destinationFileName: string);
27203:  Procedure getResourceAsFileklib( resource : TResourceklib; destinationFileName: string);
27204:  Function getResourceAsString( resource : TResourceklib) : string');
27205:  Function getResourceAsStringklib( resource : TResourceklib) : string');
27206:  Function getResourceAsStreamklib( resource : TResourceklib) : TResourceStream');
27207:  Procedure unzipklib( zipFileName : string; destinationDir : string; deleteZipAfterUnzip : boolean);
27208:  //Function checkRequiredFTPProperties( FTPCredentials : TFTPCredentials) : boolean');
27209:  Function getValidItalianTelephoneNumberklib( number : string) : string');
27210:  Function getValidTelephoneNumberklib( number : string) : string');
27211:  Function getRandStringklib( size : integer) : string');
27212:  Function getFirstFileNameInDirklib( dirName : string; fileType: string; fullPath: boolean): string');
27213:  Function getFileNamesListInDirklib( dirName : string; fileType: string; fullPath: boolean): TStringList');
27214:  Procedure saveToFileklib( source : string; fileName : string);
27215:  Function getCombinedPathklib( path1 : string; path2 : string) : string');
27216:  Function getCurrentDayOfWeekAsStringklib : string');
27217:  Function getDayOfWeekAsStringklib( date : TDateTime) : string');
27218:  Function getCurrentDateTimeAsStringklib : string');
27219:  Function getDateTimeAsStringklib( date : TDateTime) : string');
27220:  Function getCurrentDateAsStringklib : string');
27221:  Function getDateAsStringklib( date : TDateTime) : string');
27222:  Function getCurrentTimeStampklib : string');
27223:  Function getCurrentDateTimeAsStringWithFormattingklib( formatting : string) : string');
27224:  Function getDateTimeAsStringWithFormattingklib( value : TDateTime; formatting: string): string');
27225:  Function getCurrentDateTimeklib : TDateTime');
27226:  Function getParsedXMLstringklib( mainString : string) : string');
27227:  Function getDoubleQuotedStringklib( mainString : string): string');
27228:  Function getSingleQuotedStringklib( mainString : string): string');
27229:  Function
        getMainStringWithSubStringInsertedklib(mainString:string;insertedString:string;index:integer):string;
27230:  Function getStringWithoutLineBreaksklib( mainString : string; substituteString : string) : string');
27231:  Function getCSVFieldFromStringAsDateklib( mainString :string; index : integer; delimiter : Char): TDate;
27232:  Function getCSVFieldFromStringAsDate1klib(mainString:string;index:integer;formatSettings:TFormatSettings;
        delimiter:Char):TDate;
27233:  Function getCSVFieldFromStringAsDoubleklib( mainString:string; index:integer;delimiter:Char):Double;
27234:  Function
        getCSVFieldFromStringAsDouble2klib(mainString:string;index:integer;formatSettings:TFormatSettings;
        delimiter:Char):Double;
27235:  Function getCSVFieldFromStringAsIntegerklib(mainString:string;index:integer;delimiter:Char):integer');
27236:  Function getCSVFieldFromStringklib( mainString : string; index : integer; delimiter : Char): string');
27237:  Function getNumberOfLinesInStrFixedWordWrapklib( source : string) : integer');
27238:  Function stringToStrFixedWordWrapklib( source : string; fixedLen : Integer) : string');
27239:  Function stringToStringListWithFixedLenklib( source : string; fixedLen : Integer) : TStringList');
27240:  Function stringToStringListWithDelimiterklib( value : string; delimiter : Char) : TStringList');
27241:  Function stringToTStringListklib( source : string) : TStringList');
27242:  Function arrayOfStringToTStringListklib( arrayOfStrings : array of string) : TStringList');
27243:  Procedure splitStringsklib(source:str;delimiter:str;var destFirstString:str;var destSecondString: string);
27244:  Procedure splitStrings2klib(source:str;delimiterPosition:int;var destFirstString:str;var
        destSecondString:str);
27245:  Function getMergedStringsklib( firstString : string; secondString : string; delimiter: string): string');
27246:  Function checkIfEmailIsValidklib( email : string) : boolean');
27247:  Function
        checkIfMainStringContainsSubStringNoCaseSensitiveklib(mainString:string;subString:string):boolean;
27248:  Function
        checkIfMainStringContainsSubStringklib(mainString:str;subString:str;caseSensitiveSearch:bool):bool;
27249:  Function getDoubleAsStringklib( value : Double; decimalSeparator : char) : string');
27250:  Function getFloatToStrDecimalSeparatorklib : char');
27251:  Procedure tryToExecuteProcedureklib( myProcedure: TAnonymousMethodklib; raiseExceptionEnabled: boolean);
27252:  Procedure tryToExecuteProcedure2klib( myProcedure : TCallBackklib; raiseExceptionEnabled : boolean);
```

```
27253:  //Procedure tryToExecuteProcedure3( myProcedure : TProcedure; raiseExceptionEnabled : boolean);
27254:  Procedure executeProcedureklib( myProcedure : TAnonymousMethodklib);
27255:  Procedure executeProcedure2klib( myProcedure : TCallBackklib);
27256:  function setResourceHInstanceklib(aresource: longword): longword;
27257:  function setResourceHInstanceklib(aresource: longword): longword;
27258:  function executeAndWaitExe(fileName:string; params:string; exceptionIfReturnCodeIsNot0:boolean):LongInt;
27259: end;
27260:
27261: procedure SIRegister_KLibWindows(CL: TPSPascalCompiler);
27262: begin
27263:  //CL.AddDelphiFunction('Procedure downloadFile( info : TDownloadInfo; forceOverwrite : boolean);
27264:  Function getFirstPortAvaliableklib( defaultPort : integer; host : string) : integer');
27265:  Function checkIfPortIsAvaliableklib( host : string; port : Word) : boolean');
27266:  Function checkIfAddressIsLocalhostklib( address : string) : boolean');
27267:  Function getIPFromHostNameklib( hostName : string) : string');
27268:  Function getIPklib : string');
27269:  Function checkIfRunUnderWineklib : boolean');
27270:  Function checkIfWindowsArchitectureIsX64klib : boolean');
27271:   CL.AddTypeS('TPIDCredentials', 'record ownerUserName : string; domain : string; end');
27272:   CL.AddTypeS('TWindowsArchitectureklib', '( WindowsX86, WindowsX64 );
27273:  Function getWindowsArchitectureklib : TWindowsArchitectureklib');
27274:  Function checkIfUserIsAdminklib : boolean');
27275:  Function IsUserAnAdminklib : boolean');
27276:   CL.AddTypeS('TShowWindowType','( _SW_HIDE, _SW_SHOWNORMAL, _SW_NORMAL, _SW_S'
27277:    +'HOWMINIMIZED, _SW_SHOWMAXIMIZED, _SW_MAXIMIZE, _SW_SHOWNOACTIVATE, _SW_SHO'
27278:    +'W, _SW_MINIMIZE,_SW_SHOWMINNOACTIVE,_SW_SHOWNA,_SW_RESTORE,_SW_SHOWDEFAULT,_SW_FORCEMINIMIZE,_SW_MAX
);
27279:  Procedure openWebPageWithDefaultBrowserklib( url : string);
27280:  Function shellExecuteOpenklib( fileName : string; params : string; directory : string; showWindowType :
TShowWindowType; exceptionIfFunctionFails : boolean) : integer');
27281:  Function shellExecuteExeAsAdminklib(fileName:string;params:string;showWindowType : TShowWindowType;
exceptionIfFunctionFails:boolean) : integer');
27282:  Function shellExecuteExeklib(fileName:string;params:string;showWindowType:TShowWindowType;
exceptionIfFunctionFails:boolean;operation: string) : integer');
27283:  Function myShellExecuteklib( handle : integer; operation : string; fileName : string; params : string;
directory : string; showWindowType : TShowWindowType; exceptionIfFunctionFails : boolean) : integer');
27284:  Function shellExecuteExCMDAndWaitklib(params: string; runAsAdmin: boolean; showWindowType:
TShowWindowType; exceptionIfReturnCodeIsNot0 : boolean) : LongInt');
27285:  Function shellExecuteExAndWaitklib( fileName: string; params: string; runAsAdmin : boolean;
showWindowType : TShowWindowType; exceptionIfReturnCodeIsNot0: boolean) : LongInt');
27286:  Function executeAndWaitExeklib(fileName:string;params:string;exceptionIfReturnCodeIsNot0:bool):LongInt');
27287:  Function netShareklib( targetDir : string; netName : string; netPassw : string;
grantAllPermissionToEveryoneGroup : boolean) : string');
27288:  Procedure addTCP_IN_FirewallExceptionklib(ruleName:string;port:Word;description:string;grouping:string;
executable:string)
27289:  Procedure deleteFirewallExceptionklib( ruleName : string);
27290:  // CL.AddTypeS('TExplicitAccess', 'EXPLICIT_ACCESS_A');
27291:  Procedure grantAllPermissionsNetToTheObjectForTheEveryoneGroupklib( myObject : string);
27292:  Procedure grantAllPermissionsNetToTheObjectForTheUsersGroupklib( myObject : string);
27293:  Procedure grantAllPermissionNetToTheObjectklib( windowsGroupOrUser : string; myObject : string);
27294:  Procedure grantAllPermissionsToTheObjectForTheEveryoneGroupklib( myObject : string);
27295:  Procedure grantAllPermissionsToTheObjectForTheUsersGroupklib( myObject : string);
27296:  Procedure grantAllPermissionsToTheObjectklib( windowsGroupOrUser : string; myObject : string);
27297:  Procedure grantAllPermissionsToTheObjectForTheEveryoneGroup2klib( myObject : string);
27298:  Procedure grantAllPermissionsToTheObjectForTheUsersGroup2klib( myObject : string);
27299:  Procedure grantAllPermissionsToTheObject2klib( windowsGroupOrUser : string; myObject : string);
27300:  Function checkIfWindowsGroupOrUserExistsklib( windowsGroupOrUser : string) : boolean');
27301:  Procedure createDesktopLinkklib( fileName : string; nameDesktopLink : string; description: string);
27302:  Function getDesktopDirPathklib : string');
27303:  Procedure copyDirIntoTargetDirklib( sourceDir : string; targetDir : string; forceOverwrite : boolean);
27304:  Procedure copyDirklib( sourceDir : string; destinationDir : string; silent : boolean);
27305:  Procedure createHideDirklib( dirName : string; forceDelete : boolean);
27306:  Procedure deleteDirectoryIfExistsklib( dirName : string; silent : boolean);
27307:  Procedure myMoveFileklib( sourceFileName : string; targetFileName : string);
27308:  Procedure createEmptyFileIfNotExistsklib( filename : string);
27309:  Procedure createEmptyFileklib( filename : string);
27310:  Function checkIfIsWindowsSubDirklib( subDir : string; mainDir : string) : boolean');
27311:  Function getParentDirklib( source : string) : string');
27312:  Function getValidFullPathInWindowsStyleklib( path : string) : string');
27313:  Function getPathInWindowsStyleklib( path : string) : string');
27314:  Function getStringWithEnvVariablesReadedklib( source : string) : string');
27315:  Function setProcessWindowToForegroundklib( processName : string) : boolean');
27316:  Function getPIDOfCurrentUserByProcessNameklib( nameProcess : string): DWORD');
27317:  Function getWindowsUsernameklib : string');
27318:  Function checkUserOfProcessklib( userName : String; PID : DWORD) : boolean');
27319:  Function getPIDCredentialsklib( PID : DWORD) : TPIDCredentials');
27320:  Function getPIDByProcessNameklib( nameProcess : string) : DWORD');
27321:  Function getMainWindowHandleByPIDklib( PID : DWORD) : DWORD');
27322:  Procedure closeApplicationklib( handle : THandle);
27323:  Function sendMemoryStreamUsing_WM_COPYDATAklib( handle : THandle; data : TMemoryStream) : integer');
27324:  Function sendStringUsing_WM_COPYDATAklib(handle:THandle; data:string; msgIdentifier: integer):integer');
27325:  Procedure mySetForegroundWindowklib( handle : THandle);
27326:  Function checkIfWindowExistsklib( className : string; captionForm : string) : boolean');
27327:  Function myFindWindowklib( className : string; captionForm : string) : THandle');
27328:  Function checkIfExistsKeyIn_HKEY_LOCAL_MACHINEklib( key : string) : boolean');
27329:  Procedure waitForMultipleklib( processHandle : THandle; timeout : DWORD; modalMode : boolean);
27330:  Procedure waitForklib( processHandle : THandle; timeout : DWORD; modalMode : boolean);
27331:  Procedure raiseLastSysErrorMessageklib');
27332:  Function getLastSysErrorMessageklib : string');
```

```
27333:  Function getLocaleDecimalSeparatorklib : char');
27334:  Procedure terminateCurrentProcessklib( exitCode : Cardinal; raiseExceptionEnabled : boolean);
27335:  Procedure myTerminateProcessklib(processHandle:THandle;exitCode:Cardinal;raiseExceptionEnabled: boolean);
27336:  //Function fixedGetNamedSecurityInfo( pObjectName : LPWSTR; ObjectType : SE_OBJECT_TYPE; SecurityInfo :
        SECURITY_INFORMATION; ppsidOwner, ppsidGroup : PPSID; ppDacl, ppSacl : PPACL; var ppSecurityDescriptor :
        PSECURITY_DESCRIPTOR) : DWORD');
27337: end;
27338:
27339: https://github.com/lonewolfonline/delphi-utilities/blob/master/AzuliaUtils.pas
27340:
27341: procedure SIRegister_TAzuliaHTML(CL: TPSPascalCompiler);
27342: begin
27343:   //with RegClassS(CL,'TComponent', 'TAzuliaHTML') do
27344:   with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliaHTML') do  begin
27345:     RegisterMethod('Function HeadTop( Title : string) : string');
27346:     Function HeadClose : string');
27347:     Function BodyTop(LeftMargin,RightMargin:Integer;TextColor,LinkColor,VisitedLinkColor,ActiveLinkColor,
        BackgroundColor,BackgroundImage:string;FixedBackground:bool;Extra:string):string');
27348:     Function BodyClose : string');
27349:     Function ConvertDOSPathToHTML( WindowsPath : string) : string');
27350:     Function ConvertHTMLPathToDOS( WindowsPath : string) : string');
27351:   end;
27352: end;
27353: (*----------------------------------------------------------------------------*)
27354: procedure SIRegister_TAzuliaDisk(CL: TPSPascalCompiler);
27355: begin
27356:   //with RegClassS(CL,'TComponent', 'TAzuliaDisk') do
27357:   with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliaDisk') do  begin
27358:     Function GetDiskLabel( Drive : char) : string');
27359:     Function GetDiskSerial( Drive : char) : string');
27360:     Function GetDiskFileSystem( Drive : char) : string');
27361:     Function GetDiskType( Drive : char) : integer');
27362:     Function GetDiskFlags( Drive : char) : integer');
27363:     Procedure ChangeDiskLabel( Drive : char; NewLabel : string);
27364:   end;
27365: end;
27366: (*----------------------------------------------------------------------------*)
27367: procedure SIRegister_TAzuliaSpeaker(CL: TPSPascalCompiler);
27368: begin
27369:   //with RegClassS(CL,'TComponent', 'TAzuliaSpeaker') do
27370:   with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliaSpeaker') do begin
27371:     Procedure Delay( MSecs : Integer);
27372:     Procedure Play( Freq : Word; MSecs : Integer);
27373:     Procedure Stop');
27374:   end;
27375: end;
27376: (*----------------------------------------------------------------------------*)
27377: procedure SIRegister_TAzuliaFiles(CL: TPSPascalCompiler);
27378: begin
27379:   //with RegClassS(CL,'TOBJECT', 'TAzuliaFiles') do
27380:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TAzuliaFiles') do begin
27381:     Procedure FindRecursive(const path: string;const mask:string;LogFunction:TLogFunct;Re_curse:Boolean);
27382:     Procedure SearchFileExt( const Dir, Ext : string; Files : TStrings);
27383:     Function GetFileSize( const FileName : string) : LongInt');
27384:     Function GetKBFileSize( FileSize : integer) : string');
27385:     Function LongToShortFilename( LongFilename : string) : string');
27386:     Function ShortToLongFilename( LongFilename : string) : string');
27387:     Procedure AppendDataToFilename( Filename, Data : string);
27388:     Procedure SaveDataToFilename( Filename, Data : string);
27389:     Procedure CopyFile( const FileName, DestName : string);
27390:     Procedure MoveFile( const FileName, DestName : string);
27391:     Function HasAttr( const FileName : string; Attr : Word) : Boolean');
27392:     Function ExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle');
27393:     Function GetWindowsDir : string');
27394:     Function GetTempDir : string');
27395:     Function GetINIString( Section, Ident, DefaultString, Filename : string): string');
27396:   end;
27397: end;
27398: (*----------------------------------------------------------------------------*)
27399: procedure SIRegister_TAzuliaStrings(CL: TPSPascalCompiler);
27400: begin
27401:   //with RegClassS(CL,'TOBJECT', 'TAzuliaStrings') do
27402:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TAzuliaStrings') do begin
27403:     Function NPos( C : Char; S : string; N : Byte) : Byte');
27404:     Function IntToRoman( Value : Longint) : string');
27405:     Function RomanToInt( const S : string) : Longint');
27406:     Function GetDirFromStr( DirectoryString : string; IndexFromRight : integer) : string');
27407:     Function RemoveChars( InputString : string; CharToRemove : Char) : string');
27408:     Function GetStringFromGarbage( InputString : string; Index : integer) : string');
27409:     Function SplitupWithGarbage( InputString : string; CommonSplitter : char) : string');
27410:     Function GetCount( Character : char; InputString : string) : integer');
27411:     Function RemoveTrailingChars( StringToRemoveFrom : string; Trailer : char) : string');
27412:   end;
27413: end;
27414: (*----------------------------------------------------------------------------*)
27415: procedure SIRegister_TSystemRegistry(CL: TPSPascalCompiler);
27416: begin
27417:   //with RegClassS(CL,'TOBJECT', 'TSystemRegistry') do
27418:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TSystemRegistry') do begin
```

```
27419:     Function GetStringFromRegistry( Root : integer; Key, Value : string) : string');
27420:     Function SaveStringToRegistry( Root : integer; Key, Value, Name : string) : bool');
27421:     Procedure ToggleBinaryValue( RootKey : integer; Path, Key : string);
27422:     Procedure DeleteValueInRegistry( RootKey : integer; Path, Name : string);
27423:     Procedure RenameValueInRegistry( RootKey : integer; Path, OldName, NewName : string);
27424:   end;
27425: end;
27426: (*----------------------------------------------------------------------------*)
27427: procedure SIRegister_TBrowseForFolderDialog(CL: TPSPascalCompiler);
27428: begin
27429:   //with RegClassS(CL,'TOBJECT', 'TBrowseForFolderDialog') do
27430:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TBrowseForFolderDialog') do  begin
27431:     RegisterProperty('Title', 'string', iptrw);
27432:     Function Execute : Boolean');
27433:     RegisterProperty('Path', 'string', iptr);
27434:   end;
27435: end;
27436: (*----------------------------------------------------------------------------*)
27437: procedure SIRegister_IShellLink(CL: TPSPascalCompiler);
27438: begin
27439:   //with RegClassS(CL,'TOBJECT', 'IShellLink') do
27440:   with CL.AddClassN(CL.FindClass('TOBJECT'),'IShellLink') do begin
27441:     Function GetPath(pszFile:LPSTR;cchMaxPath:integer; var pfd:TWin32FindData;fFlags:DWORD):HResult');
27442:     Function GetIDList( var ppidl : PITEMIDLIST) : HResult');
27443:     Function SetIDList( pidl : PITEMIDLIST) : HResult');
27444:     Function GetDescription( pszName : LPSTR; cchMaxName : integer) : HResult');
27445:     Function SetDescription( pszName : LPSTR) : HResult');
27446:     Function GetWorkingDirectory( pszDir : LPSTR; cchMaxPath : integer) : HResult');
27447:     Function SetWorkingDirectory( pszDir : LPSTR) : HResult');
27448:     Function GetArguments( pszArgs : LPSTR; cchMaxPath : integer) : HResult');
27449:     Function SetArguments( pszArgs : LPSTR) : HResult');
27450:     Function GetHotkey( var pwHotkey : word) : HResult');
27451:     Function SetHotkey( wHotkey : word) : HResult');
27452:     Function GetShowCmd( var piShowCmd : integer) : HResult');
27453:     Function SetShowCmd( iShowCmd : integer) : HResult');
27454:     Function GetIconLocation( pszIconPath LPSTR;cchIconPath:integer;var piIcon: integer): HResult');
27455:     Function SetIconLocation( pszIconPath : LPSTR; iIcon : integer) : HResult');
27456:     Function SetRelativePath( pszPathRel : LPSTR; dwReserved : DWORD) : HResult');
27457:     Function Resolve( Wnd : HWND; fFlags : DWORD) : HResult');
27458:     Function SetPath( pszFile : LPSTR) : HResult');
27459:   end;
27460: end;
27461: (*----------------------------------------------------------------------------*)
27462: procedure SIRegister_AzuliaUtils(CL: TPSPascalCompiler);
27463: begin
27464:   CL.AddTypeS('azucharSet', 'set of char');
27465:   SIRegister_IShellLink(CL);
27466:   SIRegister_TBrowseForFolderDialog(CL);
27467:   SIRegister_TSystemRegistry(CL);
27468:   SIRegister_TAzuliaStrings(CL);
27469:   CL.AddTypeS(TLogFunct', 'Function ( const path : string; const SRec : TSearchRec) : Boolean');
27470:   SIRegister_TAzuliaFiles(CL);
27471:   SIRegister_TAzuliaSpeaker(CL);
27472:   SIRegister_TAzuliaDisk(CL);
27473:   SIRegister_TAzuliaHTML(CL);
27474:   CL.AddTypeS('GrabType', '( GTSCREEN, GTWINDOW, GTCLIENT );
27475:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidDest');
27476:   CL.AddClassN(CL.FindClass('TOBJECT'),'EFCantMove');
27477:  Function azuBoolToStr( value : boolean) : string');
27478:  Procedure azuChangeWallpaper( FName : string; IsTiled, IsStretch : Boolean);
27479:  Function azuDoIExist( WndTitle : string) : Boolean');
27480:  Procedure azuAlert( Text : string);
27481:  Procedure azuTileImage( SourceImage, DestImage : TImage);
27482:  Function azuIncChar( iX : Char) : Char');
27483:  Function azuDecChar( iX : Char) : Char');
27484:  Procedure azuSaveForm( F : TForm; Filename : string);
27485:  Procedure azuLoadForm( F : TForm; Filename : string);
27486:  Function azuToggleBool( totoggle : bool) : bool');
27487:  Function azuremoveTrailingChars( const s : string; chars : azucharSet) : string');
27488:  Function azuremoveLeadChars( const s : string; chars : azucharSet) : string');
27489:  Function azuremoveChars( const s : string; chars : char) : string');
27490:  Procedure azuDelay( MSecs : Integer);
27491:  Function azuSHBrowseDialog( title : string; Handle : integer) : string');
27492:  Procedure azuregisterfiletype( Extention, REGDesc, UserDesc, Icon, Appl : string);
27493:  Procedure azuRemoveFileType( Extention, RegDesc : string);
27494:  Procedure azuGetFunctionNamesFromDLL( DLLName : string; List : TStrings);
27495:  Procedure azuSaveListViewToFile( AListView : TListView; sFileName : string);
27496:  Procedure azuLoadListViewToFile( AListView : TListView; sFileName : string);
27497:  Procedure azuLoadCSV( Filename : string; sg : TStringGrid);
27498:  Function azuJPEGDimensions( Filename : string; var X, Y : Word) : boolean');
27499:  Procedure azuResizeImage( FileName : string; MaxWidth : Integer);
27500:  Function azuSaveJPEGPictureFile(Bitmap:TBitmap;FilePath,FileName:string;Quality:Integer): Boolean');
27501:  Function azuLoadJPEGPictureFile( Bitmap : TBitmap; FilePath, FileName : string) : Boolean');
27502:  Procedure azuSmoothResize( Src, Dst : TBitmap);
27503:  Function azuMsecToStr( Milli : Cardinal) : string');
27504:  Procedure azuDirToStrings( APath : string; AStrings : TStrings; WithExt, WithPath : boolean);
27505:  Procedure azuSetWallPaper( ImageFileName : string; IsTiled : boolean);
27506:  Function azuGetFileSize( FileName : string) : int64');
27507:  Function azuGetExeByExtension( sExt : string) : string');
```

```
27508:  Function azuRefreshScreenIcons : Boolean');
27509:  Function azuIndexToColor( AIndex : integer) : TColor');
27510:  Function azuColorToIndex( AColor : TColor) : integer');
27511:  Function azuTitleCase( Text2 : string) : string');
27512:  Function azuToggleCase( Text2 : string) : string');
27513:  Function azuSentenceCase( Text2 : string) : string');
27514:  Procedure azuGrabScreen( bm : TBitMap; gt : GrabType);
27515:  Procedure azuSaveStringGrid( StringGrid : TStringGrid; FileName : String);
27516:  Procedure azuLoadStringGrid( StringGrid : TStringGrid; FileName : String);
27517:  Function azuGetInetFile( const fileURL, FileName : String) : boolean');
27518:  Procedure azuRunOnStartup( sProgTitle, sCmdLine : string; bRunOnce : boolean);
27519:  Procedure azuRemoveOnStartup( sProgTitle : string);
27520:  CL.AddConstantN('SInvalidDest','String').SetString( 'Destination %s does not exist');
27521:  SFCantMove','String').SetString( 'Cannot move file %s');
27522:  AzuMsg1','String').SetString( 'File "%s" does not exist!');
27523:  AzuMsg2','String').SetString( '"%s" is not a ListView file!');
27524:  sr_WindowMetrics','String').SetString( 'Control Panel\Desktop\WindowMetrics\');
27525:  sr_ShellIconSize','String').SetString( 'Shell Icon Size');
27526: end;
27527:
27528: procedure SIRegister_TALWinInetHTTPClient2(CL: TPSPascalCompiler);
27529: begin
27530:   //with RegClassS(CL,'TALHTTPClient2', 'TALWinInetHTTPClient2') do
27531:   with CL.AddClassN(CL.FindClass('TALHTTPClient2'),'TALWinInetHTTPClient2') do begin
27532:     RegisterMethod('Constructor Create');
27533:     Procedure Free');
27534:     Procedure Connect');
27535:     Procedure Disconnect');
27536:     AccessType', 'TALWinInetHttpInternetOpenAccessType', iptrw);
27537:     InternetOptions', 'TAlWininetHTTPClientInternetOptionSet', iptrw);
27538:     OnStatus', 'TAlWinInetHTTPClientStatusEvent', iptrw);
27539:     IgnoreSecurityErrors', 'Boolean', iptrw);
27540:   end;
27541: end;
27542:
27543: procedure SIRegister_TALHTTPClient2(CL: TPSPascalCompiler);
27544: begin
27545:   //with RegClassS(CL,'TObject', 'TALHTTPClient') do
27546:   with CL.AddClassN(CL.FindClass('TObject'),'TALHTTPClient2') do begin
27547:     Constructor Create');
27548:     Procedure Free');
27549:     Procedure Get(const aUrl:AnsiString;const aRequestFields: TALStrings;const aResponseContent:TStream;
      const aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray;const
      aEncodeRequestFields:Bool);');
27550:     Procedure Get1( const aUrl: AnsiString; const aResponseContent : TStream; const aResponseHeader :
      TALHTTPResponseHeader2; const ARequestHeaderValues : TALNameValueArray);');
27551:     Function Get2(const aUrl: AnsiString; const ARequestHeaderValues : TALNameValueArray) : AnsiString;');
27552:     Function Get3(const aUrl: AnsiString;const aRequestFields:TALStrings;const ARequestHeaderValues:
      TALNameValueArray;const aEncodeRequestFields:Boolean): AnsiString;');
27553:     Procedure Post(const aUrl:AnsiString; const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);');
27554:     Procedure Post1(const aUrl: AnsiString; const aPostDataStream: TStream; const aResponseContent:
      TStream; const aResponseHeader:TALHTTPResponseHeader2; const ARequestHeaderValues: TALNameValueArray);');
27555:     Function Post2(const aUrl: AnsiString; const ARequestHeaderValues: TALNameValueArray) : AnsiString;');
27556:     Function Post3(const aUrl:AnsiString;const aPostDataStream:TStream;const ARequestHeaderValues:
      TALNameValueArray):AnsiString;');
27557:     Procedure PostUrlEncoded(const aUrl:AnsiString;const aRequestFields:TALStrings;const aResponseContent
      : TStream; const aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray;const
      aEncodeRequestFields:Boolean);
27558:     Function PostUrlEncoded1(const aUrl:AnsiString;const aRequestFields:TALStrings;const
      ARequestHeaderValues : TALNameValueArray;const aEncodeRequestFields:Boolean):AnsiString;');
27559:     Procedure PostMultipartFormData(const aUrl: AnsiString; const aRequestFields : TALStrings; const
      aRequestFiles : TALMultiPartFormDataContents; const aResponseContent : TStream; const aResponseHeader' +
27560:       ' : TALHTTPResponseHeader2; const ARequestHeaderValues : TALNameValueArray);');
27561:     Function PostMultiPartFormData1(const aUrl: AnsiString; const aRequestFields: TALStrings; const
      aRequestFiles:TALMultiPartFormDataContents;const ARequestHeaderValues: TALNameValueArray): AnsiString;');
27562:     Procedure Head(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);');
27563:     Function Head1( const aUrl: AnsiString; const ARequestHeaderValues: TALNameValueArray) : AnsiString;');
27564:     Procedure Trace(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);');
27565:     Function trace1( const aUrl : AnsiString; const ARequestHeaderValues : TALNameValueArray) :
      AnsiString;');
27566:     Procedure Put(const aUrl:AnsiString;const aPutDataStream:TStream;const aResponseContent:TStream; const
      aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues: TALNameValueArray;');
27567:     Function Put1(const aURL:Ansistring;const aPutDataStream:TStream;const ARequestHeaderValues:
      TALNameValueArray):AnsiString;;
27568:     Procedure Delete(const aUrl: AnsiString;const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues : TALNameValueArray);');
27569:     Function Delete1(const aURL:Ansistring;const ARequestHeaderValues: TALNameValueArray): AnsiString;');
27570:     Procedure Options( const aUrl : AnsiString; const aResponseContent : TStream; const aResponseHeader :
      TALHTTPResponseHeader2; const ARequestHeaderValues : TALNameValueArray);');
27571:     Function Options1(const aURL:Ansistring; const ARequestHeaderValues:TALNameValueArray) : AnsiString;');
27572:     RegisterProperty('ConnectTimeout', 'Integer', iptrw);
27573:     SendTimeout', 'Integer', iptrw);
27574:     ReceiveTimeout', 'Integer', iptrw);
27575:     UploadBufferSize', 'cardinal', iptrw);
27576:     ProxyParams', 'TALHTTPClientProxyParams', iptr);
27577:     RequestHeader', 'TALHTTPRequestHeader2', iptr);
```

```
27578:     ProtocolVersion', 'TALHTTPProtocolVersion', iptrw);
27579:     UserName', 'AnsiString', iptrw);
27580:     Password', 'AnsiString', iptrw);
27581:     OnUploadProgress', 'TALHTTPClientUploadProgressEvent', iptrw);
27582:     OnDownloadProgress', 'TALHTTPClientDownloadProgressEvent', iptrw);
27583:     OnRedirect', 'TAlHTTPClientRedirectEvent', iptrw);
27584:   end;
27585: end;
27586:
27587: procedure SIRegister_ALHttpClient2(CL: TPSPascalCompiler);
27588: begin
27589:   TALHTTPPropertyChangeEvent', 'Procedure ( sender: Tobject; const PropertyIndex: Integer)');
27590:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )');
27591:   TALHTTPMethod2', '( HTTPmt_Get, HTTPmt_Post, HTTPmt_Head, HTTPmt_'
27592:                      +'Trace, HTTPmt_Put, HTTPmt_Delete, HTTPmt_Options )');
27593:   SIRegister_TALHTTPRequestHeader2(CL);
27594:   TALNameValuePair', 'record Name : ansistring; Value : ansistring; end');
27595:   TALNameValueArray', 'array of TALNameValuePair');
27596:   SIRegister_TALHTTPCookie2(CL);
27597:   SIRegister_TALHTTPCookieCollection2(CL);
27598:   SIRegister_TALHTTPResponseHeader2(CL);
27599:   SIRegister_EALHTTPClientException2(CL);
27600:   SIRegister_TALHTTPClientProxyParams2(CL);
27601:   TAlHTTPClientRedirectEvent', 'Procedure ( sender : Tobject; const NewURL : AnsiString)');
27602:   TALHTTPClientUploadProgressEvent', 'Procedure ( sender : Tobject; Sent: Integer; Total: Integer)');
27603:   TALHTTPClientDownloadProgressEvent', 'Procedure ( sender: Tobject; Read: Integer; Total: Integer)');
27604:   SIRegister_TALHTTPClient2(CL);
27605:  Procedure ALHTTPEncodeParamNameValues2( const ParamValues : TALStrings)');
27606:  //Procedure ALExtractHTTPFields( Separators,WhiteSpace,Quotes: TSysCharSet; Content:PAnsiChar; Strings :
       TALStrings; StripQuotes : Boolean)');
27607:  Function AlRemoveShemeFromUrl2( const aUrl : AnsiString) : ansiString');
27608:  Function AlExtractShemeFromUrl2( const aUrl : AnsiString) : TInternetScheme');
27609:  Function AlExtractHostNameFromUrl2( const aUrl : AnsiString) : AnsiString');
27610:  Function AlExtractDomainNameFromUrl2( const aUrl : AnsiString) : AnsiString');
27611:  Function AlExtractUrlPathFromUrl2( const aUrl : AnsiString) : AnsiString');
27612:  Function AlInternetCrackUrl2(const aUrl:AnsiString;var SchemeName,HostName,UserName,Password,UrlPath,
       ExtraInfo : AnsiString;var PortNumber:integer):Bool;
27613:  Function AlInternetCrackUrl12(const aUrl:AnsiString;var SchemeName,HostName,UserName,Password,UrlPath,
       Anchor: AnsiString;const Query:TALStrings;var PortNumber:integer): Boolean;
27614:  Function AlInternetCrackUrl22( var Url:AnsiString;var Anchor:AnsiString;const Query:TALStrings):
       Boolean;');
27615:  Function AlRemoveAnchorFromUrl2( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;');
27616:  Function AlRemoveAnchorFromUrl12( const aUrl : AnsiString) : AnsiString;');
27617:  Function AlCombineUrl2( const RelativeUrl, BaseUrl : AnsiString) : AnsiString;');
27618:  Function AlCombineUrl12( const RelativeUrl,BaseUrl,Anchor AnsiString;const Query:TALStrings):
       AnsiString;');
27619:  Function ALGmtDateTimeToRfc822Str2( const aValue : TDateTime) : AnsiString');
27620:  Function ALDateTimeToRfc822Str2( const aValue : TDateTime) : AnsiString');
27621:  Function ALTryRfc822StrToGMTDateTime2( const S : AnsiString; out Value : TDateTime) : Boolean');
27622:  Function ALRfc822StrToGMTDateTime2( const s : AnsiString) : TDateTime');
27623:  Function ALTryIPV4StrToNumeric2( const aIPv4Str : AnsiString; var aIPv4Num : Cardinal) : Boolean');
27624:  Function ALIPV4StrToNumeric2( const aIPv4 : AnsiString) : Cardinal');
27625:  Function ALNumericToIPv4Str2( const aIPv4 : Cardinal) : ansiString');
27626:  Function ALIPv4EndOfRange2( const aStartIPv4 : Cardinal; aMaskLength : integer) : Cardinal');
27627:  Function ALZeroIpV62 : TALIPv6Binary');
27628:  Function ALIsValidIPv6BinaryStr2( const aIPV6BinaryStr : ansiString) : boolean');
27629:  Function ALTryIPV6StrToBinary2( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean');
27630:  Function ALIPV6StrTobinary2( const aIPv6 : AnsiString) : TALIPv6Binary');
27631:  Function ALBinaryToIPv6Str2( const aIPv6 : TALIPv6Binary) : ansiString');
27632:  Function ALBinaryStrToIPv6Binary2( const aIPV6BinaryStr : ansiString) : TALIPv6Binary');
27633:  Function ALBinaryStrToIPv6Str2( const aIPV6BinaryStr : ansiString) : ansiString');
27634:  Function ALIPv6EndOfRange2( const aStartIPv6 : TALIPv6Binary; aMaskLength : integer) : TALIPv6Binary');
27635:  Procedure ALIPv6SplitParts2(const aIPv6: TALIPv6Binary;var aLowestPart:UInt64; var aHigestPart: UInt64)');
27636:  CL.AddConstantN('cALHTTPCLient_MsgInvalidURL2','String').SetString('Invalid url ''%s'' - only supports
       ''http'' and ''https'' schemes');
27637:  CL.AddConstantN('cALHTTPCLient_MsgInvalidHTTPRequest2','String').SetString('Invalid HTTP Request:Length
       is 0');
27638:  CL.AddConstantN('cALHTTPCLient_MsgEmptyURL2','String').SetString( 'Empty URL');
27639:  function ALHTTPEncode(const AStr: AnsiString): AnsiString;');
27640:  function ALHTTPDecode(const AStr: AnsiString): AnsiString;');
27641: end;
27642:
27643: CL.AddDelphiFunction('procedure RunDosInMemo(DosApp: string; AMemo:TMemo);');
27644:
27645: {type TTextHandler =} procedure TTextHandlerQ(const aText: string);
27646: begin
27647:   memo2.lines.add(atext);
27648: end;
27649: writeln(itoa(JExecute('cmd /C dir *.*',@TTextHandlerQ, true, false)));
27650: writeln(itoa(JExecute3('cmd /C dir *.*',Nil, true, false)));
27651:
27652: memo2.lines.add(getIPs.text);
27653:    srlist:= TStringlist.create;
27654:    GetDNSServers(srlist)
27655:    writeln(srlist.text)
27656:    srlist.Free;
27657:
27658: https://presse.tuv.com/en/photovoltaics-tuv-rheinland-investigates-potential-for-applications-on-rail-
       infrastructure/
```

```
27659:
27660: procedure SIRegister_TRestUtils(CL: TPSPascalCompiler);
27661: begin
27662:   //with RegClassS(CL,'TOBJECT', 'TRestUtils') do
27663:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TRestUtils') do begin
27664:     Function Base64Encode( const AValue : String) : String');
27665:     Function Base64Decode( const AValue : String) : String');
27666:   end;
27667: end;
27668:
27669: (*----------------------------------------------------------------------------*)
27670: procedure SIRegister_TStatusCode(CL: TPSPascalCompiler);
27671: begin
27672:   //with RegClassS(CL,'TOBJECT', 'TStatusCode') do
27673:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TStatusCode') do begin
27674:     Function CONTINUE : TReponseCode');
27675:     Function SWITCHING_PROTOCOLS : TReponseCode');
27676:     Function OK : TReponseCode');
27677:     Function CREATED : TReponseCode');
27678:     Function ACCEPTED : TReponseCode');
27679:     Function NON_AUTHORITATIVE_INFORMATION : TReponseCode');
27680:     Function NO_CONTENT : TReponseCode');
27681:     Function RESET_CONTENT : TReponseCode');
27682:     Function PARTIAL_CONTENT : TReponseCode');
27683:     Function MULTIPLE_CHOICES : TReponseCode');
27684:     Function MOVED_PERMANENTLY : TReponseCode');
27685:     Function FOUND : TReponseCode');
27686:     Function SEE_OTHER : TReponseCode');
27687:     Function NOT_MODIFIED : TReponseCode');
27688:     Function USE_PROXY : TReponseCode');
27689:     Function TEMPORARY_REDIRECT : TReponseCode');
27690:     Function BAD_REQUEST : TReponseCode');
27691:     Function UNAUTHORIZED : TReponseCode');
27692:     Function PAYMENT_REQUIRED : TReponseCode');
27693:     Function FORBIDDEN : TReponseCode');
27694:     Function NOT_FOUND : TReponseCode');
27695:     Function METHOD_NOT_ALLOWED : TReponseCode');
27696:     Function NOT_ACCEPTABLE : TReponseCode');
27697:     Function PROXY_AUTHENTICATION_REQUIRED : TReponseCode');
27698:     Function REQUEST_TIMEOUT : TReponseCode');
27699:     Function CONFLICT : TReponseCode');
27700:     Function GONE : TReponseCode');
27701:     Function LENGTH_REQUIRED : TReponseCode');
27702:     Function PRECONDITION_FAILED : TReponseCode');
27703:     Function REQUEST_ENTITY_TOO_LARGE : TReponseCode');
27704:     Function REQUEST_URI_TOO_LONG : TReponseCode');
27705:     Function UNSUPPORTED_MEDIA_TYPE : TReponseCode');
27706:     Function REQUESTED_RANGE_NOT_SATISFIABLE : TReponseCode');
27707:     Function EXPECTATION_FAILED : TReponseCode');
27708:     Function UNPROCESSABLE_ENTITY : TReponseCode');
27709:     Function INTERNAL_SERVER_ERROR : TReponseCode');
27710:     Function NOT_IMPLEMENTED : TReponseCode');
27711:     Function BAD_GATEWAY : TReponseCode');
27712:     Function SERVICE_UNAVAILABLE : TReponseCode');
27713:     Function GATEWAY_TIMEOUT : TReponseCode');
27714:     Function HTTP_VERSION_NOT_SUPPORTED : TReponseCode');
27715:   end;
27716: end;
27717:
27718: (*----------------------------------------------------------------------------*)
27719: procedure SIRegister_RestUtils(CL: TPSPascalCompiler);
27720: begin
27721:  MediaType_Json','String').SetString( 'application/json');
27722:  MediaType_Xml','String').SetString( 'text/xml');
27723:  LOCALE_PORTUGUESE_BRAZILIAN','String').SetString( 'pt-BR');
27724:  LOCALE_US','String').SetString( 'en-US');
27725:   CL.AddTypeS('TReponseCode', 'record StatusCode : Integer; Reason : string; end');
27726:   SIRegister_TStatusCode(CL);
27727:   SIRegister_TRestUtils(CL);
27728: end;
27729:
27730: procedure SIRegister_DataSetConverter4DUtil(CL: TPSPascalCompiler);
27731: begin
27732:  CL.AddTypeS('TBooleanFieldType', '( bftUnknown, bftBoolean, bftInteger )');
27733:  CL.AddTypeS('TDataSetFieldType', '( dftUnknown, dftJSONObject, dftJSONArray )');
27734:  CL.AddDelphiFunction('Function DateTimeToISOTimeStamp( const dateTime : TDateTime) : string');
27735:  Function DateToISODate( const date : TDateTime) : string');
27736:  Function TimeToISOTime( const time : TTime) : string');
27737:  Function ISOTimeStampToDateTime( const dateTime : string) : TDateTime');
27738:  Function ISODateToDate( const date : string) : TDate');
27739:  Function ISOTimeToTime( const time : string) : TTime');
27740:  Function NewDataSetField(dataSet:TDataSet;const fieldType:TFieldType;const fieldName:string;const
       size:Integer; const origin:string;const displaylabel:string): TField');
27741:  Function BooleanToJSON( const value : Boolean) : TJSONValue');
27742:  Function BooleanFieldToType( const booleanField : TBooleanField) : TBooleanFieldType');
27743:  Function DataSetFieldToType( const dataSetField : TDataSetField) : TDataSetFieldType');
27744:  Function MakeValidIdent( const s : string) : string');
27745: end;
27746:
```

```
27747:  function wget3(aURL, afile: string; opendoc: boolean): boolean;
27748:  procedure RegisterProtocol(const Name, Describtion, ExecuteStr: string)');
27749:  procedure UnregisterProtocol(const Name: string);');
27750:  function GetNetworkConnections: string;
27751:  function GetNetworkDrives: string;
27752:
27753: wget3('https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Ffoodanddrink.scotsman.com%2Fwp-
       content%2Fuploads%2F2016%2F08%2Fitem-4-9-1024x683.jpg',
27754:            exepath+'examples/downpic.png', true);
27755:
27756: procedure SIRegister_PSResources(CL: TPSPascalCompiler);
27757: begin
27758:  CL.AddConstantN('_sBeginCompile','String').SetString( 'Compiling');
27759:  ('_sSuccessfullyCompiled','String').SetString( 'Succesfully compiled');
27760:  ('_sSuccessfullyExecuted','String').SetString( 'Succesfully executed');
27761:  ('_sRuntimeError','String').SetString( '[Runtime error] %s(%d:%d), bytecode(%d:%d): %s');
27762:  ('_sTextNotFound','String').SetString( 'Text not found');
27763:  ('_sUnnamed','String').SetString( 'Unnamed');
27764:  ('_sEditorTitle','String').SetString( 'Editor');
27765:  ('_sEditorTitleRunning','String').SetString( 'Editor - Running');
27766:  ('_sEditorTitlePaused','String').SetString( 'Editor - Paused');
27767:  ('_sEditorTitleStopped','String').SetString( 'Editor - Stopped');
27768:  ('_sInputBoxTiyle','String').SetString( 'Script');
27769:  ('_sFileNotSaved','String').SetString( 'File has not been saved, save now?');
27770:  ('_isRunningOrPaused','LongInt').Value.ts32 := ord(isRunning) or ord(isPaused);
27771:   SIRegister_TPSUtils(CL);
27772:   //https://github.com/fabriciocolombo/PascalScriptEx/tree/master/src
27773:   CL.AddDelphiFunction('function list_functions: TStringlist;');
27774:   function list_functions2: TStringlist;');
27775:   procedure InternalRaiseException(E: Exception);');
27776:   function ListFiles(ADirectory, AFilter: String; AList: TStrings): Boolean;');
27777:   Procedure CommandToList(S : String; List : TStrings);');
27778:  end;
27779:
27780: CL.AddDelphiFunction('function list_functions: TStringlist;');
27781: procedure InternalRaiseException(E: Exception);');
27782: function ListFiles(ADirectory, AFilter: String; AList: TStrings) : Boolean;
27783: Procedure CommandToList(S : String; List : TStrings);
27784: CL.AddDelphiFunction('procedure SetSynchroTime');
27785: CL.AddDelphiFunction('procedure letSynchroTime');
27786: procedure TimeSync');    //get internettime service
27787: function HttpGetDirect(const Url: string): string;
27788: function HttpGetDirect2(const Url: string): string;
27789: function HttpGetDirect4(const Url, agent: string): string;
27790: function HttpGet_Direct(const Url: string): string;');
27791: function wininet_HttpGetDirect(const Url: string): string;');
27792: function GetWinInetError(ErrorCode:Cardinal): string;
27793: function HttpGetDirect3(const ServerName,Resource,sUserAgent:string;Var Response:AnsiString):Integer;
27794: procedure WinInetCheck(Success: Boolean; aFunction: PChar);
27795: function WebGetData(const UserAgent: string; const Server: string; const Resource: string):string;
27796: procedure WebPostData(const UserAgent: string; const Server: string;
27797:                         const Resource: string; const Data: AnsiString);
27798: function URLEncode3(const Url: string): string;
27799:
27800: function FloatAsInteger(X: single): integer;
27801: begin
27802:   result:= PInteger(@X)^;
27803: end;
27804: function IntegerAsFloat(X: integer): single;
27805: begin
27806:   result:= PSingle(@X)^;
27807: end;
27808:
27809: procedure ParseString4(s,sep: string; sl: TStrings);');
27810: procedure LoadGridFromfile(SG: TStringGrid; fname: string);
27811: procedure SaveGridTofile(SG: TStringGrid; fname: string);
27812: procedure WriteVectorToGrid(SG: TStringGrid; vname: string; wval: TAffineVector);
27813: procedure WriteStringToGrid(SG: TStringGrid; vname: string; icol: longword; wval: string);
27814: procedure WriteFloatToGrid(SG: TStringGrid; vname: string; icol: longword; wval: double);
27815: https://github.com/P33a/SimTwo/blob/master/Utils.pas
27816:
27817: Recompiled 4.7.6.20
27818:
27819: 07/10/2022  21:16          16,829 FindReplDlg.dcu
27820: 07/10/2022  20:53         483,995 fMain.dcu
27821: 07/10/2022  21:16          35,172 gsUtils.dcu
27822: 07/10/2022  21:16         194,742 IFSI_WinForm1puzzle.dcu
27823: 07/10/2022  21:16         145,651 uPSI_fMain.dcu
27824: 07/10/2022  21:16          10,776 uPSI_IdPOP3.dcu
27825: 07/10/2022  21:16           5,649 uPSI_JvParsing.dcu
27826: 07/10/2022  21:16          21,122 uPSI_uTPLb_AES.dcu
27827: 07/10/2022  21:16          25,457 VListView.dcu
27828: 07/10/2022  21:16          70,846 WinForm1.dcu
27829: 07/10/2022  17:59          82,034 uPSI_Chart.dcu
27830: 07/10/2022  21:16          11,936 uPSI_SynEditMiscProcs.dcu
27831: 07/10/2022  11:08          10,972 uPSI_SynHighlighterPas.dcu
27832: 07/10/2022  17:44         100,843 uPSI_TeCanvas.dcu
27833: 07/10/2022  18:28         177,508 uPSI_TeEngine.dcu
27834: 07/10/2022  17:52          74,924 uPSI_TeeProcs.dcu
```

```
27835: 07/10/2022  21:16          105,297 AdoMain.dcu
27836: 07/10/2022  21:16           71,777 ALHttpClient2.dcu
27837: 07/10/2022  21:16           13,508 BlocksUnit.dcu
27838: 07/10/2022  21:16           11,799 frmExportMain.dcu
27839: 07/10/2022  21:16           30,871 MathsLib.dcu
27840: 07/10/2022  21:16           37,441 neuraldatasets.dcu
27841: 07/10/2022  21:16           50,133 neuralfit.dcu
27842: 07/10/2022  21:16          317,297 neuralnetworkCAI.dcu
27843: 07/10/2022  21:16          114,799 neuralvolume.dcu
27844: 07/10/2022  21:16           11,119 PSResources.dcu
27845: 07/10/2022  21:16            6,569 PXLTiming.dcu
27846: 07/10/2022  21:16            7,434 PythonAction.dcu
27847: 07/10/2022  21:16          278,943 PythonEngine.dcu
27848: 07/10/2022  21:16            6,447 SeSHA256.dcu
27849: 07/10/2022  21:16            3,879 simplecomport.dcu
27850: 07/10/2022  21:16           11,331 U_Splines.dcu
27851: 07/10/2022  21:16           66,636 uPSI_ALHttpClient2.dcu
27852: 07/10/2022  21:16           20,494 uPSI_BlocksUnit.dcu
27853: 07/10/2022  21:16           16,019 uPSI_DFFUtils.dcu
27854: 07/10/2022  11:08           12,554 uPSI_frmExportMain.dcu
27855: 07/10/2022  21:16           25,351 uPSI_JclSysUtils.dcu
27856: 07/10/2022  21:16           24,307 uPSI_neuraldatasets.dcu
27857: 07/10/2022  20:55            6,188 uPSI_PSResources.dcu
27858: 07/10/2022  21:16            8,644 uPSI_PXLTiming.dcu
27859: 07/10/2022  21:16          175,125 uPSI_PythonEngine.dcu
27860: 07/10/2022  21:16           46,231 VarPyth.dcu
27861: 07/10/2022  21:16           19,484 ViewToDoFm.dcu
27862: 07/10/2022  11:08           28,736 uconvMain.dcu
27863: 07/10/2022  16:52           63,798 uPSI_Series.dcu
27864: 07/10/2022  11:08           19,799 uPSI_uconvMain.dcu
27865: 07/10/2022  11:45           47,688 SynHighlighterPas.dcu
27866:
27867: 08/10/2022  00:43          484,364 fMain.dcu
27868: 08/10/2022  13:00           82,689 uPSI_Chart.dcu
27869: 08/10/2022  12:22          102,809 uPSI_TeCanvas.dcu
27870: 08/10/2022  12:46           75,065 uPSI_TeeProcs.dcu
27871: 08/10/2022  00:44           71,779 ALHttpClient2.dcu
27872: 08/10/2022  00:44           13,517 BlocksUnit.dcu
27873: 08/10/2022  16:05           30,866 MathsLib.dcu
27874: 08/10/2022  14:53           19,247 process.dcu
27875: 08/10/2022  16:03           13,555 PSResources.dcu
27876: 08/10/2022  00:44            6,568 PXLTiming.dcu
27877: 08/10/2022  00:44            7,433 PythonAction.dcu
27878: 08/10/2022  00:44          278,838 PythonEngine.dcu
27879: 08/10/2022  00:44            6,446 SeSHA256.dcu
27880: 08/10/2022  00:44           66,636 uPSI_ALHttpClient2.dcu
27881: 08/10/2022  00:33           37,434 uPSI_AzuliaUtils.dcu
27882: 08/10/2022  00:44           20,494 uPSI_BlocksUnit.dcu
27883: 08/10/2022  14:50           17,956 uPSI_process.dcu
27884: 08/10/2022  14:53            6,869 uPSI_PSResources.dcu
27885: 08/10/2022  00:44            8,643 uPSI_PXLTiming.dcu
27886: 08/10/2022  00:44          175,123 uPSI_PythonEngine.dcu
27887: 08/10/2022  00:44           46,252 VarPyth.dcu
27888: 08/10/2022  14:50           22,801 W32VersionInfo.dcu
27889:
27890: 09/10/2022  16:35          211,349 IFSI_SysUtils_max.dcu
27891: 09/10/2022  16:54          195,259 IFSI_WinForm1puzzle.dcu
27892: 09/10/2022  19:16           30,861 MathsLib.dcu
27893: 09/10/2022  19:14           14,498 PSResources.dcu
27894: 09/10/2022  17:54           48,542 uPSI_AzuliaUtils.dcu
27895: 09/10/2022  18:12            7,017 uPSI_PSResources.dcu
27896:
27897: 13/10/2022  16:31           82,689 uPSI_Chart.dcu
27898: 13/10/2022  18:24          178,625 uPSI_TeEngine.dcu
27899: 13/10/2022  16:36           75,069 uPSI_TeeProcs.dcu
27900: 13/10/2022  19:00           30,862 MathsLib.dcu
27901: 13/10/2022  17:01           49,467 uPSI_AzuliaUtils.dcu
27902:
27903: 15/10/2022  12:08          348,647 TeEngine.dcu
27904: 15/10/2022  12:05          179,899 uPSI_TeEngine.dcu
27905: 15/10/2022  17:57           30,859 MathsLib.dcu
27906: 15/10/2022  17:53           59,779 neuralfit.dcu
27907: 15/10/2022  13:45           53,663 uPSI_AzuliaUtils.dcu
27908: 15/10/2022  16:33           29,161 uPSI_neuralfit.dcu
27909:
27910: 16/10/2022  17:27          484,338 fMain.dcu
27911: 16/10/2022  17:30           71,766 ALHttpClient2.dcu
27912: 16/10/2022  17:30           13,517 BlocksUnit.dcu
27913: 16/10/2022  17:25           59,939 neuralfit.dcu
27914: 16/10/2022  17:30            6,566 PXLTiming.dcu
27915: 16/10/2022  17:30            7,427 PythonAction.dcu
27916: 16/10/2022  17:30          278,951 PythonEngine.dcu
27917: 16/10/2022  17:30            6,446 SeSHA256.dcu
27918: 16/10/2022  17:30           66,636 uPSI_ALHttpClient2.dcu
27919: 16/10/2022  17:30           20,495 uPSI_BlocksUnit.dcu
27920: 16/10/2022  17:30            8,645 uPSI_PXLTiming.dcu
27921: 16/10/2022  17:30          175,131 uPSI_PythonEngine.dcu
27922: 16/10/2022  17:30           46,237 VarPyth.dcu
27923:
```

```
27924: 17/10/2022  11:11            49,666 JclNTFS2.dcu
27925: 17/10/2022  11:27            30,846 MathsLib.dcu
27926: 17/10/2022  11:26            56,533 uPSI_AzuliaUtils.dcu
27927: 17/10/2022  11:16            47,572 uPSI_JclNTFS2.dcu
27928:
27929: C:\maXbox>dir /s /o N *.dcu | find "23/09/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
27930: 23/09/2022  16:15           483,806 fMain.dcu
27931: 23/09/2022  10:00           194,750 IFSI_WinForm1puzzle.dcu
27932: 23/09/2022  12:43            81,932 uPSI_Chart.dcu
27933: 23/09/2022  11:39            97,061 uPSI_TeCanvas.dcu
27934: 23/09/2022  11:39           177,404 uPSI_TeEngine.dcu
27935: 23/09/2022  11:20            73,888 uPSI_TeeProcs.dcu
27936: 23/09/2022  16:16            71,782 ALHttpClient2.dcu
27937: 23/09/2022  16:16            13,522 BlocksUnit.dcu
27938: 23/09/2022  17:45            30,862 MathsLib.dcu
27939: 23/09/2022  16:16             6,568 PXLTiming.dcu
27940: 23/09/2022  16:16             7,434 PythonAction.dcu
27941: 23/09/2022  16:16           278,867 PythonEngine.dcu
27942: 23/09/2022  16:06            10,517 RestUtils.dcu
27943: 23/09/2022  16:16             6,446 SeSHA256.dcu
27944: 23/09/2022  16:16            66,632 uPSI_ALHttpClient2.dcu
27945: 23/09/2022  17:45            31,846 uPSI_AzuliaUtils.dcu
27946: 23/09/2022  16:16            20,494 uPSI_BlocksUnit.dcu
27947: 23/09/2022  16:16             8,642 uPSI_PXLTiming.dcu
27948: 23/09/2022  16:16           175,123 uPSI_PythonEngine.dcu
27949: 23/09/2022  16:15            11,240 uPSI_RestUtils.dcu
27950: 23/09/2022  16:16            46,235 VarPyth.dcu
27951: 23/09/2022  15:11            63,422 uPSI_Series.dcu
27952:
27953: ****************************************************************
27954: Release Notes maXbox 4.7.6.20 Oct 2022 mX476
27955: ****************************************************************
27956: Add 62 Units + 18 Tutorials
27957:
27958: 1441 unit uPSI_neuralgeneric.pas; CAI
27959: 1442 unit uPSI_neuralthread.pas; CAI
27960: 1443 unit uPSI_uSysTools; TuO
27961: 1444 unit upsi_neuralsets; mX4
27962: 1445 unit uPSI_uWinNT.pas mX4
27963: 1446 unit uPSI_URungeKutta4.pas ICS
27964: 1447 unit uPSI_UrlConIcs.pas ICS
27965: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
27966: 1449 unit uPSI_Numedit2 mX4
27967: 1450 unit uPSI_PsAPI_3.pas mX4
27968: 1451 unit uPSI_SeSHA256.pas
27969: 1452 unit IdHashMessageDigest_max2;
27970: 1453 unit uPSI_BlocksUnit.pas
27971: 1454 unit uPSI_DelticsCommandLine.pas
27972: 1455 unit uPSI_DelticsStrUtils;
27973: 1456 unit uPSI_DelticsBitField;
27974: 1457 unit uPSI_DelticsSysUtils;
27975: 1458 unit uPSI_ALIniFiles2.pas
27976: 1459 unit uPSI_StarCalc2.pas
27977: 1460 unit uPSI_IdHashMessageDigest2.pas
27978: 1461 unit uPSI_U_Splines;
27979: 1462 unit uPSI_U_CoasterB.pas;
27980: 1463 U_SpringMass2.pas
27981: 1464 uPSI_MARSCoreUtils;
27982: 1465 unit uPSI_clJsonParser.pas
27983: 1466 unit uPSI_SynHighlighterPython.pas
27984: 1467 unit uPSI_DudsCommonDelphi;
27985: 1468 unit uPSI_AINNNeuron;
27986: 1469 unit uPSI_PJConsoleApp2;
27987: 1470 unit uPSI_PJPipeFilters2;
27988: 1471 unit uPSI_uHTMLBuilder;
27989: 1472 unit uPSI_PJPipe2;
27990: 1473 uPSI_WinApiDownload,
27991: 1474 uPSI_pxQRcode,    //beta
27992: 1475 unit uPSI_neuralplanbuilder2
27993: 1476 unit uPSI_DelphiZXingQRCode;
27994: 1477 unit uPSI_RestJsonUtils;
27995: 1478 unit UtilsTimeCode;
27996: 1479 unit uPSC_classes2.pas; //TList
27997: 1480 unit uPSC_std2.pas
27998: 1481 unit uPSI_maxIniFiles.pas
27999: 1482 unit uROPSImports.pas
28000: 1483 unit uROPSServerLink.pas
28001: 1484 unit uPSI_KLibUtils;
28002: 1485 unit uPSI_PathFunc2;  //inno setup
28003: 1486 unit KLibVC_Redist.pas;
28004: 1487 unit HTTPApp2.pas;
28005: 1488 unit uPSI_XCollection2;
28006: 1489 unit uPSI_KLibWindows;
28007: 1490 unit KlibConstants;
28008: 1491 unit uPSI_AzuliaUtils.pas
28009: 1492 unit uPSI_ALHttpClient2;
28010: 1493 unit uPSI_ALWininetHttpClient2;
28011: 1494 unit uPSI_UtilsMax41.pas
28012: 1495 unit uPSI_JclSysUtils1;
```

```
28013: 1496 unit uPSI_RestUtils;
28014: 1497 unit uPSI_TeEngine2.pas
28015: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
28016: 1499 unit uPSI_PSResources.pas
28017: 1500 unit uPSI_TeCanvas2_1.pas
28018: 1501 unit uPSI_DataSetConverter4DUtil
28019: 1502 unit uPSI_neuralfit2.pas;
28020:
28021: Total of Function Calls: 35958
28022: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
28023: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
28024: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
28025: Entry Point 24783132 - Contained Sections 10
28026: Amount of Functions: 21314
28027: Amount of Procedures: 12650
28028: Amount of Constructors: 1980
28029: Amount of Destructors: 14
28030: Totals of Calls: 35958
28031: SHA1: of 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
28032: CRC32: 9E995FA2:  33805592 bytes
28033: ▯▯▯ mX4 executed: 17/10/2022 12:23:45  Runtime: 0:14:16.719  Memload: 36% use
28034:
28035: ****************************************************************
28036: Release Notes maXbox 4.7.6.10 II November 2021 mX476
28037: ****************************************************************
28038: Add 10 Units + 3 Tutorials
28039:
28040: 1441 unit uPSI_neuralgeneric.pas; CAI
28041: 1442 unit uPSI_neuralthread.pas; CAI
28042: 1443 unit uPSI_uSysTools; TuO
28043: 1444 unit upsi_neuralsets; mX4
28044: 1445 unit uPSI_uWinNT.pas mX4
28045: 1446 unit uPSI_URungeKutta4.pas ICS
28046: 1447 unit uPSI_UrlConIcs.pas ICS
28047: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
28048: 1449 unit uPSI_Numedit2 mX4
28049: 1450 unit uPSI_PsAPI_3.pas mX4
28050:
28051: Total of Function Calls: 35078
28052: SHA1: of 4.7.6.10  D4B0A36E42E9E89642A140CCEE2B7CCDDE3D041A
28053: CRC32: B8F2450F 30.6 MB (32,101,704 bytes)
28054:
28055: Amount of Functions: 20591
28056: Amount of Procedures: 12332
28057: Amount of Constructors: 1943
28058: Amount of Destructors: 14
28059: Totals of Calls: 34880
28060: SHA1: of 4.7.6.10 CF939E3A8D4723DB1DEF383C5FC961E06728C58F
28061: CRC32: 38F88218:  32022344 bytes
28062: ▯▯▯ mX4 executed: 14/11/2021 11:02:18  Runtime: 0:12:36.968  Memload: 42% use
28063:
28064: ****************************************************************
28065: Tech Notes maXbox 4.7.5.90 V October 2021 mX47
28066: ****************************************************************
28067: add uPSI_flcFloats.pas to complete Fundamentals5
28068: Registermethod: count, capacity in TNNetList
28069:  with CL.Add(TNNetList) do RegisterMethod(@TNNetList.Count, 'Count');
28070: alias flots() - floattostr()
28071: Function flots( const A : Float) : String');
28072: plus resources MOON_FULL, maxbox4logo.jpg, EARTH.bmp
28073:  Bugfix TSize Type
28074:  CL.AddClassN(CL.FindClass('TOBJECT'),'TInteger'); Bigints for BigFloat
28075:  - RegisterMethod(FindClass('TObject'),'TBigFloat'),'Procedure Assign4( A : TInteger);
28076: fix type TSize, add type TIntegerArray
28077:   CL.AddTypeS('TSize', 'record cx: Longint; cy: Longint; end');   //not anymore in JvVCLUtils
28078:   asize:= MakeSize(2,34);
28079:   wGetTextExtentPoint32(aDC, 'f' ,4, asize)
28080:   writeln(botostr(GetViewportExtEx( adc, asize)));
28081: Function AddLayer( pLayer : TNNetLayer) : TNNetLayer;
28082: Function AddLayer1( strData : string) : TNNetLayer;
28083: Function AddLayer2( pLayers : array of TNNetLayer) : TNNetLayer;
28084: TestConvolutionAPI Routine internal
28085:
28086: Docu of main Neural Unit for maXbox: http://www.softwareschule.ch/examples/uPSI_NeuralNetworkCAI.txt
28087: http://www.softwareschule.ch/examples/uPSI_neuralnetworkcai.txt
28088: http://www.softwareschule.ch/examples/uPSI_neuralvolume.txt
28089: http://www.softwareschule.ch/examples/uPSI_neuraldatasets.txt
28090: http://www.softwareschule.ch/examples/uPSI_neuralfit.txt
28091: Source https://github.com/joaopauloschuler/neural-api
28092:
28093: ******************************************************************************
28094: Procedure
28095: Proc SIZE 12456
28096: Proc ****************************Now the Proc list*******************
28097: Procedure( ACol, ARow : Int; Items : TStrings)
28098: Procedure( Agg : TAggregate)
28099: Procedure( ASender : TComponent; const AReplyStatus : TReplyStatus)
28100: Procedure( ASender : TComponent; const AString :Str; var AMsg : TIdMessage)
28101: Procedure( ASender : TComponent; var AMsg : TIdMessage)
```

```
28102: Procedure( ASender : TObject; const ABytes : Int)
28103: Procedure( ASender : TObject; VStream : TStream)
28104: Procedure( AThread : TIdThread)
28105: Procedure( AWebModule : TComponent)
28106: Procedure( Column : TColumn)
28107: Procedure(const AUsername:Str; const APassword:Str; AAuthenticationResult: Bool)
28108: Procedure( const iStart : Int; const sText :Str)
28109: Procedure(Control:TCustomTabControl; TabIndex:Int; const Rect : TRect; Active :Bool)
28110: Procedure( Database : TDatabase; LoginParams : TStrings)
28111: Proc DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:TReconcileAction)
28112: Proc ( DATASET : TDATASET)
28113: Proc ( DataSet:TDataSet;E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
28114: Procedure( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
28115: Procedure( DataSet : TDataSet; UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction)
28116: Procedure( DATASET : TDATASET; var ACCEPT :Bool)
28117: Procedure( DBCtrlGrid : TDBCtrlGrid; Index : Int)
28118: Procedure( Done : Int)
28119: Procedure( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
28120: Procedure(HeaderControl:TCustomHeaderControl;Sect:THeaderSection;const Rect:TRect;Pressed:Bool)
28121: Procedure(HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
28122: Procedure( HeaderControl : THeaderControl; Section : THeaderSection)
28123: Procedure(HeaderControl:THeaderControl;Section:THeaderSection;const Rect:TRect;Pressed:Bool)
28124: Procedure(HeaderControl:THeaderControl;Sectin:THeaderSection;Width:Int;State:TSectionTrackState)
28125: Procedure(Sender:TCustomListView;const ARect:TRect;Stage:TCustomDrawStage;var DefaultDraw: Bool)
28126: Procedure(Sender: TCustomListView; const ARect : TRect; var DefaultDraw :Bool)
28127: Procedure(Sender: TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
28128: Procedure(Sender: TCustomTreeView; const ARect : TRect; var DefaultDraw :Bool)
28129: Procedure(SENDER: TFIELD; const TEXT :Str)
28130: Procedure(SENDER: TFIELD; var TEXT :Str; DISPLAYTEXT :Bool)
28131: Procedure(Sender: TIdTelnet; const Buffer :Str)
28132: Procedure(Sender: TIdTelnet; Status : TIdTelnetCommand)
28133: Procedure(SENDER: TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED :Bool)
28134: Procedure(SENDER: TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
28135: Procedure(SENDER: TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : Int)
28136: Procedure(Sender: TObject; ACol, ARow : Longint; const Value :Str)
28137: Procedure(Sender: TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
28138: Procedure(Sender: TObject; ACol, ARow : Longint; var CanSelect :Bool)
28139: Procedure(Sender: TObject; ACol, ARow : Longint; var Value :Str)
28140: Procedure(Sender: TObject; Button : TMPBtnType)
28141: Procedure(Sender: TObject; Button : TMPBtnType; var DoDefault :Bool)
28142: Procedure(Sender: TObject; Button : TUDBtnType)
28143: Procedure(Sender: TObject; Canvas: TCanvas; PageRect:TRect; var DoneDrawing:Bool)
28144: Procedure(SenderTObject;ClientSocket:TServerClientWinSocket;var SocketThread: TServerClientThread)
28145: Procedure(Sender: TObject; Column : TListColumn)
28146: Procedure(Sender: TObject; Column : TListColumn; Point : TPoint)
28147: Procedure(Sender: TObject; Connecting :Bool)
28148: Procedure(Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
28149: Procedure(Sendr:TObject;const Rect:TRect;DataCol:Int;Column:TColumn;State:TGridDrawState)
28150: Procedure( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
28151: Procedure(Sender:TObject;const UserStr:str;var DateAndTime:TDateTime;var AllowChange:Bool)
28152: Procedure(Sender:TObject; E : Exception; var Handled :Bool)
28153: Procedure(Sender:TObject; FromIndex, ToIndex : Longint)
28154: Procedure(Sender:TObject; FromSection,ToSection: THeaderSection; var AllowDrag:Bool)
28155: Procedure(Sender:TObject; Index : LongInt)
28156: Procedure(Sender:TObject; Item : TListItem)
28157: Procedure(Sender:TObject; Item : TListItem; Change : TItemChange)
28158: Procedure(Sender:TObject;Item: TListItem; Change:TItemChange;var AllowChange:Boolean)
28159: Procedure(Sender:TObject; Item : TListItem; Selected :Bool)
28160: Procedure(Sender:TObject; Item : TListItem; var AllowEdit :Bool)
28161: Procedure(Sender:TObject; Item : TListItem; var S :Str)
28162: Procedure(Sender:TObject; Item1, Item2 : TListItem; Data : Int; var Compare : Int)
28163: Procedure(Sender:TObject; ModalResult : TModalResult; var CanClose :Bool)
28164: Procedure(Sender:TObject; Month : LongWord; var MonthBoldInfo : LongWord)
28165: Procedure(Sender:TObject; NewTab : Int; var AllowChange :Bool)
28166: Procedure(Sender:TObject; Node : TTreeNode)
28167: Procedure(Sender:TObject; Node : TTreeNode; var AllowChange :Bool)
28168: Procedure(Sender:TObject; Node : TTreeNode; var AllowCollapse :Bool)
28169: Procedure(Sender:TObject; Node : TTreeNode; var AllowEdit :Bool)
28170: Procedure(Sender:TObject; Node : TTreeNode; var AllowExpansion :Bool)
28171: Procedure(Sender:TObject; Node : TTreeNode; var S :Str)
28172: Procedure(Sender:TObject; Node1, Node2 : TTreeNode; Data : Int; var Compare : Int)
28173: Procedure(Sender : TObject; NumObjects, NumChars : Int; var SaveClipboard :Bool)
28174: Procedure(Sender : TObject; Rect : TRect)
28175: Procedure(Sender:TObject; Request:TWebRequest;Response:TWebResponse; var Handled:Bool)
28176: Procedure(Sender:TObject;Shift:TShiftState;X,Y:Int;Orient:TPageScrollerOrientation;var Delta:Int
28177: Procedure(Sender TObject; Socket : TCustomWinSocket)
28178: Procedure(Sender:TObject;Socket:TCustomWinSocket;ErrorEvent:TErrorEvent;var ErrorCode: Int)
28179: Procedure( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
28180: Procedure(Sender: TObject; Socket : TSocket; var ClientSocket: TServerClientWinSocket)
28181: Procedure(SENDER: TOBJECT; SOURCE : TMENUITEM; REBUILD :Bool)
28182: Procedure(Sender: TObject; StartPos, EndPos : Int; var AllowChange :Bool)
28183: Procedure(Sender: TObject; TabCanvas: TCanvas;R:TRect; Index: Int; Selected :Bool)
28184: Procedure(Sender: TObject; TabIndex : Int; var ImageIndex : Int)
28185: Procedure(Sender: TObject; Thread : TServerClientThread)
28186: Procedure(Sender: TObject; TickCount :Card; var Reset :Bool)
28187: Procedure(Sender: TObject; Username, Password :Str)
28188: Procedure(Sender: TObject; var AllowChange :Bool)
28189: Procedure(Sender: TObject; var AllowChange:Boolean; NewValue:SmallInt; Direction:TUpDownDirection)
```

```
28190: Procedure(Sender: TObject; var Caption :Str; var Alignment : THTMLCaptionAlignment)
28191: Procedure( Sender : TObject; var Continue :Bool)
28192: Procedure(Sender:TObject;var dest:str;var NumRedirect:Int;var Handled:bool;var VMethod:TIdHTTPMethod)
28193: Procedure(Sender : TObject; var Username :Str)
28194: Procedure(Sender : TObject; Wnd : HWND)
28195: Procedure(Sender : TToolbar; Button : TToolButton)
28196: Procedure(Sender:TToolBar;const ARect:TRect;Stage:TCustomDrawStage;var DefaultDraw:Bool)
28197: Procedure(Sender : TToolBar; const ARect : TRect; var DefaultDraw :Bool)
28198: Procedure(Sender : TToolbar; Index : Int; var Allow :Bool)
28199: Procedure(Sender : TToolbar; Index : Int; var Button : TToolButton)
28200: Procedure(StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
28201: Procedure(StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
28202: Procedure(var FieldNames:TWideStrings;SQL:WideStr;var BindAllFields:Bool;var TableName:WideStr)
28203: Procedure(var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
28204: procedure(Sender: TObject)
28205: procedure(Sender: TObject; var Done:Bool)
28206: procedure(Sender: TObject; var Key: Word; Shift: TShiftState);
28207: Proc _T(Name: tbtString; v: Variant);
28208: Proc AbandonSignalHandler( RtlSigNum : Int)
28209: Proc Abort
28210: Proc About1Click( Sender : TObject)
28211: Proc Accept( Socket : TSocket)
28212: Proc AESSymetricExecute(const plaintext, ciphertext, password:Str)
28213: Proc AESEncryptFile(const plaintext, ciphertext, password:Str)
28214: Proc AESDecryptFile(const replaintext, ciphertext, password:Str)
28215: Proc AESEncryptString(const plaintext:str;var ciphertext:str;password:Str)
28216: Proc AESDecryptString(var plaintext:str;const ciphertext:str;password:Str)
28217: Proc Add(Addend1, Addend2 : TMyBigInt)
28218: Proc ADD(const AKEY, AVALUE : VARIANT)
28219: Proc Add(const Key :Str; Value : Int)
28220: Proc ADD(const NAME, FIELDS :Str; OPTIONS : TINDEXOPTIONS)
28221: Proc ADD(FIELD : TFIELD)
28222: Proc ADD(ITEM : TMENUITEM)
28223: Proc ADD(POPUP : TPOPUPMENU)
28224: Proc AddCharacters( xCharacters : TCharSet)
28225: Proc AddDriver( const Name :Str; List : TStrings)
28226: Proc AddImages( Value : TCustomImageList)
28227: Proc AddIndex(const Name,Fields:str;Options:TIndexOptions;const DescFields:str)
28228: Proc AddLambdaTransitionTo( oState : TniRegularExpressionState)
28229: Proc AddLoader( Loader : TBitmapLoader)
28230: Proc ADDPARAM( VALUE : TPARAM)
28231: Proc AddPassword( const Password :Str)
28232: Proc AddStandardAlias( const Name, Path, DefaultDriver :Str)
28233: Proc AddState( oState : TniRegularExpressionState)
28234: Proc AddStrings( Strings : TStrings);
28235: Proc AddStrings(Strings: TStrings);
28236: Proc AddStrings1( Strings : TWideStrings);
28237: Proc AddStringTerm(var sString:str;const sTerm:str;const sSeparator:str)
28238: Proc AddToRecentDocs( const Filename :Str)
28239: Proc AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
28240: Proc AllFunctionsList1Click( Sender : TObject)
28241: Proc AllObjectsList1Click(Sender: TObject);
28242: Proc Allocate( AAllocateBytes : Int)
28243: Proc AllResourceList1Click(Sender: TObject);
28244: Proc AnsiAppend( var dst : Ansistr; const src : Ansistr)
28245: Proc AnsiAssign( var dst : Ansistr; var src : Ansistr)
28246: Proc AnsiDelete( var dst : Ansistr; index, count : Int)
28247: Proc AnsiFree( var s : Ansistr)
28248: Proc AnsiFromWide( var dst : Ansistr; const src : WideString)
28249: Proc AnsiInsert( var dst : Ansistr; const src : Ansistr; index : Int)
28250: Proc AnsiSetLength( var dst : Ansistr; len : Int)
28251: Proc Ansistr_to_stream( const Value : Ansistr; Destin : TStream)
28252: Proc AntiFreeze;
28253: Proc APPEND
28254: Proc Append( const S : WideString)
28255: Proc Append(S:Str);
28256: Proc AppendByte( var VBytes : TIdBytes; AByte : byte)
28257: Proc AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
28258: Proc AppendChunk( Val : OleVariant)
28259: Proc AppendData( const Data : OleVariant; HitEOF :Bool)
28260: Proc AppendStr( var Dest :Str; S :Str)
28261: Proc AppendString( var VBytes : TIdBytes; const AStr :Str; ALength : Int)
28262: Proc ApplyRange
28263: Proc Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
28264: Proc Arrange( Code : TListArrangement)
28265: Proc Assert(expr :Bool; const msg:Str);
28266: Proc Assert2(expr :Bool; const msg:Str);
28267: Proc Assign( AList : TCustomBucketList)
28268: Proc Assign( Other : TObject)
28269: Proc Assign( Source : TDragObject)
28270: Proc Assign( Source : TPersistent)
28271: Proc Assign(Source: TPersistent)
28272: Proc Assign2(mystring, mypath:Str);
28273: Proc AssignCurValues( Source : TDataSet);
28274: Proc AssignCurValues1( const CurValues : Variant);
28275: Proc ASSIGNFIELD( FIELD : TFIELD)
28276: Proc ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
28277: Proc AssignFile(var F: Text; FileName:Str)
28278: Proc AssignFile(var F: TextFile; FileName:Str)
```

```
28279: Proc AssignFileRead(var mystring, myfilename:Str);
28280: Proc AssignFileWrite(mystring, myfilename:Str);
28281: Proc AssignTo( Other : TObject)
28282: Proc AssignValues( Value : TParameters)
28283: Proc ASSIGNVALUES( VALUE : TPARAMS)
28284: Proc AssociateExtension( IconPath, ProgramName, Path, Extension :Str)
28285: Proc Base64_to_stream( const Base64 : Ansistr; Destin : TStream)
28286: Proc Base64ToVar( NatData : Pointer; const SoapData : WideString);
28287: Proc Base64ToVar1( var V : Variant; const SoapData : WideString);
28288: Proc BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
28289: Proc BcdDivide( Dividend, Divisor :Str; var bcdOut : TBcd);
28290: Proc BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
28291: Proc BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
28292: Proc BcdDivide3( const Dividend : TBcd; const Divisor :Str; var bcdOut : TBcd);
28293: Proc BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
28294: Proc BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
28295: Proc BcdMultiply2( const bcdIn : TBcd; const StringIn :Str; var bcdOut : TBcd);
28296: Proc BcdMultiply3( StringIn1, StringIn2 :Str; var bcdOut : TBcd);
28297: Proc BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
28298: Proc BcdToBytes( Value : TBcd; Bytes : array of byte)
28299: Proc Beep
28300: Proc BeepOk
28301: Proc BeepQuestion
28302: Proc BeepHand
28303: Proc BeepExclamation
28304: Proc BeepAsterisk
28305: Proc BeepInformation
28306: Proc BEGINDRAG(IMMEDIATE:BOOLEAN)
28307: Proc BeginLayout
28308: Proc BeginTimer( const Delay, Resolution :Card)
28309: Proc BeginUpdate
28310: Proc BeginUpdate;
28311: Proc BigScreen1Click(Sender: TObject);
28312: Proc BinToHex(Buffer: PChar; Text: PChar; BufSize: Int);
28313: Proc BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits :Bool);
28314: Proc BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits :Bool);
28315: Proc BitsToBooleans2( const Bits : Int; var B : TBooleanArray; AllBits :Bool);
28316: Proc BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits:Bool);
28317: Proc BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
28318: Proc BooleansToBits( var Dest : Byte; const B : TBooleanArray);
28319: Proc BooleansToBits1( var Dest : Word; const B : TBooleanArray);
28320: Proc BooleansToBits2( var Dest : Int; const B : TBooleanArray);
28321: Proc BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
28322: Proc BreakPointMenuClick( Sender : TObject)
28323: Proc BRINGTOFRONT
28324: Proc BringToFront;
28325: Proc btnBackClick( Sender : TObject)
28326: Proc btnBrowseClick( Sender : TObject)
28327: Proc BtnClick( Index : TNavigateBtn)
28328: Proc btnLargeIconsClick( Sender : TObject)
28329: Proc BuildAndSendRequest( AURI : TIdURI)
28330: Proc BuildCache
28331: Proc BurnMemory( var Buff, BuffLen : Int)
28332: Proc BurnMemoryStream( Destructo : TMemoryStream)
28333: Proc CalculateFirstSet
28334: Proc Cancel
28335: Proc CancelDrag;
28336: Proc CancelEdit
28337: Proc CANCELHINT
28338: Proc CancelRange
28339: Proc CancelUpdates
28340: Proc CancelWriteBuffer
28341: Proc Capture1(ADest:TStream;out VLineCount:Int;const ADelim:str;const AIsRFCMessage:Bool;
28342: Proc Capture2(ADest: TStrings; const ADelim:Str; const AIsRFCMessage:Boolean);
28343: Proc Capture3(ADest:TStrings;out VLineCount:Int;const ADelim:str;const AIsRFCMessage:Bool
28344: Proc CaptureScreenFormat(vname:Str; vextension:Str);
28345: Proc CaptureScreenPNG(vname:Str);
28346: Proc CardinalsToI64(var I: Int64; const LowPart, HighPart:Card);
28347: Proc CASCADE
28348: Proc CastNativeToSoap(Info:PTypeInfo;var SoapData:WideStr;NatData:Pointer;var IsNull:Bool)
28349: Proc CastSoapToVariant(SoapInfo:PTypeInfo;const SoapData:WideString;NatData:Pointer);
28350: Proc cbPathClick( Sender : TObject)
28351: Proc cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
28352: Proc cedebugAfterExecute( Sender : TPSScript)
28353: Proc cedebugBreakpoint(Sender: TObject;const FileName:str;Position,Row,Col:Card)
28354: Proc cedebugCompile( Sender : TPSScript)
28355: Proc cedebugExecute( Sender : TPSScript)
28356: Proc cedebugIdle( Sender : TObject)
28357: Proc cedebugLineInfo(Sender:TObject; const FileName:Str; Position,Row,Col:Card)
28358: Proc CenterHeight( const pc, pcParent : TControl)
28359: Proc CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
28360: Proc CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
28361: Proc Change
28362: Proc ChangeBiDiModeAlignment(var Alignment: TAlignment);
28363: Proc Changed
28364: Proc ChangeDir( const ADirName :Str)
28365: Proc ChangeDirUp
28366: Proc ChangeEntryTransitions( oNewState : TniRegularExpressionState)
28367: Proc ChangeLevelBy( Value : TChangeRange)
```

```
28368: Proc ChDir(const s:Str)
28369: Proc Check(Status: Int)
28370: Proc CheckCommonControl( CC : Int)
28371: Proc CHECKFIELDNAME( const FIELDNAME :Str)
28372: Proc CHECKFIELDNAMES( const FIELDNAMES :Str)
28373: Proc CheckForDisconnect(const ARaiseExceptionIfDisconnected:bool;const AIgnoreBuffer:bool)
28374: Proc CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected :Bool)
28375: Proc CheckToken( T : Char)
28376: Proc CheckToken(t:char)
28377: Proc CheckTokenSymbol( const S :Str)
28378: Proc CheckTokenSymbol(s:str)
28379: Proc CheckToolMenuDropdown( ToolButton : TToolButton)
28380: Proc Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
28381: Proc CIED65ToCIED50( var X, Y, Z : Extended)
28382: Proc CIELABToBGR( const Source, Target : Pointer; const Count :Card);
28383: Proc CipherFile1Click(Sender: TObject);
28384: Proc Clear;
28385: Proc Clear1Click( Sender : TObject)
28386: Proc ClearColor( Color : TColor)
28387: Proc CLEARITEM( AITEM : TMENUITEM)
28388: Proc ClearMapping
28389: Proc ClearSelection( KeepPrimary :Bool)
28390: Proc ClearWriteBuffer
28391: Proc Click
28392: Proc Close
28393: Proc closeMP3;
28394: Proc Close1Click( Sender : TObject)
28395: Proc CloseDatabase( Database : TDatabase)
28396: Proc CloseDataSets
28397: Proc CloseDialog
28398: Proc CloseFile(var F: Text);
28399: Proc Closure
28400: Proc CMYKToBGR(const Source,Target:Pointer;const BitsPerSample:Byte;Count:Card);
28401: Proc CMYKToBGR1(const C,M,Y,K, Target:Pointer;const BitsPerSample: Byte; Count:Card);
28402: Proc CodeCompletionList1Click( Sender : TObject)
28403: Proc ColEnter
28404: Proc Collapse
28405: Proc Collapse( Recurse :Bool)
28406: Proc ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
28407: Proc CommaSeparatedToStringList( AList : TStrings; const Value :Str)
28408: Proc CommitFreeAndNil( var Transaction : TDBXTransaction)
28409: Proc Compile1Click( Sender : TObject)
28410: Proc ComponentCount1Click(Sender: TObject);
28411: Proc Compress(azipfolder, azipfile:Str)
28412: Proc DeCompress(azipfolder, azipfile:Str)
28413: Proc XZip(azipfolder, azipfile:Str)
28414: Proc XUnZip(azipfolder, azipfile:Str)
28415: Proc Connect( const ATimeout : Int)
28416: Proc Connect( Socket : TSocket)
28417: Proc Console1Click(Sender: TObject);
28418: Proc Continue
28419: Proc ContinueCount( var Counter : TJclCounter)
28420: Proc CONTROLDESTROYED(CONTROL:TCONTROL)
28421: Proc ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : Int)
28422: Proc ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : Int)
28423: Proc ConvertImage(vsource, vdestination:Str);
28424: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
28425: Proc ConvertBitmap(vsource, vdestination:Str);
28426: Proc ConvertToGray(Cnv: TCanvas);
28427: Proc Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Int; Length : Int)
28428: Proc Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Int; Count : Int);
28429: Proc Copy1( Source : TBytes; Offset : Int; Buffer : TValueBuffer; Count : Int);
28430: Proc CopyBytesToHostLongWord(const Source:TIdBytes;const ASourceIdx:Int;var VDest:LongWord)
28431: Proc CopyBytesToHostWord(const ASource:TIdBytes;const ASourceIndex:Int;var VDest: Word)
28432: Proc CopyFrom( mbCopy : TMyBigInt)
28433: Proc CopyMemoryStream( Source, Destination : TMemoryStream)
28434: Proc CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
28435: Proc CopyTIdByteArray(const ASource:array of Byte; const ASourceIndex:Int; var VDest:array of Byte; const
       ADestIndex : Int; const ALength : Int)
28436: Proc CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
28437: Proc CopyTIdCardinal(const ASource:Card;var VDest:TIdBytes;const ADestIndex:Int)
28438: Proc CopyTIdInt64(const ASource: Int64;var VDest: TIdBytes; const ADestIndex : Int)
28439: Proc CopyTIdIPV6Address(const ASource:TIdIPv6Address;var VDest:TIdBytes;const ADestIdx:Int)
28440: Proc CopyTIdLongWord(const ASource:LongWord;var VDest:TIdBytes;const ADestIndex:Int)
28441: Proc CopyTIdNetworkLongWord(const ASource:LongWord;var VDest:TIdBytes;const ADestIndex:Int)
28442: Proc CopyTIdNetworkWord(const ASource:Word;var VDest:TIdBytes;const ADestIndex:Int)
28443: Proc CopyTIdString(const ASource:Str;var VDest:TIdBytes;const ADestIndex:Int;ALength:Int)
28444: Proc CopyTIdWord( const ASource: Word; var VDest: TIdBytes; const ADestIndex : Int)
28445: Proc CopyToClipboard
28446: Proc CountParts
28447: Proc CreateDataSet
28448: Proc CreateEmptyFile( const FileName :Str)
28449: Proc CreateFileFromString( const FileName, Data :Str)
28450: Proc CreateFromDelta( Source : TPacketDataSet)
28451: Proc CREATEHANDLE
28452: Proc CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
28453: Proc CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine :Str)
```

```
28454: Proc CreateProcAsUserEx(const UserDoman,UserNme,Passwo,CommandLine:str;const Environ:PChar)
28455: Proc CreateTable
28456: Proc CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
28457: Proc CSyntax1Click(Sender: TObject);
28458: Proc CurrencyToComp( Value : Currency; var Result : Comp)
28459: Proc CURSORPOSCHANGED
28460: Proc CutFirstDirectory(var S:Str)
28461: Proc DataBaseError(const Message:Str)
28462: Proc DateTimeToString( var Result :Str; Format :Str; DateTime : TDateTime);
28463: Proc DateTimeToString(var Result:Str; const Format:Str; DateTime: TDateTime)
28464: Proc DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
28465: Proc DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
28466: Proc DBIError(errorCode: Int)
28467: Proc DebugOutput( const AText :Str)
28468: Proc Debugln(DebugLOGFILE:Str; Event_message:Str);
28469: Proc DebugRun1Click( Sender : TObject)
28470: Proc Dec;
28471: Proc DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
28472: Proc DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
28473: Proc DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
28474: Proc DecodeDateMonthWeek(const AVal:TDateTime;out AYear,AMonth,AWeekOfMonth,ADayOfWeek :Word)
28475: Proc DecodeDateTime(const AVal:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMSec:Word)
28476: Proc DecodeDateWeek(const AValue: TDateTime;out AYear,AWeekOfYear,ADayOfWeek: Word)
28477: Proc DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
28478: Proc DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
28479: Proc DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
28480: Proc Decompile1Click( Sender : TObject)
28481: Proc DefaultDrawColumnCell(const Rect:TRect;DataCol:Int;Column:TColum;State:TGridDrawState)
28482: Proc DefaultDrawDataCell(const Rect: TRect; Field: TField; State: TGridDrawState)
28483: Proc DeferLayout
28484: Proc defFileread
28485: Proc DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
28486: Proc DelayMicroseconds( const MicroSeconds : Int)
28487: Proc Delete
28488: Proc Delete(const AFilename :Str)
28489: Proc Delete(const Index : Int)
28490: Proc DELETE(INDEX : Int)
28491: Proc Delete(Index : LongInt)
28492: Proc Delete(Node : TTreeNode)
28493: Proc Delete(var s: AnyString; ifrom, icount: Longint);
28494: Proc DeleteAlias( const Name :Str)
28495: Proc DeleteDriver( const Name :Str)
28496: Proc DeleteIndex( const Name :Str)
28497: Proc DeleteKey( const Section, Ident :Str)
28498: Proc DeleteRecords
28499: Proc DeleteRecords( AffectRecords : TAffectRecords)
28500: Proc DeleteString( var pStr :Str; const pDelStr :Str)
28501: Proc DeleteTable
28502: Proc DelphiSite1Click(Sender: TObject);
28503: Proc Deselect
28504: Proc Deselect( Node : TTreeNode)
28505: Proc DestroyComponents
28506: Proc DestroyHandle
28507: Proc Diff( var X : array of Double)
28508: Proc Diff(var X: array of Double);
28509: Proc DirCreate( const DirectoryName :Str);
28510: Proc DISABLEALIGN
28511: Proc DisableConstraints
28512: Proc Disconnect
28513: Proc Disconnect( Socket : TSocket)
28514: Proc Dispose
28515: Proc Dispose(P: PChar)
28516: Proc DivMod( Dividend : Int; Divisor : Word; var Result, Remainder : Word)
28517: Proc DoKey( Key : TDBCtrlGridKey)
28518: Proc DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
28519: Proc DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
28520: Proc Dormant
28521: Proc DoubleToBcd1( const AValue : Double; var bcd : TBcd);
28522: Proc DoubleToBytes( Value : Double; Bytes : array of byte)
28523: Proc DoubleToComp( Value : Double; var Result : Comp)
28524: Proc doWebCamPic(picname:Str); //eg: c:\mypic.png
28525: Proc Draw( Canvas : TCanvas; X, Y, Index : Int; Enabled :Bool);
28526: Proc Draw(X, Y: Int; Graphic: TGraphic);
28527: Proc Draw1(Canvas:TCanvas;X,Y,Indx:Int;ADrawingStyle:TDrawingStyl;AImageType:TImageType;Enabled:Bool);
28528: Proc DrawArrow(ACanvas:TCanvas; Direction:TScrollDirection;Location:TPoint;Size:Int)
28529: Proc DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Int; Shadow :Bool)
28530: Proc DrawChevron(ACanvas:TCanvas; Direction: TScrollDirection; Location:TPoint; Size : Int)
28531: Proc DrawColumnCell( const Rect : TRect; DataCol : Int; Column : TColumn; State : TGridDrawState)
28532: Proc DrawFocusRect(const Rect: TRect);
28533: Proc DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
28534: Proc DRAWMENUITEM(MENUITEM: TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE: TOWNERDRAWSTATE)
28535: Proc DrawOverlay(Canvas:TCanvas;X,Y:Int;ImageIndex:Int;Overlay:TOverlay;Enabled:Bool);
28536: Proc DrawOverlay1(Canvas:TCanvas;X,Y:Int;ImageIndex:Int;Overlay:TOverlay;ADrawingStyle:
       TDrawingStyle;AImageType:TImageType;Enabled:Bool);
28537: Proc drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: Int);
28538: Proc DrawPolyLine(const Canvas:TCanvas;var Points:TPointArray;const ClipRect: TRect)
28539: Proc DropConnections
28540: Proc DropDown
28541: Proc DumpDescription( oStrings : TStrings)
```

```
28542: Proc DumpStateTable( oStrings : TStrings)
28543: Proc EDIT
28544: Proc EditButtonClick
28545: Proc EditFont1Click( Sender : TObject)
28546: Proc Ellipse(X1, Y1, X2, Y2: Int);
28547: Proc Ellipse1( const Rect : TRect);
28548: Proc EMMS
28549: Proc Encode( ADest : TStream)
28550: Proc ENDDRAG(DROP:BOOLEAN)
28551: Proc EndEdit( Cancel :Bool)
28552: Proc EndTimer
28553: Proc EndUpdate
28554: Proc EraseSection( const Section :Str)
28555: Proc Error( const Ident :Str)
28556: Proc Error(Ident:Int)
28557: Proc ErrorFmt( const Ident :Str; const Args : array of const)
28558: Proc ErrorStr( const Message :Str)
28559: Proc ErrorStr(Message:str)
28560: Proc Exchange( Index1, Index2 : Int)
28561: Proc Exchange(Index1, Index2: Int);
28562: Proc Exec( FileName, Parameters, Directory :Str)
28563: Proc ExecProc
28564: Proc ExecSQL( UpdateKind : TUpdateKind)
28565: Proc Execute
28566: Proc Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
28567: Proc ExecuteAndWait( FileName :Str; Visibility : Int)
28568: Proc ExecuteCommand(executeFile, paramstring:Str)
28569: Proc ExecuteShell(executeFile, paramstring:Str)
28570: Proc ShellExecuteAndWait(executeFile, paramstring:Str);
28571: Proc ExitThread(ExitCode: Int); stdcall;
28572: Proc ExitProcess(ExitCode: Int); stdcall;
28573: Proc Expand( AUserName :Str; AResults : TStrings)
28574: Proc Expand( Recurse :Bool)
28575: Proc ExportClipboard1Click( Sender : TObject)
28576: Proc ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
28577: Proc ExtractContentFields( Strings : TStrings)
28578: Proc ExtractCookieFields( Strings : TStrings)
28579: Proc ExtractFields(Separators,WhiteSpace:TSysCharSet; Content:PChar; Strings : TStrings)
28580: Proc ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
28581: Proc ExtractHTTPFields(Separators,WhiteSpace: TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
28582: Proc ExtractQueryFields( Strings : TStrings)
28583: Proc FastDegToGrad
28584: Proc FastDegToRad
28585: Proc FastGradToDeg
28586: Proc FastGradToRad
28587: Proc FastRadToDeg
28588: Proc FastRadToGrad
28589: Proc FileClose( Handle : Int)
28590: Proc FileClose(handle: Int)
28591: Proc FilesFromWildcard(Dir,Mask:str;var Files:TStringList;Subdirs,ShowDirs,Multitask:Bool)
28592: Proc FileStructure( AStructure : TIdFTPDataStructure)
28593: Proc FillByte2(var X: Byte ; count: Int; value: byte)
28594: Proc FillBytes( var VBytes : TIdBytes; const ACount : Int; const AValue : Byte)
28595: Proc FillChar( Buffer : TRecordBuffer; Length : Int; value : Byte)
28596: Proc FillChar2(var X: PChar ; count: Int; value: char)
28597: Proc FillCharS(var p:Str; count: Int; value: char); //fix3.8
28598: Proc FillIPList
28599: Proc FillRect(const Rect: TRect);
28600: Proc FillTStrings( AStrings : TStrings)
28601: Proc FilterOnBookmarks( Bookmarks : array of const)
28602: Proc FinalizePackage(Module: HMODULE)
28603: Proc FindClose;
28604: Proc FindClose2(var F: TSearchRec)
28605: Proc FindMatches( const sString:str; xNotify:TniRegularExpressionMatchFoundEvent);
28606: Proc FindMatches1(const sString:str;iStart:Int;xNotify:TniRegularExpressnMatchFoundEvent);
28607: Proc FindNearest( const KeyValues : array of const)
28608: Proc FinishContext
28609: Proc FIRST
28610: Proc FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
28611: Proc FloatToDecimal(var Result:TFloatRec;const Val:extend;ValueType:TFloatValue;Precis,Decs:Int);
28612: Proc FloodFill( X, Y : Int; Color : TColor; FillStyle : TFillStyle)
28613: Proc FlushSchemaCache( const TableName :Str)
28614: Proc FmtStr(var Result:Str; const Format:Str; const Args: array of const)
28615: Proc FOCUSCONTROL(CONTROL:TWINCONTROL)
28616: Proc Form1Close( Sender : TObject; var Action : TCloseAction)
28617: Proc FormActivate( Sender : TObject)
28618: Proc FormatLn(const format:Str; const args: array of const);  //alias
28619: Proc FormClose( Sender : TObject; var Action : TCloseAction)
28620: Proc FormCreate( Sender : TObject)
28621: Proc FormDestroy( Sender : TObject)
28622: Proc FormKeyPress( Sender : TObject; var Key : Char)
28623: Proc FormOutput1Click(Sender: TObject);
28624: Proc FormToHtml( Form : TForm; Path :Str)
28625: Proc FrameRect(const Rect: TRect);
28626: Proc Frame3D(Canvas:TCanvas;var Rect:TRect;TopColor,BottomColor:TColor; Width : Int)
28627: Proc NotebookHandlesNeeded( Notebook : TNotebook)
28628: Proc Free( Buffer : TRecordBuffer)
28629: Proc Free( Buffer : TValueBuffer)
```

```
28630: Proc Free;
28631: Proc FreeAndNil(var Obj:TObject)
28632: Proc FreeImage
28633: Proc FreeMem(P: PChar; Size: Int)
28634: Proc FreeTreeData( Tree : TUpdateTree)
28635: Proc Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Int)
28636: Proc FullCollapse
28637: Proc FullExpand
28638: Proc GenerateDPB(sl: TStrings; var DPB:Str; var DPBLength: Short);  //InterBase
28639: Proc GenerateTPB(sl: TStrings; var TPB:Str; var TPBLength: Short);
28640: Proc OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
28641: Proc Get1( AURL :Str; const AResponseContent : TStream);
28642: Proc Get1( const ASourceFile :Str; ADest : TStream; AResume :Bool);
28643: Proc Get2(const ASourceFile,ADestFile:Str;const ACanOverwrite: bool; AResume: Bool);
28644: Proc GetAliasNames( List : TStrings)
28645: Proc GetAliasParams( const AliasName :Str; List : TStrings)
28646: Proc GetApplicationsRunning( Strings : TStrings)
28647: Proc getBox(aURL, extension:Str);
28648: Proc GetCommandTypes( List : TWideStrings)
28649: Proc GetConfigParams( const Path, Section :Str; List : TStrings)
28650: Proc GetConnectionNames( List : TStrings; Driver :Str; DesignMode :Bool)
28651: Proc GetConvFamilies( out AFamilies : TConvFamilyArray)
28652: Proc GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
28653: Proc GetDatabaseNames( List : TStrings)
28654: Proc GetDataPacket( DataSet : TDataSet; var RecsOut : Int; out Data : OleVariant)
28655: Proc GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize:longWORD; var ImageSize : longWORD)
28656: Proc GetDir(d: byte; var s:Str)
28657: Proc GetDirList( const Search :Str; List : TStrings; Recursive :Bool)
28658: Proc GetDriverNames( List : TStrings)
28659: Proc GetDriverNames( List : TStrings; DesignMode :Bool)
28660: Proc GetDriverParams( const DriverName :Str; List : TStrings)
28661: Proc GetEMails1Click( Sender : TObject)
28662: Proc getEnvironmentInfo;
28663: Func getEnvironmentString:Str;
28664: Proc GetFieldNames(const DatabaseName, TableName :Str; List : TStrings)
28665: Proc GetFieldNames(const TableName :Str; List : TStrings)
28666: Proc GetFieldNames(const TableName :Str; List : TStrings);
28667: Proc GetFieldNames(const TableName : WideString; List : TWideStrings);
28668: Proc GETFIELDNAMES(LIST : TSTRINGS)
28669: Proc GetFieldNames1(const TableName :Str; List : TStrings);
28670: Proc GetFieldNames1(const TableName :Str; SchemaName :Str; List : TStrings);
28671: Proc GetFieldNames2(const TableName : WideString;SchemaName : WideString; List : TWideStrings);
28672: Proc GetFieldNames3(const TableName : WideString; List : TWideStrings);
28673: Proc GetFileAttributeList( const Items : TStrings; const Attr : Int)
28674: Proc GetFileAttributeListEx( const Items : TStrings; const Attr : Int)
28675: Proc GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
28676: Proc GetFormatSettings
28677: Proc GetFromDIB( var DIB : TBitmapInfo)
28678: Proc GetFromHDIB( HDIB : HBitmap)
28679: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
28680: Proc GetGEOMap(C_form,apath:Str; const Data:Str); //c_form: [html/json/xml]
28681: Proc GetIcon( Index : Int; Image : TIcon);
28682: Proc GetIcon1(Index:Int; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
28683: Proc GetIndexInfo( IndexName :Str)
28684: Proc GetIndexNames( const TableName, SchemaName :Str; List : TStrings);
28685: Proc GetIndexNames( List : TStrings)
28686: Proc GetIndexNames1(const TableName : WideString; List : TWideStrings);
28687: Proc GetIndexNames2(const TableName, SchemaName : WideString; List : TWideStrings);
28688: Proc GetIndexNames4(const TableName :Str; List : TStrings);
28689: Proc GetInternalResponse
28690: Proc GETITEMNAMES( LIST : TSTRINGS)
28691: Proc GetMem(P: PChar; Size: Int)
28692: Proc GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:Int;GROUPS:array of Int)
28693: Proc GetPackageDescription(ModuleName: PChar):Str)
28694: Proc GetPackageNames( List : TStrings);
28695: Proc GetPackageNames1( List : TWideStrings);
28696: Proc GetParamList( List : TList; const ParamNames : WideString)
28697: Proc GetProcedureNames( List : TStrings);
28698: Proc GetProcedureNames( List : TWideStrings);
28699: Proc GetProcedureNames1(const PackageName :Str; List : TStrings);
28700: Proc GetProcedureNames1(List : TStrings);
28701: Proc GetProcedureNames2(const PackageName, SchemaName :Str; List : TStrings);
28702: Proc GetProcedureNames3(List : TWideStrings);
28703: Proc GetProcedureNames4(const PackageName : Widestring; List : TWideStrings);
28704: Proc GetProcedureNames5(const PackageName,SchemaName: WideString; List : TWideStrings);
28705: Proc GetProcedureParams(ProcedureName : WideString; List : TList);
28706: Proc GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
28707: Proc GetProcedureParams2(ProcedureName,PackageName,SchemaName:Widestring;List:TList);
28708: Proc GetProviderNames( Names : TWideStrings);
28709: Proc GetProviderNames( Proc : TGetStrProc)
28710: Proc GetProviderNames1( Names : TStrings);
28711: Proc GetQrCode2(Width,Height:Word;Correct_Level:str;const Data:str;apath:Str);
28712: Proc GetQrCode3(Width,Height:Word;Correct_Level:str;const Data:str;apath:str);//no open image
28713: Func GetQrCode4(Width,Height:Word;Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
28714: Proc GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
28715: Proc GetSchemaNames( List : TStrings);
28716: Proc GetSchemaNames1( List : TWideStrings);
28717: Proc getScriptandRunAsk;
28718: Proc getScriptandRun(ascript:Str);
```

```
28719: Proc getScript(ascript:Str);   //alias
28720: Proc getWebScript(ascript:Str);   //alias
28721: Proc GetSessionNames( List : TStrings)
28722: Proc GetStoredProcNames( const DatabaseName :Str; List : TStrings)
28723: Proc GetStrings( List : TStrings)
28724: Proc GetSystemTime; stdcall;
28725: Proc GetTableNames(const DatabseName,Pattern:str;Extensins,SystemTables:Bool;List:TStrings)
28726: Proc GetTableNames( List : TStrings; SystemTables :Bool)
28727: Proc GetTableNames( List : TStrings; SystemTables :Bool);
28728: Proc GetTableNames( List : TWideStrings; SystemTables :Bool);
28729: Proc GetTableNames1(List: TStrings; SchemaName : WideString; SystemTables Boolean);
28730: Proc GetTableNames1( List : TStrings; SystemTables :Bool);
28731: Proc GetTableNames2(List: TWideStrings; SchemaName:WideString;SystemTables:Boolean);
28732: Proc GetTransitionsOn( cChar : char; oStateList : TList)
28733: Proc GetVisibleWindows( List : Tstrings)
28734: Proc GoBegin
28735: Proc GotoCurrent( DataSet : TCustomClientDataSet)
28736: Proc GotoCurrent( Table : TTable)
28737: Proc GotoEnd1Click(Sender: TObject);
28738: Proc GotoNearest
28739: Proc GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
28740: Proc HandleException( E : Exception; var Handled :Bool)
28741: Proc HANDLEMESSAGE
28742: Proc HandleNeeded;
28743: Proc Head( AURL :Str)
28744: Proc Help( var AHelpContents : TStringList; ACommand :Str)
28745: Proc HexToBinary( Stream : TStream)
28746: Proc HexToBinary(Stream:TStream)
28747: Proc HideDragImage
28748: Proc HideFormCaption( FormHandle : THandle; Hide :Bool)
28749: Proc HideTraybar
28750: Proc HideWindowForSeconds(secs: Int);   {//3 seconds}
28751: Proc HideWindowForSeconds2(secs: Int; apphandle, aself: TForm);   {//3 seconds}
28752: Proc HookOSExceptions
28753: Proc HookSignal( RtlSigNum : Int)
28754: Proc HSLToRGB( const H, S, L : Single; out R, G, B : Single);
28755: Proc HTMLSyntax1Click( Sender : TObject)
28756: Proc IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
28757: Proc IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
28758: Proc ImportfromClipboard1Click( Sender : TObject)
28759: Proc ImportfromClipboard2Click( Sender : TObject)
28760: Proc IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Int)
28761: Proc Incb(var x: byte);
28762: Proc Incb2(var x: byte; inc: integer);
28763: Proc Include1Click( Sender : TObject)
28764: Proc IncludeOFF;  //preprocessing
28765: Proc IncludeON;
28766: Proc Info1Click(Sender: TObject);
28767: Proc InitAltRecBuffers( CheckModified :Bool)
28768: Proc InitContext( Request : TWebRequest; Response : TWebResponse)
28769: Proc InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
28770: Proc InitData( ASource : TDataSet)
28771: Proc InitDelta( ADelta : TPacketDataSet);
28772: Proc InitDelta1( const ADelta : OleVariant);
28773: Proc InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
28774: Proc Initialize
28775: Proc InitializePackage(Module: HMODULE)
28776: Proc INITIATEACTION
28777: Proc initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
28778: Proc InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
28779: Proc InitModule( AModule : TComponent)
28780: Proc InitStdConvs
28781: Proc InitTreeData( Tree : TUpdateTree)
28782: Proc INSERT
28783: Proc Insert(Index : Int; AClass : TClass)
28784: Proc Insert(Index : Int; AComponent : TComponent)
28785: Proc Insert(Index : Int; AObject : TObject)
28786: Proc Insert(Index : Int; const S : WideString)
28787: Proc Insert(Index : Int; Image, Mask : TBitmap)
28788: Proc Insert(Index: Int; const S:Str);
28789: Proc Insert(Index: Int; S:Str);
28790: Proc Insert(s: AnyString; var s2: AnyString; iPos: Longint);
28791: Proc InsertComponent(AComponent:TComponent)
28792: Proc InsertControl(AControl: TControl);
28793: Proc InsertIcon( Index : Int; Image : TIcon)
28794: Proc InsertMasked( Index : Int; Image : TBitmap; MaskColor : TColor)
28795: Proc InsertObject( Index : Int; const S : WideString; AObject : TObject)
28796: Proc InsertObject(Index:Int;S:str;AObject:TObject)
28797: Proc Int16ToBytes( Value : SmallInt; Bytes : array of byte)
28798: Proc Int32ToBytes( Value : Int; Bytes : array of byte)
28799: Proc Int64ToBytes( Value : Int64; Bytes : array of byte)
28800: Proc I64ToCardinals(I: Int64; var LowPart, HighPart:Card);
28801: Proc InternalBeforeResolve( Tree : TUpdateTree)
28802: Proc InvalidateModuleCache
28803: Proc InvalidateTitles
28804: Proc InvalidDateDayError( const AYear, ADayOfYear : Word)
28805: Proc InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
28806: Proc InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const ABaseDate:TDateTime)
```

```
28807: Proc InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
28808: Proc InvalidDayOfWeekInMonthError(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek: Word)
28809: Proc JavaSyntax1Click(Sender: TObject);
28810: Proc JclLocalesInfoList( const Strings : TStrings; InfoType : Int)
28811: Proc KillDataChannel
28812: Proc Largefont1Click( Sender : TObject)
28813: Proc LAST
28814: Proc LaunchCpl( FileName :Str)
28815: Proc Launch( const AFile :Str)
28816: Proc LaunchFile( const AFile :Str)
28817: Proc LetFileList(FileList: TStringlist; apath:Str);
28818: Proc lineToNumber( xmemo :Str; met :Bool)
28819: Proc ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var DefaultDraw:Bool)
28820: Proc
       ListViewCustomDrawSubItem(Sender:TCustomListView;Item:TListItem;SubItem:Int;State:TCustomDrawState;var DefaultDraw:Boo
28821: Proc ListViewData( Sender : TObject; Item : TListItem)
28822: Proc ListViewDataFind(Sender:TObject;Find:TItemFind;const FindString:Str;const
       FindPosition:TPoint;FindData:Pointer;StartIndex:Int;Direction:TSearchDirection;Wrap:Boolean;var Index:Int)
28823: Proc ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Int)
28824: Proc ListViewDblClick( Sender : TObject)
28825: Proc ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
28826: Proc ListDLLExports(const FileName:Str; List: TStrings);
28827: Proc Load( const WSDLFileName : WideString; Stream : TMemoryStream)
28828: Proc LoadBytecode1Click(Sender: TObject);
28829: Proc LoadFilefromResource(const FileName:Str; ms: TMemoryStream);
28830: Proc LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
28831: Proc LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
28832: Proc LoadFromFile( AFileName :Str)
28833: Proc LoadFromFile( const AFileName :Str; const AHeadersOnly :Bool)
28834: Proc LoadFromFile( const FileName :Str)
28835: Proc LOADFROMFILE( const FILENAME :Str; BLOBTYPE : TBLOBTYPE)
28836: Proc LoadFromFile( const FileName :Str; DataType : TDataType)
28837: Proc LoadFromFile( const FileName : WideString)
28838: Proc LoadFromFile( const FileName, FileType :Str; Bitmap : TLinearBitmap)
28839: Proc LoadFromFile(const AFileName:Str)
28840: Proc LoadFromFile(FileName:Str);
28841: Proc LoadFromFile(FileName:str)
28842: Proc LoadFromResourceID( Instance : THandle; ResID : Int)
28843: Proc LoadFromResourceName( Instance : THandle; const ResName :Str)
28844: Proc LoadFromStream( AStream : TStream; const AHeadersOnly :Bool)
28845: Proc LoadFromStream( const Stream : TStream)
28846: Proc LoadFromStream( S : TStream)
28847: Proc LoadFromStream(Stream: TSeekableStream; const Ext :Str; Bitmap : TLinarBitmap)
28848: Proc LoadFromStream(Stream: TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
28849: Proc LoadFromStream( Stream : TStream)
28850: Proc LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
28851: Proc LoadFromStream( Stream : TStream; DataType : TDataType)
28852: Proc LoadFromStream(Stream: TStream);
28853: Proc LoadFromStream1( Stream : TSeekableStream; const FormatExt :Str);
28854: Proc LoadFromStream2( Stream : TStream; const FormatExt :Str);
28855: Proc LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
28856: Proc LoadLastFile1Click( Sender : TObject)
28857: { LoadIcoToImage loads two icons from resource named NameRes,
28858:   into two image lists ALarge and ASmall}
28859: Proc LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes:Str);
28860: Proc LoadMemo
28861: Proc LoadParamsFromIniFile( FFileName : WideString)
28862: Proc Lock
28863: Proc Login
28864: Proc MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
28865: Proc MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
28866: Proc MakeCaseInsensitive
28867: Proc MakeDeterministic( var bChanged :Bool)
28868: Proc MakeGrayPal( var Palette, ColorCount : Int)
28869: // type TVolumeLevel = 0..127;, savaFilePath as C:\MyFile.wav
28870: Proc MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
28871: Proc MakeComplexSound(N:Int;freqlist:TStrgs;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
28872: type MSArray = array[0..1] of TMemoryStream;
28873: Func MakeComplexSound2(N:integer {stream # to use};
       freqlist:TStrings;Duration{mSec}:Integer;pinknoise:Bool;shape:integer;Volume:TVolumeLevel):MSArray;
28874: Proc SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:Int);
28875: Proc SetRectComplexFormatStr( const S :Str)
28876: Proc SetPolarComplexFormatStr( const S :Str)
28877: Proc AddComplexSoundObjectToList(newf,newp,newa,news:Int; freqlist: TStrings);
28878: Proc MakeVisible
28879: Proc MakeVisible( PartialOK :Bool)
28880: Proc Manual1Click( Sender : TObject)
28881: Proc MarkReachable
28882: Proc maXbox;  //shows the exe version data in a win box
28883: Proc MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
28884: Proc Memo1Change( Sender : TObject)
28885: Proc Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:str;Line,Column:Int;var
       Action:TSynReplaceAction)
28886: Proc Memo1SpecialLineColors(Sender:TObject;Line:Int;var Special:Boolean;var FG,BG:TColor)
28887: Proc Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
28888: Proc Memory1Click(Sender: TObject);
28889: Proc MERGE( MENU : TMAINMENU)
28890: Proc MergeChangeLog
28891: Proc MINIMIZE
```

```
28892: Proc MinimizeMaxbox;
28893: Proc MyCopyFile(Name1,Name2:str);
28894: Proc MkDir(const s:Str)
28895: Proc MakeDir(const s:Str);
28896: Proc ChangeDir(const s:Str);
28897: Func  makeFile(const FileName:Str): Int);
28898: Proc mnuPrintFont1Click( Sender : TObject)
28899: Proc ModalStarted
28900: Proc Modified
28901: Proc ModifyAlias( Name :Str; List : TStrings)
28902: Proc ModifyDriver( Name :Str; List : TStrings)
28903: Proc MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4,Skew,Kurtosis:Extended)
28904: Proc MouseToCell( X, Y : Int; var ACol, ARow : Longint)
28905: Proc Move( CurIndex, NewIndex : Int)
28906: Proc Move(CurIndex, NewIndex: Int);
28907: Proc Move2(const Source: TByteArray; var Dest: TByteArray; Count: Int)
28908: Proc MoveChars(const ASource:str;ASourceStart:int;var ADest:str;ADestStart,ALen:int)
28909: Proc moveCube( o : TMyLabel)
28910: Proc MoveTo( Destination : LongInt; AttachMode : TAttachMode)
28911: Proc MoveTo(X, Y: Int);
28912: Proc MoveWindowOrg(DC: HDC; DX, DY: Int);
28913: Proc MovePoint(var x,y:Extended; const angle:Extended);
28914: Proc Multiply( Multiplier1, Multiplier2 : TMyBigInt);
28915: Proc Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Int);
28916: Proc MsgAbout(Handle:Int;const Msg,Caption:str;const IcoName:str='MAINICON';Flags:DWORD=MB_OK);
28917: Proc mxButton(x,y,width,height,top,left,ahandle: Int);
28918: Proc New( Width, Height : Int; PixFormat : TPixelFormat)
28919: Proc New(P: PChar)
28920: Proc New1Click(Sender: TObject);
28921: Proc NewInstance1Click(Sender: TObject);
28922: Proc NEXT
28923: Proc NextMonth
28924: Proc Noop
28925: Proc NormalizePath( var APath :Str)
28926: Proc ObjectBinaryToText(Input, Output: TStream)
28927: Proc ObjectBinaryToText1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
28928: Proc ObjectResourceToText(Input, Output: TStream)
28929: Proc ObjectResourceToText1(Input,Output:TStream;var OriginalFormat:TStreamOriginalFormat)
28930: Proc ObjectTextToBinary(Input, Output: TStream)
28931: Proc ObjectTextToBinary1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
28932: Proc ObjectTextToResource(Input, Output: TStream)
28933: Proc ObjectTextToResource1(Input,Output:TStream;var OriginalFormat:TStreamOriginalFormat)
28934: Proc Open( const Name, Address, Service :Str; Port : Word; Block :Bool)
28935: Proc Open( const UserID : WideString; const Password : WideString);
28936: Proc Open;
28937: Proc open1Click( Sender : TObject)
28938: Proc OpenCdDrive
28939: Proc OpenCloseCdDrive( OpenMode :Bool; Drive : Char)
28940: Proc OpenCurrent
28941: Proc OpenFile(vfilenamepath:Str)
28942: Proc OpenDirectory1Click( Sender : TObject)
28943: Proc OpenDir(adir:Str);
28944: Proc OpenIndexFile( const IndexName :Str)
28945: Proc OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
28946: Proc OpenWriteBuffer( const AThreshhold : Int)
28947: Proc OptimizeMem
28948: Proc Options1( AURL :Str);
28949: Proc OutputDebugString(lpOutputString : PChar)
28950: Proc PackBuffer
28951: Proc Paint
28952: Proc PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch :Bool)
28953: Proc PaintToTBitmap( Target : TBitmap)
28954: Proc PaletteChanged
28955: Proc ParentBiDiModeChanged
28956: Proc PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
28957: Proc PasteFromClipboard;
28958: Proc PasteImage( Source : TLinearBitmap; X, Y : Int)
28959: Proc PathExtractElements(const Source:str;var Drive,Path,FileName,Ext:Str)
28960: Proc PerformEraseBackground(Control: TControl; DC: HDC);
28961: Proc PError( Text :Str)
28962: Proc Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
28963: Proc Pie(X1:Int;Y1:Int;X2:Int;Y2:Int;X3:Int;Y3:Int;X4:Int;Y4:Int);
28964: Proc Play( FromFrame, ToFrame : Word; Count : Int)
28965: Proc playmp3(mpath:Str);
28966: Proc closeMP3;
28967: Proc PlayMP31Click( Sender : TObject)
28968: Proc PointCopy( var Dest : TPoint; const Source : TPoint)
28969: Proc PointMove( var P : TPoint; const DeltaX, DeltaY : Int)
28970: Proc PolyBezier(const Points: array of TPoint);
28971: Proc PolyBezierTo(const Points: array of TPoint);
28972: Proc Polygon(const Points: array of TPoint);
28973: Proc Polyline(const Points: array of TPoint);
28974: Proc Pop
28975: Proc POPULATEOLE2MENU(SHAREDMENU:HMENU;GROUPS:array of INT;var WIDTHS:array of LONGINT)
28976: Proc PopulationVarianceAndMean(const X: TDynFloatArray;var Variance, Mean: Float)
28977: Proc POPUP( X, Y : Int)
28978: Proc PopupURL(URL : WideString);
28979: Proc POST
```

```
28980: Proc Post4(AURL:Str;const ASource: TStrings; const AResponseContent : TStream);
28981: Proc Post5( AURL :Str; const ASource, AResponseContent : TStream);
28982: Proc Post6(AURL:str;const ASource:TIdMultiPartFormDataStream;AResponseContent:TStream);
28983: Proc PostUser( const Email, FirstName, LastName : WideString)
28984: Proc PostKeyEx32(key: Word; const shift: TShiftState; specialkey:Bool);
28985: Proc Pred(X: int64);
28986: Proc Prepare
28987: Proc PrepareStatement
28988: Proc PreProcessXML( AList : TStrings)
28989: Proc PreventDestruction
28990: Proc Print( const Caption :Str)
28991: Proc PrintBitmap(aGraphic: TGraphic; Title:Str);
28992: Proc printf(const format:Str; const args: array of const);
28993: Proc PrintList(Value: TStringList);
28994: Proc PrintImage(aValue:TBitmap;Style:TBitmapStyle);
28995: //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
28996: Proc Printout1Click( Sender : TObject)
28997: Proc ProcessHeaders
28998: Proc PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
28999: Proc ProcessMessage( AMsg : TIdMessage; AHeaderOnly :Bool);
29000: Proc ProcessMessage1(AMsg:TIdMessage;const AStream TStream; AHeaderOnly:Bool);
29001: Proc ProcessMessage2(AMsg: TIdMessage; const AFilename :Str; AHeaderOnly :Bool);
29002: Proc ProcessMessagesOFF;  //application.processmessages
29003: Proc ProcessMessagesON;
29004: Proc ProcessPath(const EditText:str;var Drive:Char;var DirPart:str;var FilePart:str)
29005: Proc ProcessPath1(const EditText:str;var Drive:Char;var DirPart:str;var FilePart:str);
29006: Proc Proclist Size is: 3797 /1415
29007: Proc procMessClick( Sender : TObject)
29008: Proc PSScriptCompile( Sender : TPSScript)
29009: Proc PSScriptExecute( Sender : TPSScript)
29010: Proc PSScriptLine( Sender : TObject)
29011: Proc Push( ABoundary :Str)
29012: Proc PushItem(AItem: Pointer)
29013: Proc Put2( AURL :Str; const ASource, AResponseContent : TStream);
29014: Proc Put2(const ASourceFile:Str; const ADestFile:Str;const AAppend:boolean);
29015: Proc PutLinuxLines(const Value:Str)
29016: Proc Quit
29017: Proc RaiseConversionError( const AText :Str);
29018: Proc RaiseConversionError1( const AText :Str; const AArgs : array of const);
29019: Proc RaiseConversionRegError( AFamily : TConvFamily; const ADescription :Str)
29020: Proc RaiseException(Ex: TIFException; Param:Str);
29021: Proc RaiseExceptionForLastCmdResult;
29022: Proc RaiseLastException;
29023: Proc RaiseException2;
29024: Proc RaiseException3(const Msg:Str);
29025: Proc RaiseExcept(const Msg:Str);
29026: Proc RaiseLastOSError
29027: Proc RaiseLastWin32;
29028: Proc RaiseLastWin32Error)
29029: Proc RaiseListError( const ATemplate :Str; const AData : array of const)
29030: Proc RandomFillStream( Stream : TMemoryStream)
29031: Proc randomize;
29032: Proc Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
29033: Proc RCS
29034: Proc Read( Socket : TSocket)
29035: Proc Readln1(var ast:Str);  //of inputquery
29036: Proc ReadBlobData
29037: Proc ReadBuffer(Buffer:str;Count:LongInt)
29038: Proc ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
29039: Proc ReadSection( const Section :Str; Strings : TStrings)
29040: Proc ReadSections( Strings : TStrings)
29041: Proc ReadSections( Strings : TStrings);
29042: Proc ReadSections1( const Section :Str; Strings : TStrings);
29043: Proc ReadSectionValues( const Section :Str; Strings : TStrings)
29044: Proc ReadStream( AStream:TStream;AByteCount:LongInt;const AReadUntilDisconnect:Bool)
29045: Proc ReadStrings( ADest : TStrings; AReadLinesCount : Int)
29046: Proc ReadVersion2(aFileName:Str; aVersion : TStrings);
29047: Func  ReadVersion(aFileName:Str; aVersion : TStrings):Bool;
29048: Proc Realign;
29049: Proc Rectangle(X1, Y1, X2, Y2: Int);
29050: Proc Rectangle1( const Rect : TRect);
29051: Proc RectCopy( var Dest : TRect; const Source : TRect)
29052: Proc RectFitToScreen( var R : TRect)
29053: Proc RectGrow( var R : TRect; const Delta : Int)
29054: Proc RectGrowX( var R : TRect; const Delta : Int)
29055: Proc RectGrowY( var R : TRect; const Delta : Int)
29056: Proc RectMove( var R : TRect; const DeltaX, DeltaY : Int)
29057: Proc RectMoveTo( var R : TRect; const X, Y : Int)
29058: Proc RectNormalize( var R : TRect)
29059: //  TFileCallbackProc = procedure(filename:str);
29060: Proc RecurseDirectory(Dir:Str;IncludeSubs:Bool;callback: TFileCallbackProcedure);
29061: Proc RecurseDirectory2(Dir:Str; IncludeSubs :Bool);
29062: Proc RedirectTransition(oOldState:TniRegularExpressionState;oNewState:TniRegularExpressionState)
29063: Proc Refresh;
29064: Proc RefreshData( Options : TFetchOptions)
29065: Proc REFRESHLOOKUPLIST
29066: Proc regExPathfinder(Pathin, fileout, firstp, aregex, ext:Str; asort:Bool);
29067: Proc RegExPathfinder2(Pathin,fileout,firstp, aregex, ext:Str; asort, acopy:Bool);
29068: Proc RegisterAuthenticationMethod(MethodName:Str; AuthClass : TIdAuthenticationClass)
```

```
29069: Proc RegisterChanges( Value : TChangeLink)
29070: Proc RegisterConversionFormat(const AExtension:str;AConversionClass:TConversionClass)
29071: Proc RegisterFileFormat(const AExtension,ADescription:str;AGraphicClass:TGraphicClass)
29072: Proc RegisterFileFormat(Extension,AppID:str;Description:str;Executable:str;IconIndex:Int)
29073: Proc ReInitialize( ADelay :Card)
29074: Proc RELEASE
29075: Proc Remove( const AByteCount : Int)
29076: Proc REMOVE( FIELD : TFIELD)
29077: Proc REMOVE( ITEM : TMENUITEM)
29078: Proc REMOVE( POPUP : TPOPUPMENU)
29079: Proc RemoveAllPasswords
29080: Proc RemoveComponent(AComponent:TComponent)
29081: Proc RemoveDir( const ADirName :Str)
29082: Proc RemoveLambdaTransitions( var bChanged :Bool)
29083: Proc REMOVEPARAM( VALUE : TPARAM)
29084: Proc RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
29085: Proc RemoveTransitionTo1( oState : TniRegularExpressionState);
29086: Proc Rename( const ASourceFile, ADestFile :Str)
29087: Proc Rename( const FileName :Str; Reload :Bool)
29088: Proc RenameTable( const NewTableName :Str)
29089: Proc Replace( Index : Int; Image, Mask : TBitmap)
29090: Proc Replace1Click( Sender : TObject)
29091: Proc ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
29092: Proc ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
29093: Proc ReplaceIcon( Index : Int; Image : TIcon)
29094: Proc ReplaceMasked( Index : Int; NewImage : TBitmap; MaskColor : TColor)
29095: Proc ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
29096: Proc ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
29097: Proc Requery( Options : TExecuteOptions)
29098: Proc Reset
29099: Proc Reset1Click( Sender : TObject)
29100: Proc ResizeCanvas( XSiz, YSiz, XPos, YPos : Int; Color : TColor)
29101: Proc ResourceExplore1Click(Sender: TObject);
29102: Proc RestoreContents
29103: Proc RestoreDefaults
29104: Proc RestoreOtherInstance( MainFormClassName, MainFormCaption :Str)
29105: Proc RetrieveHeaders
29106: Proc RevertRecord
29107: Proc RGBAToBGRA(const Source, Target: Pointer; const BitsPerSample:Byte;Count :Card)
29108: Proc RGBToBGR( const Source, Target:Pointer; const BitsPerSample:Byte; Count :Card);
29109: Proc RGBToBGR1(const R,G,B,Target: Pointer; const BitsPerSample : Byte; Count :Card);
29110: Proc RGBToHSL( const R, G, B : Single; out H, S, L : Single);
29111: Proc RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
29112: Proc RGBToHSV( r, g, b : Int; var h, s, v : Int)
29113: Proc RleCompress2( Stream : TStream)
29114: Proc RleDecompress2( Stream : TStream)
29115: Proc RmDir(const S:Str)
29116: Proc Rollback
29117: Proc Rollback( TransDesc : TTransactionDesc)
29118: Proc RollbackFreeAndNil( var Transaction : TDBXTransaction)
29119: Proc RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
29120: Proc RollbackTrans
29121: Proc RoundRect(X1, Y1, X2, Y2, X3, Y3: Int);
29122: Proc RoundToAllocGranularity64( var Value : Int64; Up :Bool)
29123: Proc RoundToAllocGranularityPtr( var Value : Pointer; Up :Bool)
29124: Proc RunDll32Internal(Wnd:HWnd; const DLLName,FuncName,CmdLine:str;CmdShow : Int)
29125: procedure RunDosInMemo(DosApp: string; AMemo:TMemo);');
29126: Proc S_AddMessageToStrings( AMessages : TStrings; AMsg :Str)
29127: Proc S_EBox( const AText :Str)
29128: Proc S_GetEncryptionKeys(DatTime1,DTime2:TDateTime;var StartKey:int;var MultKey:int;var AddKey:int
29129: Proc S_IBox( const AText :Str)
29130: Proc S_ReplaceChar( var cStr :Str; cOldChr, cNewChr : char)
29131: Proc S_ReplaceStringInFile(AFileName:str; ASearchString, AReplaceString :Str)
29132: Proc S_TokenInit( cBuffer : PChar; const cDelimiters :Str)
29133: Proc SampleVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
29134: Proc Save2Click( Sender : TObject)
29135: Proc Saveas3Click( Sender : TObject)
29136: Proc Savebefore1Click( Sender : TObject)
29137: Proc SaveBytesToFile(const Data: TBytes; const FileName:Str);
29138: Proc SaveCanvas2(vCanvas: TCanvas; FileName:Str);
29139: Proc SaveConfigFile
29140: Proc SaveOutput1Click( Sender : TObject)
29141: Proc SaveScreenshotClick(Sender: TObject);
29142: Proc SaveLn(pathname,content:Str); //Saveln(exepath+'mysavelntest.txt', memo2.text);
29143: Proc SaveToClipboardFormat(var AFormat:Word; var AData: THandle; var APalette : HPALETTE)
29144: Proc SaveToClipboardFormat(var Format: Word; var Data : THandle; var APalette : HPALETTE)
29145: Proc SaveToFile( AFileName :Str)
29146: Proc SAVETOFILE( const FILENAME :Str)
29147: Proc SaveToFile( const FileName : WideString)
29148: Proc SaveToFile( const FileName : WideString; Format : TPersistFormat)
29149: Proc SaveToFile( const FileName, FileType :Str; Bitmap : TLinearBitmap)
29150: Proc SaveToFile(FileName:Str);
29151: Proc SaveToFile(FileName:str)
29152: Proc SaveToStream( AStream : TStream; const AHeadersOnly :Bool)
29153: Proc SaveToStream(OutStream TSeekableStream; const Ext :Str; Bitmap : TLinarBitmap)
29154: Proc SaveToStream( S : TStream)
29155: Proc SaveToStream( Stream : TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
29156: Proc SaveToStream( Stream : TStream)
29157: Proc SaveToStream( Stream : TStream; Format : TDataPacketFormat)
```

```
29158: Proc SaveToStream(Stream: TStream);
29159: Proc SaveToStream(Stream:TStream)
29160: Proc SaveToStream1( Stream : TSeekableStream; const FormatExt :Str);
29161: Proc SaveToStream2( Stream : TStream; const FormatExt :Str);
29162: Proc SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
29163: Proc Say(const sText:Str)
29164: Proc SBytecode1Click( Sender : TObject)
29165: Proc ScaleImage(const SourceBitmap, ResizedBitmap:TBitmap;const ScaleAmount:Double)
29166: Proc ScriptExplorer1Click(Sender: TObject);
29167: Proc Scroll( Distance : Int)
29168: Proc Scroll( DX, DY : Int)
29169: Proc ScrollBy(DeltaX, DeltaY: Int);
29170: Proc SCROLLINVIEW(ACONTROL:TCONTROL)
29171: Proc ScrollTabs( Delta : Int)
29172: Proc Search1Click( Sender : TObject)
29173: Proc SearchAndOpenDoc(vfilenamepath:Str)
29174: Proc SearchAndOpenFile(vfilenamepath:Str)
29175: Proc SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr:Str)
29176: Proc SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr:Str; offset: Int);
29177: Proc SearchNext1Click( Sender : TObject)
29178: Proc Select( Node : TTreeNode; ShiftState : TShiftState);
29179: Proc Select1( const Nodes : array of TTreeNode);
29180: Proc Select2( Nodes : TList);
29181: Proc SelectNext( Direction :Bool)
29182: Proc SelectNextPage( GoForward :Bool; CheckTabVisible :Bool)
29183: Proc SelfTestPEM  //unit uPSI_cPEM
29184: Proc Send( AMsg : TIdMessage)
29185: //config forst in const MAILINIFILE = 'maildef.ini';
29186: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
29187: Proc SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
29188: Proc SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
29189: Proc SendMsg( AMsg : TIdMessage; const AHeadersOnly :Bool)
29190: Proc SendMsg(AMsg: TIdMessage; const AHeadersOnly:Bool = False)
29191: Proc SendResponse
29192: Proc SendStream( AStream : TStream)
29193: Proc SendMCICommand(Cmd:Str);   !
29194: Proc Set8087CW( NewCW : Word)
29195: Proc SetAll( One, Two, Three, Four : Byte)
29196: Proc SetAltRecBuffers( Old, New, Cur : PChar)
29197: Proc SetAppDispatcher( const ADispatcher : TComponent)
29198: Proc SetArrayLength;
29199: Proc SetArrayLength2String(arr: T2StringArray; asize1, asize2: Int);  //2 dimension
29200: Proc SetArrayLength2Int(arr: T2IntArray; asize1, asize2: Int);
29201: Proc SetArrayLength2String2(arr: T2StringArray; asize1, asize2: Int);  //all init
29202: Proc SetArrayLength2Int2(arr: T2IntArray; asize1, asize2: Int);
29203: Proc SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: Int);;
29204: Proc Set2DimStrArray(var arr: T2StringArray; asize1, asize2: Int);;
29205: Proc Set2DimIntArray(var arr: T2IntArray; asize1, asize2: Int);;
29206: Proc Set3DimIntArray(var arr: T3IntArray; asize1, asize2, asize3: Int);
29207: Proc Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: Int);
29208: Proc SetAsHandle( Format : Word; Value : THandle)
29209: Proc SetBounds(ALeft, ATop, AWidth, AHeight: Int)
29210: Proc SetCaptureControl(Control: TControl);
29211: Proc SetColumnAttributes
29212: Proc SetCookieField(Values:TStrings;const ADoman,APath:str;AExpires:TDateTime;ASecure:Bool)
29213: Proc SetCustomHeader( const Name, Value :Str)
29214: Proc SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
29215: Proc SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
29216: Proc SetFocus
29217: Proc SetFocus; virtual;
29218: Proc SetInitialState
29219: Proc SetKey
29220: Proc SetLastError(ErrorCode: Int)
29221: Proc SetLength;
29222: Proc SetLength2(var S:Str; NewLength: Int);
29223: Proc SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
29224: Proc SETOLE2MENUHANDLE( HANDLE : HMENU)
29225: Proc SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
29226: Proc SETPARAMS(APOSITION,AMIN,AMAX:Int)
29227: Proc SetParams1( UpdateKind : TUpdateKind);
29228: Proc SetPassword( const Password :Str)
29229: Proc SetPointer( Ptr : Pointer; Size : Longint)
29230: Proc SetPrimalityTest( const Method : TPrimalityTestMethod)
29231: Proc SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
29232: Proc SetProvider( Provider : TComponent)
29233: Proc SetProxy( const Proxy :Str)
29234: Proc SetPSResult( var PSResult : TPSResult; Value : TObject)
29235: Proc SetRange( const StartValues, EndValues : array of const)
29236: Proc SetRangeEnd
29237: Proc SetRate( const aPercent, aYear : Int)
29238: Proc SetRate(const aPercent, aYear: Int)
29239: Proc Set_ReportMemoryLeaksOnShutdown(abo:Bool)
29240: Proc SetSafeCallExceptionMsg( Msg :Str)
29241: Proc SETSELTEXTBUF(BUFFER:PCHAR)
29242: Proc SetSize( AWidth, AHeight : Int)
29243: Proc SetSize(NewSize:LongInt)
29244: Proc SetString(var s:Str; buffer: PChar; len: Int)
29245: Proc SetStrings( List : TStrings)
```

```
29246: Proc SetText( Text : PwideChar)
29247: Proc SetText(Text: PChar);
29248: Proc SetTextBuf( Buffer : PChar)
29249: Proc SETTEXTBUF(BUFFER:PCHAR)
29250: Proc SetTick( Value : Int)
29251: Proc SetTimeout( ATimeOut : Int)
29252: Proc SetTraceEvent( Event : TDBXTraceEvent)
29253: Proc SetUserName( const UserName :Str)
29254: Proc SetWallpaper( Path :Str);
29255: Proc ShellStyle1Click(Sender: TObject);
29256: Proc SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
29257: Proc ShowFileProperties( const FileName :Str)
29258: Proc ShowInclude1Click( Sender : TObject)
29259: Proc ShowInterfaces1Click( Sender : TObject)
29260: Proc ShowLastException1Click( Sender : TObject)
29261: Proc ShowMessage( const Msg :Str)
29262: Proc ShowMessageBig(const aText :Str);
29263: Proc ShowMessageBig2(const aText :Str; aautosize:Bool);
29264: Proc ShowMessageBig3(const aText :Str; fsize: byte; aautosize:Bool);
29265: Proc MsgBig(const aText :Str);        //alias
29266: Proc showmessage(mytext:Str);
29267: Proc ShowMessageFmt( const Msg :Str; Params : array of const)
29268: Proc ShowMessageFmt(const Msg:Str; Params: array of const))
29269: Proc ShowMessagePos( const Msg :Str; X, Y : Int)
29270: Proc ShowMessagePos(const Msg:Str; X: Int; Y: Int))
29271: Proc ShowSearchDialog( const Directory :Str)
29272: Proc ShowSpecChars1Click( Sender : TObject)
29273: Proc ShowBitmap(bmap: TBitmap);  //draw in a form!
29274: Proc ShredFile( const FileName :Str; Times : Int)
29275: Proc Shuffle(vQ: TStringList);
29276: Proc ShuffleList( var List : array of Int; Count : Int)
29277: Proc SimulateKeystroke( Key : byte; Shift : TShiftState)
29278: Proc SinCos( const Theta : Extended; var Sin, Cos : Extended)
29279: Proc SinCosE( X : Extended; out Sin, Cos : Extended)
29280: Proc Site( const ACommand :Str)
29281: Proc SkipEOL
29282: Proc Sleep( ATime :Card)
29283: Proc Sleep( milliseconds :Card)
29284: Func  SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
29285: Proc Slinenumbers1Click( Sender : TObject)
29286: Proc Sort
29287: Proc SortColorArray(ColorArray:TColorArr;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
29288: Proc SoundAlarm;  //beep seq
29289: Proc Speak(const sText:Str)   //async like voice
29290: Proc Speak2(const sText:Str)  //sync
29291: Proc Split(Str:Str;  SubStr:Str; List: TStrings);
29292: Proc SplitNameValue( const Line :Str; var Name, Value :Str)
29293: Proc SplitColumns( const AData :Str; AStrings : TStrings; const ADelim :Str)
29294: Proc SplitColumnsNoTrim(const AData:str; AStrings:TStrings;const ADelim :Str)
29295: Proc SplitLines( AData : PChar; ADataSize : Int; AStrings : TStrings)
29296: Proc SplitString( const AStr, AToken :Str; var VLeft, VRight :Str)
29297: Proc SQLSyntax1Click(Sender: TObject);
29298: Proc SRand( Seed : RNG_IntType)
29299: Proc Start
29300: Proc StartCount( var Counter : TJclCounter; const Compensate :Bool)
29301: Proc StartFileFinder3(spath,aext,searchstr:Str; arecursiv:Bool; reslist: TStringlist);
29302: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
29303: Proc StartTransaction( TransDesc : TTransactionDesc)
29304: Proc Status( var AStatusList : TStringList)
29305: Proc StatusBar1DblClick( Sender : TObject)
29306: Proc StepInto1Click( Sender : TObject)
29307: Proc StepIt
29308: Proc StepOut1Click( Sender : TObject)
29309: Proc Stop
29310: Proc stopmp3;
29311: Proc StartWeb(aurl:Str);
29312: Proc Str(aint: Int; astr:Str); //of system
29313: Proc StrDispose( Str : PChar)
29314: Proc StrDispose(Str: PChar)
29315: Proc StrReplace(var Str:Str; Old, New:Str);
29316: Proc StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch: Bool)
29317: Proc StretchDraw(const Rect: TRect; Graphic: TGraphic);
29318: Proc StringToBytes( Value :Str; Bytes : array of byte)
29319: Proc StrSet(c : Char; I : Int; var s :Str);
29320: Proc StrSplitP(const Delimiter: Char; Input:Str; const Strings: TStrings);
29321: Proc StructureMount( APath :Str)
29322: Proc STYLECHANGED(SENDER:TOBJECT)
29323: Proc Subselect( Node : TTreeNode; Validate :Bool)
29324: Proc Succ(X: int64);
29325: Proc SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares:Extended)
29326: Proc SwapChar(var X,Y: char);  //swapX follows
29327: Proc SwapFloats( var X, Y : Float)
29328: Proc SwapGrid(grd: TStringGrid);
29329: Proc SwapOrd( var I, J : Byte);
29330: Proc SwapOrd( var X, Y : Int)
29331: Proc SwapOrd1( var I, J : Shortint);
29332: Proc SwapOrd2( var I, J : Smallint);
29333: Proc SwapOrd3( var I, J : Word);
29334: Proc SwapOrd4( var I, J : Int);
```

```
29335: Proc SwapOrd5( var I, J :Card);
29336: Proc SwapOrd6( var I, J : Int64);
29337: Proc SymetricCompareFiles(const plaintext, replaintext:Str)
29338: Proc Synchronize1( Method : TMethod);
29339: Proc SyntaxCheck1Click(Sender: TObject);
29340: Proc SysFreeString(const S: WideString); stdcall;
29341: Proc TakeOver( Other : TLinearBitmap)
29342: Proc Talkln(const sText:Str)  //async voice
29343: Proc tbtn6resClick(Sender: TObject);
29344: Proc tbtnUseCaseClick( Sender : TObject)
29345: Proc TerminalStyle1Click(Sender: TObject);
29346: Proc Terminate
29347: Proc texSyntax1Click( Sender : TObject)
29348: Proc TextOut(X, Y: Int; Text:Str);
29349: Proc TextRect( Rect : TRect; X, Y : Int; const Text :Str);
29350: Proc TextRect(Rect: TRect; X: Int; Y: Int; const Text:Str);
29351: Proc TextRect1( var Rect : TRect; var Text :Str; TextFormat:TTextFormat);
29352: Proc TextStart
29353: Proc TILE
29354: Proc TimeStampToBytes( Value : TBcd; Bytes : array of byte)
29355: Proc TitleClick( Column : TColumn)
29356: Proc ToDo
29357: Proc Tone(500 hz, 10000 ms length);
29358: Proc toolbtnTutorialClick(Sender: TObject);
29359: Proc Trace1( AURL :Str; const AResponseContent : TStream);
29360: Proc TransferMode( ATransferMode : TIdFTPTransferMode)
29361: Proc Truncate
29362: Proc Tutorial101Click(Sender: TObject);
29363: Proc Tutorial10Statistics1Click(Sender: TObject);
29364: Proc Tutorial11Forms1Click(Sender: TObject);
29365: Proc Tutorial12SQL1Click(Sender: TObject);
29366: Proc tutorial1Click( Sender : TObject)
29367: Proc tutorial21Click( Sender : TObject)
29368: Proc tutorial31Click( Sender : TObject)
29369: Proc tutorial4Click( Sender : TObject)
29370: Proc Tutorial5Click( Sender : TObject)
29371: Proc Tutorial6Click(Sender: TObject);
29372: Proc Tutorial91Click(Sender: TObject);
29373: Proc UnhookSignal( RtlSigNum : Int; OnlyIfHooked :Bool)
29374: Proc UniqueString(var str: Ansistr)
29375: Proc UnloadLoadPackage(Module: HMODULE)
29376: Proc Unlock
29377: Proc UNMERGE( MENU : TMAINMENU)
29378: Proc UnRegisterChanges( Value : TChangeLink)
29379: Proc UnregisterConversionFamily( const AFamily : TConvFamily)
29380: Proc UnregisterConversionType( const AType : TConvType)
29381: Proc UnRegisterProvider( Prov : TCustomProvider)
29382: Proc UPDATE
29383: Proc UpdateBatch( AffectRecords : TAffectRecords)
29384: Proc UPDATECURSORPOS
29385: Proc UpdateFile
29386: Proc UpdateItems( FirstIndex, LastIndex : Int)
29387: Proc UpdateResponse( AResponse : TWebResponse)
29388: Proc UpdateScrollBar
29389: Proc UpdateView1Click( Sender : TObject)
29390: Proc UpdateExeResource(Const Source,Dest:str);  //!
29391: Proc Val(const s:Str; var n, z: Int)
29392: Proc VarArraySet(c : Variant; I : Int; var s : Variant);
29393: Proc VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
29394: Proc VariantAdd( const src : Variant; var dst : Variant)
29395: Proc VariantAnd( const src : Variant; var dst : Variant)
29396: Proc VariantArrayRedim( var V : Variant; High : Int)
29397: Proc VariantCast( const src : Variant; var dst : Variant; vt : Int)
29398: Proc VariantClear( var V : Variant)
29399: Proc VariantCpy( const src : Variant; var dst : Variant)
29400: Proc VariantDiv( const src : Variant; var dst : Variant)
29401: Proc VariantMod( const src : Variant; var dst : Variant)
29402: Proc VariantMul( const src : Variant; var dst : Variant)
29403: Proc VariantOr( const src : Variant; var dst : Variant)
29404: Proc VariantPutElement( var V : Variant; const data : Variant; i1 : Int);
29405: Proc VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : Int);
29406: Proc VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : Int);
29407: Proc VariantPutElement3(var V: Variant; const data : Variant; i1, i2, i3, i4 : Int);
29408: Proc VariantPutElement4(var V:Variant;const data: Variant; i1,i2,i3, i4, i5 : Int);
29409: Proc VariantShl( const src : Variant; var dst : Variant)
29410: Proc VariantShr( const src : Variant; var dst : Variant)
29411: Proc VariantSub( const src : Variant; var dst : Variant)
29412: Proc VariantXor( const src : Variant; var dst : Variant)
29413: Proc VarCastError;
29414: Proc VarCastError1( const ASourceType, ADestType : TVarType);
29415: Proc VarInvalidOp
29416: Proc VarInvalidNullOp
29417: Proc VarOverflowError( const ASourceType, ADestType : TVarType)
29418: Proc VarRangeCheckError( const ASourceType, ADestType : TVarType)
29419: Proc VarArrayCreateError
29420: Proc VarResultCheck( AResult : HRESULT);
29421: Proc VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
29422: Proc HandleConversionException( const ASourceType, ADestType : TVarType)
29423: Func VarTypeAsText( const AType : TVarType) :Str
```

```
29424: Proc Voice(const sText:Str) //async
29425: Proc Voice2(const sText:Str) //sync
29426: Proc WaitMiliSeconds( AMSec : word)
29427: Proc WaitMS( AMSec : word);
29428: Proc WebCamPic(picname:Str);  //eg: c:\mypic.png
29429: Proc WideAppend( var dst : WideString; const src : WideString)
29430: Proc WideAssign( var dst : WideString; var src : WideString)
29431: Proc WideDelete( var dst : WideString; index, count : Int)
29432: Proc WideFree( var s : WideString)
29433: Proc WideFromAnsi( var dst : WideString; const src : Ansistr)
29434: Proc WideFromPChar( var dst : WideString; src : PChar)
29435: Proc WideInsert( var dst : WideString; const src : WideString; index : Int)
29436: Proc WideSetLength( var dst : WideString; len : Int)
29437: Proc WideString2Stream( aWideString : WideString; oStream : TStream)
29438: Proc WideStringToBytes( Value : WideString; Bytes : array of byte)
29439: Proc WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
29440: Proc WinInet_HttpGet(const Url:Str; Stream:TStream);
29441: Proc HttpGet(const Url:Str; Stream:TStream);
29442: Proc WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : Int)
29443: Proc WordWrap1Click( Sender : TObject)
29444: Proc Write( const AOut :Str)
29445: Proc Write( Socket : TSocket)
29446: Proc Write(S:Str);
29447: Proc WriteBinaryStream( const Section, Name :Str; Value : TStream)
29448: Proc WriteBool( const Section, Ident :Str; Value :Bool)
29449: Proc WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow :Bool)
29450: Proc WriteBuffer(Buffer:str;Count:LongInt)
29451: Proc WriteCardinal( AValue :Card; const AConvert :Bool)
29452: Proc WriteChar( AValue : Char)
29453: Proc WriteDate( const Section, Name :Str; Value : TDateTime)
29454: Proc WriteDateTime( const Section, Name :Str; Value : TDateTime)
29455: Proc WriteFloat( const Section, Name :Str; Value : Double)
29456: Proc WriteHeader( AHeader : TStrings)
29457: Proc WriteInt( AValue : Int; const AConvert :Bool)
29458: Proc WriteInt( const Section, Ident :Str; Value : Longint)
29459: Proc WriteLn( const AOut :Str)
29460: Proc Println( AOut :Str)
29461: Proc Writeln(s:Str);
29462: Proc WriteLog( const FileName, LogLine :Str)
29463: Proc WriteRFCReply( AReply : TIdRFCReply)
29464: Proc WriteRFCStrings( AStrings : TStrings)
29465: Proc WriteSmallInt( AValue : SmallInt; const AConvert :Bool)
29466: Proc WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCnt:Bool;const ASize:Int)
29467: Proc WriteString( const Section, Ident, Value :Str)
29468: Proc WriteStrings( AValue : TStrings; const AWriteLinesCount :Bool)
29469: Proc WriteTime( const Section, Name :Str; Value : TDateTime)
29470: Proc WriteObjectResourceHeader( ObjStream, Output : TStream)
29471: Proc Write16bitResourceHeader(const AName: TBytes; DataSize : Int; Output:TStream)
29472: Proc Write32bitResourceHeader(const AName: TBytes; DataSize : Int; Output:TStream)
29473: Proc WriteDataSetToCSV(DataSet: TDataSet; FileName:Str);
29474: Proc WStrSet(c : AnyString; I : Int; var s : AnyString);
29475: Proc XMLSyntax1Click(Sender: TObject);
29476: Proc XOR_Streams2( Dest, Srce : TMemoryStream)
29477: Proc XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
29478: Proc ZeroFillStream( Stream : TMemoryStream)
29479: Proc XMLSyntax1Click(Sender: TObject);
29480: Proc ZeroMemory( Ptr : Pointer; Length : Longint)
29481: procedure(Control: TWinControl; Index: Int; Rect: TRect; State: Byte)
29482: procedure(Control: TWinControl; Index: Int; var Height: Int)
29483: procedure(Sender,Source: TObject; X, Y: Int; State: TDragState; var Accept:Bool)
29484: procedure(Sender,Source: TObject;X, Y: Int)
29485: procedure(Sender,Target: TObject; X, Y: Int)
29486: procedure(Sender:TObject; ASection, AWidth: Int)
29487: procedure(Sender:TObject; ScrollCode: TScrollCode;var ScrollPos: Int)
29488: procedure(Sender:TObject; Shift: TShiftState; X, Y: Int);
29489: procedure(Sender:TObject; var Action: TCloseAction)
29490: procedure(Sender:TObject; var CanClose:Bool)
29491: procedure(Sender:TObject; var Key: Char);
29492: ProcedureName ProcedureNames ProcedureParametersCursor @
29493:
29494: *************Now Constructors constructor *************
29495:  Size is: 1727 1640 1588
29496:  Attach( VersionInfoData : Pointer; Size : Int)
29497:  constructor Create( ABuckets : TBucketListSizes)
29498:  Create(ACallBackWnd : HWND)
29499:  Create(AClient : TCustomTaskDialog)
29500:  Create(AClient : TIdTelnet)
29501:  Create(ACollection : TCollection)
29502:  Create(ACollection : TFavoriteLinkItems)
29503:  Create(ACollection : TTaskDialogButtons)
29504:  Create(AConnection : TIdCustomHTTP)
29505:  Create(ACreateSuspended :Bool)
29506:  Create(ADataSet : TCustomSQLDataSet)
29507:  CREATE(ADATASET : TDATASET)
29508:  Create(Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
29509:  Create(AGrid : TCustomDBGrid)
29510:  Create(AGrid : TStringGrid; AIndex : Longint)
29511:  Create(AHTTP : TIdCustomHTTP)
29512:  Create(AListItems : TListItems)
```

```
29513:  Create(AOnBytesRemoved : TIdBufferBytesRemoved)
29514:  Create(AOnBytesRemoved : TIdBufferBytesRemoved)
29515:  Create(AOwner : TCommonCalendar)
29516:  Create(AOwner : TComponent)
29517:  CREATE(AOWNER : TCOMPONENT)
29518:  Create(AOwner : TCustomListView)
29519:  Create(AOwner : TCustomOutline)
29520:  Create(AOwner : TCustomRichEdit)
29521:  Create(AOwner : TCustomRichEdit; AttributeType : TAttributeType)
29522:  Create(AOwner : TCustomTreeView)
29523:  Create(AOwner : TIdUserManager)
29524:  Create(AOwner : TListItems)
29525:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
29526:  CREATE(AOWNER : TPERSISTENT)
29527:  Create(AOwner : TPersistent)
29528:  Create(AOwner : TTable)
29529:  Create(AOwner : TTreeNodes)
29530:  Create(AOwner : TWinControl; const ClassName :Str)
29531:  Create(AParent : TIdCustomHTTP)
29532:  Create(AParent : TUpdateTree; AResolver : TCustomResolver)
29533:  Create(AProvider : TBaseProvider)
29534:  Create(AProvider : TCustomProvider);
29535:  Create(AProvider : TDataSetProvider)
29536:  Create(ASocket : TCustomWinSocket; TimeOut : Longint)
29537:  Create(ASocket : TSocket)
29538:  Create(AStrings : TWideStrings)
29539:  Create(AToolBar : TToolBar)
29540:  Create(ATreeNodes : TTreeNodes)
29541:  Create(Autofill :Bool)
29542:  Create(AWebPageInfo : TAbstractWebPageInfo)
29543:  Create(AWebRequest : TWebRequest)
29544:  Create(Collection : TCollection)
29545:  Create(Collection : TIdMessageParts; ABody : TStrings)
29546:  Create(Collection : TIdMessageParts; const AFileName : TFileName)
29547:  Create(Column : TColumn)
29548:  Create(const AConvFamily : TConvFamily; const ADescription :Str)
29549:  Create(const AConvFamily  TConvFamily;const ADescription:str;const AFactor: Double)
29550:  Create(const AConvFamily:TConvFamily;const ADescription:str;const AToCommonProc,
       AFromComProc:TConversionProc)
29551:  Create(const AInitialState :Bool; const AManualReset :Bool)
29552:  Create(const ATabSet : TTabSet)
29553:  Create(const Compensate :Bool)
29554:  Create(const FileMap:TJclCustomFileMapping; Access,Size :Card; ViewOffset : Int64)
29555:  Create(const FileName :Str)
29556:  Create(const FileName :Str;FileMode:Card; const Name:str; Protect:Card; const MaximumSize : Int64; const
       SecAttr : PSecurityAttributes);
29557:  Create(const FileName :Str; FileMode : WordfmShareDenyWrite)
29558:  Create(const MaskValue :Str)
29559:  Create(const Name:str; Protect:Card;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
29560:  Create(const Prefix :Str)
29561:  Create(const sRegularExpression :Str; xFlags : TniRegularExpressionMatchFlags)
29562:  Create(const sRule :Str; xFlags : TniRegularExpressionMatchFlags)
29563:  Create(const sRule :Str; xFlags : TniRegularExpressionMatchFlags)
29564:  Create(CoolBar : TCoolBar)
29565:  Create(CreateSuspended :Bool; ASocket : TServerClientWinSocket)
29566:  Create(CreateSuspended :Bool; ASocket : TServerWinSocket)
29567:  Create(DataSet:TDataSet;const Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const
       FieldName:Widestring;DepFields:TBits;FieldMap:TFieldMap)
29568:  Create(DBCtrlGrid : TDBCtrlGrid)
29569:  Create(DSTableProducer : TDSTableProducer)
29570:  Create(DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
29571:  Create(ErrorCode : DBIResult)
29572:  Create(Field : TBlobField; Mode : TBlobStreamMode)
29573:  Create(Grid : TCustomDBGrid; ColumnClass : TColumnClass)
29574:  Create(HeaderControl : TCustomHeaderControl)
29575:  Create(HTTPRequest : TWebRequest)
29576:  Create(iStart : Int; sText :Str)
29577:  Create(iValue : Int)
29578:  Create(Kind : TMmTimerKind; Notification : TMmNotificationKind)
29579:  Create(MciErrNo : MCIERROR; const Msg :Str)
29580:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
29581:  Create(Message :Str; ErrorCode : DBResult)
29582:  Create(Msg :Str)
29583:  Create(NativeError, Context :Str; ErrCode, PrevError : Int; E : Exception)
29584:  Create(NativeError, Context :Str; ErrorCode, PreviousError : DBResult)
29585:  Create(oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
29586:  Create(oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
29587: Create(oSorce:TniRegularExpressState;oDestinat:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
29588:  Create(Owner: EDBEngineError; ErrorCode: DBIResult; NativeError:Longint;Message: PChar)
29589:  Create(Owner: TCustomComboBoxEx)
29590:  CREATE(OWNER  TINDEXDEFS; const NAME, FIELDS:Str; OPTIONS: TINDEXOPTIONS)
29591:  Create(Owner: TPersistent)
29592:  Create(Params: TStrings) Create( Size :Card)
29593:  Create(Socket: TSocket; ServerWinSocket : TServerWinSocket)
29594:  Create(StatusBar: TCustomStatusBar)
29595:  Create(WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
29596:  Create(WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
29597:  Create(AHandle:Int)
29598:  Create(AOwner: TComponent); virtual;
```

```
29599:  Create(const AURI :Str)
29600:  Create(FileName:str;Mode:Word)
29601:  Create(Instance:THandle;ResName:str;ResType:PChar)
29602:  Create(Stream : TStream)
29603:  Create1( ADataset : TDataset);
29604:  Create1(const FileHandle:THandle;const Name:str;Protect:Card;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
29605:  Create1( const FileName :Str; const AIndexOption : TJclMappedTextReaderIndex);
29606:  Create2( Other : TObject);
29607:  CreateAt(FileMap: TJclCustomFileMapping;Access,Size:Card;ViewOffset:Int64;Address:Pointer)
29608:  CreateError(const anErrCode: Int;const asReplyMessage:Str; const asErrorMessage:Str)
29609:  CreateFmt( MciErrNo : MCIERROR; const Msg :Str; const Args : array of const)
29610:  CreateFromId(Instance:THandle;ResId:Int;ResType:PChar)
29611:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
29612:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Int)
29613:  CreateRes(Ident : Int);
29614:  CreateRes(MciErrNo : MCIERROR; Ident : Int)
29615:  CreateRes(ResStringRec : PResStringRec);
29616:  CreateResHelp( Ident : Int; AHelpContext : Int);
29617:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Int);
29618:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
29619:  CreateSize( AWidth, AHeight : Int)
29620:  Open(const Name:Str; const InheritHandle:Bool; const DesiredAccess :Card)
29621:
29622: ------------------------------------------------------------------------------
29623: unit uPSI_MathMax;
29624: ------------------------------------------------------------------------------
29625:  CONSTS
29626:   Bernstein: Float = 0.2801694990238691330364364912307;  // Bernstein constant
29627:   Cbrt2: Float     = 1.2599210498948731647672106072782;  // CubeRoot(2)
29628:   Cbrt3: Float     = 1.4422495703074083823216383107801;  // CubeRoot(3)
29629:   Cbrt10: Float    = 2.1544346900318837217592935665194;  // CubeRoot(10)
29630:   Cbrt100: Float   = 4.6415888336127788924100763509194;  // CubeRoot(100)
29631:   CbrtPi: Float    = 1.4645918875615232630201425272638;  // CubeRoot(PI)
29632:   Catalan: Float   = 0.9159655941772190150546035149324;  // Catalan constant
29633:   PiJ:   Float     = 3.1415926535897932384626433832795;  // PI
29634:   PI:  Extended =  = 3.1415926535897932384626433832795);
29635:   PiOn2: Float     = 1.5707963267948966192313216916398;  // PI / 2
29636:   PiOn3: Float     = 1.0471975511965977461542144610932;  // PI / 3
29637:   PiOn4: Float     = 0.7853981633974483096156608458198 8; // PI / 4
29638:   Sqrt2: Float     = 1.4142135623730950488016887242097;  // Sqrt(2)
29639:   Sqrt3: Float     = 1.7320508075688772935274463415059;  // Sqrt(3)
29640:   Sqrt5: Float     = 2.2360679774997896964091736687313;  // Sqrt(5)
29641:   Sqrt10: Float    = 3.1622776601683793319988935444327;  // Sqrt(10)
29642:   SqrtPi: Float    = 1.7724538509055160272981674833411;  // Sqrt(PI)
29643:   Sqrt2Pi: Float   = 2.5066282746310005024157652848 11;  // Sqrt(2 * PI)
29644:   TwoPi: Float     = 6.2831853071795864769252867665 59;  // 2 * PI
29645:   ThreePi: Float   = 9.4247779607693797153879301498385;  // 3 * PI
29646:   Ln2: Float       = 0.6931471805599453094172321214581 8; // Ln(2)
29647:   Ln10: Float      = 2.3025850929940456840179914546844;  // Ln(10)
29648:   LnPi: Float      = 1.1447298858494001741434273513531;  // Ln(PI)
29649:   Log2J: Float     = 0.3010299956639811952137388947244 9; // Log10(2)
29650:   Log3: Float      = 0.4771212547196624372950279032551 2; // Log10(3)
29651:   LogPi: Float     = 0.4971498726941338543512682882909;  // Log10(PI)
29652:   LogE: Float      = 0.4342944819032518276511289189166 1; // Log10(E)
29653:   E: Float         = 2.7182818284590452353602874713527;  // Natural constant
29654:   hLn2Pi: Float    = 0.9189385332046727417803297364056 2; // Ln(2*PI)/2
29655:   inv2Pi: Float    = 0.1591549430918953357688837633725143620344596457404 6; // 0.5/Pi
29656:   TwoToPower63: Float    = 9223372036854775808.0;           // 2^63
29657:   GoldenMean: Float      = 1.6180339887498948482045868343656 38;  // GoldenMean
29658:   EulerMascheroni: Float = 0.5772156649015328606065120900824;  // Euler GAMMA
29659:   RadCor : Double        = 57.29577951308232;    {number of degrees in a radian}
29660:   StDelta         : Extended = 0.00001;  {delta for difference equations}
29661:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
29662:   StMaxIterations : Int  = 100;      {max attempts for convergence}
29663:
29664: Proc SIRegister_StdConvs(CL: TPSPascalCompiler);
29665: begin
29666:  MetersPerInch = 0.0254; // [1]
29667:  MetersPerFoot = MetersPerInch * 12;
29668:  MetersPerYard = MetersPerFoot * 3;
29669:  MetersPerMile = MetersPerFoot * 5280;
29670:  MetersPerNauticalMiles = 1852;
29671:  MetersPerAstronomicalUnit = 1.49598E11; // [4]
29672:  MetersPerLightSecond = 2.99792458E8; // [5]
29673:  MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
29674:  MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
29675:  MetersPerCubit = 0.4572; // [6][7]
29676:  MetersPerFathom = MetersPerFoot * 6;
29677:  MetersPerFurlong = MetersPerYard * 220;
29678:  MetersPerHand = MetersPerInch * 4;
29679:  MetersPerPace = MetersPerInch * 30;
29680:  MetersPerRod = MetersPerFoot * 16.5;
29681:  MetersPerChain = MetersPerRod * 4;
29682:  MetersPerLink = MetersPerChain / 100;
29683:  MetersPerPoint = MetersPerInch * 0.013837; // [7]
29684:  MetersPerPica = MetersPerPoint * 12;
29685:
29686:  SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
```

```
29687:  SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
29688:  SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
29689:  SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
29690:  SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
29691:  SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
29692:  CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
29693:  CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
29694:  CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
29695:  CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
29696:  CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
29697:  CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
29698:  CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
29699:  CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
29700:
29701:  CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
29702:  CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
29703:  CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
29704:  CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
29705:  CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
29706:  CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
29707:  CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
29708:  CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
29709:  CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
29710:  CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
29711:  CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
29712:  CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
29713:  CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
29714:  CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
29715:  CubicMetersPerUKGallon = 0.00454609; // [2][7]
29716:  CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
29717:  CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
29718:  CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
29719:  CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
29720:  CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
29721:  CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
29722:  CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
29723:  CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
29724:
29725:  GramsPerPound = 453.59237; // [1][7]
29726:  GramsPerDrams = GramsPerPound / 256;
29727:  GramsPerGrains = GramsPerPound / 7000;
29728:  GramsPerTons = GramsPerPound * 2000;
29729:  GramsPerLongTons = GramsPerPound * 2240;
29730:  GramsPerOunces = GramsPerPound / 16;
29731:  GramsPerStones = GramsPerPound * 14;
29732:
29733:  MaxAngle 9223372036854775808.0;
29734:  MaxTanH 5678.2617031470719747459655389854);
29735:  MaxFactorial( 1754);
29736:  MaxFloatingPoint(1.189731495357231765085759326628E+4932);
29737:  MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
29738:  MaxTanH( 354.8913564466919984216228461865 9);
29739:  MaxFactorial'LongInt'( 170);
29740:  MaxFloatingPointD(1.797693134862315907729305190789E+308);
29741:  MinFloatingPointD(2.225073858507201383090232717332 4E-308);
29742:  MaxTanH( 44.36141955583649980270285577332 3);
29743:  MaxFactorial'LongInt'( 33);
29744:  MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
29745:  MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
29746:  PiExt( 3.14159265358979323846264338327 95);
29747:  RatioDegToRad( PiExt / 180.0);
29748:  RatioGradToRad( PiExt / 200.0);
29749:  RatioDegToGrad( 200.0 / 180.0);
29750:  RatioGradToDeg( 180.0 / 200.0);
29751:  Crc16PolynomCCITT'LongWord $1021);
29752:  Crc16PolynomIBM'LongWord $8005);
29753:  Crc16Bits'LongInt'( 16);
29754:  Crc16Bytes'LongInt'( 2);
29755:  Crc16HighBit'LongWord $8000);
29756:  NotCrc16HighBit','LongWord $7FFF);
29757:  Crc32PolynomIEEE','LongWord $04C11DB7);
29758:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
29759:  Crc32Koopman','LongWord $741B8CD7);
29760:  Crc32Bits','LongInt'( 32);
29761:  Crc32Bytes','LongInt'( 4);
29762:  Crc32HighBit','LongWord $80000000);
29763:  NotCrc32HighBit','LongWord $7FFFFFFF);
29764:
29765:   MinByte       = Low(Byte);
29766:   MaxByte       = High(Byte);
29767:   MinWord       = Low(Word);
29768:   MaxWord       = High(Word);
29769:   MinShortInt   = Low(ShortInt);
29770:   MaxShortInt   = High(ShortInt);
29771:   MinSmallInt   = Low(SmallInt);
29772:   MaxSmallInt   = High(SmallInt);
29773:   MinLongWord   = LongWord(Low(LongWord));
29774:   MaxLongWord   = LongWord(High(LongWord));
29775:   MinLongInt    = LongInt(Low(LongInt));
```

```
29776:   MaxLongInt     = LongInt(High(LongInt));
29777:   MinInt64       = Int64(Low(Int64));
29778:   MaxInt64       = Int64(High(Int64));
29779:   MinInt     = Int(Low(Int));
29780:   MaxInt     = Int(High(Int));
29781:   MinCardinal    = Cardinal(Low(Cardinal));
29782:   MaxCardinal    = Cardinal(High(Cardinal));
29783:   MinNativeUInt = NativeUInt(Low(NativeUInt));
29784:   MaxNativeUInt = NativeUInt(High(NativeUInt));
29785:   MinNativeInt  = NativeInt(Low(NativeInt));
29786:   MaxNativeInt  = NativeInt(High(NativeInt));
29787:  Func CosH( const Z : Float) : Float;
29788:  Func SinH( const Z : Float) : Float;
29789:  Func TanH( const Z : Float) : Float;
29790:
29791:  //***********from DMath.Dll Lib of types.inc in source\dmath_dll
29792:   InvLn2      = 1.44269504088896340736;  { 1/Ln(2) }
29793:   InvLn10     = 0.43429448190325182765;  { 1/Ln(10) }
29794:   TwoPi       = 6.28318530717958647693;  { 2*Pi }
29795:   PiDiv2      = 1.57079632679489661923;  { Pi/2 }
29796:   SqrtPi      = 1.77245385090551602730;  { Sqrt(Pi) }
29797:   Sqrt2Pi     = 2.50662827463100050242;  { Sqrt(2*Pi) }
29798:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
29799:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
29800:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
29801:   Sqrt2       = 1.41421356237309504880;  { Sqrt(2) }
29802:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
29803:   Gold        = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
29804:   CGold       = 0.38196601125010515179;  { 2 - GOLD }
29805:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
29806:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
29807:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
29808:   MaxLog = 709.7827128933840;
29809:   MinLog = -708.3964185322641;
29810:   MaxFac = 170;
29811:   MaxGam = 171.624376956302;
29812:   MaxLgm = 2.556348E+305;
29813:   SingleCompareDelta   = 1.0E-34;
29814:   DoubleCompareDelta   = 1.0E-280;
29815:   {$IFDEF CLR}
29816:   ExtendedCompareDelta = DoubleCompareDelta;
29817:   {$ELSE}
29818:   ExtendedCompareDelta = 1.0E-4400;
29819:   {$ENDIF}
29820:   Bytes1KB  = 1024;
29821:   Bytes1MB  = 1024 * Bytes1KB;
29822:   Bytes1GB  = 1024 * Bytes1MB;
29823:   Bytes64KB = 64 * Bytes1KB;
29824:   Bytes64MB = 64 * Bytes1MB;
29825:   Bytes2GB  = 2 * LongWord(Bytes1GB);
29826:  clBlack32', $FF000000 ));
29827:  clDimGray32', $FF3F3F3F ));
29828:  clGray32', $FF7F7F7F ));
29829:  clLightGray32', $FFBFBFBF ));
29830:  clWhite32', $FFFFFFFF ));
29831:  clMaroon32', $FF7F0000 ));
29832:  clGreen32', $FF007F00 ));
29833:  clOlive32', $FF7F7F00 ));
29834:  clNavy32', $FF00007F ));
29835:  clPurple32', $FF7F007F ));
29836:  clTeal32', $FF007F7F ));
29837:  clRed32', $FFFF0000 ));
29838:  clLime32', $FF00FF00 ));
29839:  clYellow32', $FFFFFF00 ));
29840:  clBlue32', $FF0000FF ));
29841:  clFuchsia32', $FFFF00FF ));
29842:  clAqua32', $FF00FFFF ));
29843:  clAliceBlue32', $FFF0F8FF ));
29844:  clAntiqueWhite32', $FFFAEBD7 ));
29845:  clAquamarine32', $FF7FFFD4 ));
29846:  clAzure32', $FFF0FFFF ));
29847:  clBeige32', $FFF5F5DC ));
29848:  clBisque32', $FFFFE4C4 ));
29849:  clBlancheDalmond32', $FFFFEBCD ));
29850:  clBlueViolet32', $FF8A2BE2 ));
29851:  clBrown32', $FFA52A2A ));
29852:  clBurlyWood32', $FFDEB887 ));
29853:  clCadetblue32', $FF5F9EA0 ));
29854:  clChartReuse32', $FF7FFF00 ));
29855:  clChocolate32', $FFD2691E ));
29856:  clCoral32', $FFFF7F50 ));
29857:  clCornFlowerBlue32', $FF6495ED ));
29858:  clCornSilk32', $FFFFF8DC ));
29859:  clCrimson32', $FFDC143C ));
29860:  clDarkBlue32', $FF00008B ));
29861:  clDarkCyan32', $FF008B8B ));
29862:  clDarkGoldenRod32', $FFB8860B ));
29863:  clDarkGray32', $FFA9A9A9 ));
29864:  clDarkGreen32', $FF006400 ));
```

```
29865:  clDarkGrey32', $FFA9A9A9 ));
29866:  clDarkKhaki32', $FFBDB76B ));
29867:  clDarkMagenta32', $FF8B008B ));
29868:  clDarkOliveGreen32', $FF556B2F ));
29869:  clDarkOrange32', $FFFF8C00 ));
29870:  clDarkOrchid32', $FF9932CC ));
29871:  clDarkRed32', $FF8B0000 ));
29872:  clDarkSalmon32', $FFE9967A ));
29873:  clDarkSeaGreen32', $FF8FBC8F ));
29874:  clDarkSlateBlue32', $FF483D8B ));
29875:  clDarkSlateGray32', $FF2F4F4F ));
29876:  clDarkSlateGrey32', $FF2F4F4F ));
29877:  clDarkTurquoise32', $FF00CED1 ));
29878:  clDarkViolet32', $FF9400D3 ));
29879:  clDeepPink32', $FFFF1493 ));
29880:  clDeepSkyBlue32', $FF00BFFF ));
29881:  clDodgerBlue32', $FF1E90FF ));
29882:  clFireBrick32', $FFB22222 ));
29883:  clFloralWhite32', $FFFFFAF0 ));
29884:  clGainsBoro32', $FFDCDCDC ));
29885:  clGhostWhite32', $FFF8F8FF ));
29886:  clGold32', $FFFFD700 ));
29887:  clGoldenRod32', $FFDAA520 ));
29888:  clGreenYellow32', $FFADFF2F ));
29889:  clGrey32', $FF808080 ));
29890:  clHoneyDew32', $FFF0FFF0 ));
29891:  clHotPink32', $FFFF69B4 ));
29892:  clIndianRed32', $FFCD5C5C ));
29893:  clIndigo32', $FF4B0082 ));
29894:  clIvory32', $FFFFFFF0 ));
29895:  clKhaki32', $FFF0E68C ));
29896:  clLavender32', $FFE6E6FA ));
29897:  clLavenderBlush32', $FFFFF0F5 ));
29898:  clLawnGreen32', $FF7CFC00 ));
29899:  clLemonChiffon32', $FFFFFACD ));
29900:  clLightBlue32', $FFADD8E6 ));
29901:  clLightCoral32', $FFF08080 ));
29902:  clLightCyan32', $FFE0FFFF ));
29903:  clLightGoldenRodYellow32', $FFFAFAD2 ));
29904:  clLightGreen32', $FF90EE90 ));
29905:  clLightGrey32', $FFD3D3D3 ));
29906:  clLightPink32', $FFFFB6C1 ));
29907:  clLightSalmon32', $FFFFA07A ));
29908:  clLightSeagreen32', $FF20B2AA ));
29909:  clLightSkyblue32', $FF87CEFA ));
29910:  clLightSlategray32', $FF778899 ));
29911:  clLightSlategrey32', $FF778899 ));
29912:  clLightSteelblue32', $FFB0C4DE ));
29913:  clLightYellow32', $FFFFFFE0 ));
29914:  clLtGray32', $FFC0C0C0 ));
29915:  clMedGray32', $FFA0A0A4 ));
29916:  clDkGray32', $FF808080 ));
29917:  clMoneyGreen32', $FFC0DCC0 ));
29918:  clLegacySkyBlue32', $FFA6CAF0 ));
29919:  clCream32', $FFFFFBF0 ));
29920:  clLimeGreen32', $FF32CD32 ));
29921:  clLinen32', $FFFAF0E6 ));
29922:  clMediumAquamarine32', $FF66CDAA ));
29923:  clMediumBlue32', $FF0000CD ));
29924:  clMediumOrchid32', $FFBA55D3 ));
29925:  clMediumPurple32', $FF9370DB ));
29926:  clMediumSeaGreen32', $FF3CB371 ));
29927:  clMediumSlateBlue32', $FF7B68EE ));
29928:  clMediumSpringGreen32', $FF00FA9A ));
29929:  clMediumTurquoise32', $FF48D1CC ));
29930:  clMediumVioletRed32', $FFC71585 ));
29931:  clMidnightBlue32', $FF191970 ));
29932:  clMintCream32', $FFF5FFFA ));
29933:  clMistyRose32', $FFFFE4E1 ));
29934:  clMoccasin32', $FFFFE4B5 ));
29935:  clNavajoWhite32', $FFFFDEAD ));
29936:  clOldLace32', $FFFDF5E6 ));
29937:  clOliveDrab32', $FF6B8E23 ));
29938:  clOrange32', $FFFFA500 ));
29939:  clOrangeRed32', $FFFF4500 ));
29940:  clOrchid32', $FFDA70D6 ));
29941:  clPaleGoldenRod32', $FFEEE8AA ));
29942:  clPaleGreen32', $FF98FB98 ));
29943:  clPaleTurquoise32', $FFAFEEEE ));
29944:  clPaleVioletred32', $FFDB7093 ));
29945:  clPapayaWhip32', $FFFFEFD5 ));
29946:  clPeachPuff32', $FFFFDAB9 ));
29947:  clPeru32', $FFCD853F ));
29948:  clPlum32', $FFDDA0DD ));
29949:  clPowderBlue32', $FFB0E0E6 ));
29950:  clRosyBrown32', $FFBC8F8F ));
29951:  clRoyalBlue32', $FF4169E1 ));
29952:  clSaddleBrown32', $FF8B4513 ));
29953:  clSalmon32', $FFFA8072 ));
```

```
29954:  clSandyBrown32', $FFF4A460 ));
29955:  clSeaGreen32', $FF2E8B57 ));
29956:  clSeaShell32', $FFFFF5EE ));
29957:  clSienna32', $FFA0522D ));
29958:  clSilver32', $FFC0C0C0 ));
29959:  clSkyblue32', $FF87CEEB ));
29960:  clSlateBlue32', $FF6A5ACD ));
29961:  clSlateGray32', $FF708090 ));
29962:  clSlateGrey32', $FF708090 ));
29963:  clSnow32', $FFFFFAFA ));
29964:  clSpringgreen32', $FF00FF7F ));
29965:  clSteelblue32', $FF4682B4 ));
29966:  clTan32', $FFD2B48C ));
29967:  clThistle32', $FFD8BFD8 ));
29968:  clTomato32', $FFFF6347 ));
29969:  clTurquoise32', $FF40E0D0 ));
29970:  clViolet32', $FFEE82EE ));
29971:  clWheat32', $FFF5DEB3 ));
29972:  clWhitesmoke32', $FFF5F5F5 ));
29973:  clYellowgreen32', $FF9ACD32 ));
29974:  clTrWhite32', $7FFFFFFF ));
29975:  clTrBlack32', $7F000000 ));
29976:  clTrRed32', $7FFF0000 ));
29977:  clTrGreen32', $7F00FF00 ));
29978:  clTrBlue32', $7F0000FF ));
29979:   // Fixed point math constants
29980:   FixedOne = $10000;  FixedHalf = $7FFF;
29981:   FixedPI  = Round(PI * FixedOne);
29982:   FixedToFloat = 1/FixedOne;
29983:
29984:  Special Types
29985: *******************************************************
29986:   type Complex = record //for complex numbers
29987:     X, Y : Float; end;
29988:   type TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R :'
29989:    + Float; Theta : Float; end;
29990:   type TVector     = array of Float;
29991:   TIntVector  = array of Int;
29992:   TCompVector = array of Complex;
29993:   TBoolVector = array of Boolean;
29994:   TStrVector  = array of String;
29995:   TMatrix     = array of TVector;
29996:   TIntMatrix  = array of TIntVector;
29997:   TCompMatrix = array of TCompVector;
29998:   TBoolMatrix = array of TBoolVector;
29999:   TStrMatrix  = array of TStrVector;
30000:   TByteArray = array[0..32767] of byte; !
30001:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
30002:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
30003:   T2StringArray = array of array of string;
30004:   T2IntArray = array of array of Int;
30005:   AddTypeS('INT_PTR', 'Int
30006:   AddTypeS('LONG_PTR', 'Int
30007:   AddTypeS('UINT_PTR', 'Cardinal
30008:   AddTypeS('ULONG_PTR', 'Cardinal
30009:   AddTypeS('DWORD_PTR', 'ULONG_PTR
30010:   TIntDynArray', 'array of Int
30011:   TCardinalDynArray', 'array of Cardinal
30012:   TWordDynArray', 'array of Word
30013:   TSmallIntDynArray', 'array of SmallInt
30014:   TByteDynArray', 'array of Byte
30015:   TShortIntDynArray', 'array of ShortInt
30016:   TInt64DynArray', 'array of Int64
30017:   TLongWordDynArray', 'array of LongWord
30018:   TSingleDynArray', 'array of Single
30019:   TDoubleDynArray', 'array of Double
30020:   TBooleanDynArray', 'array of Boolean
30021:   TStringDynArray', 'array of string
30022:   TWideStringDynArray', 'array of WideString
30023:   TDynByteArray     = array of Byte;
30024:   TDynShortintArray = array of Shortint;
30025:   TDynSmallintArray = array of Smallint;
30026:   TDynWordArray     = array of Word;
30027:   TDynIntArray  = array of Int;
30028:   TDynLongintArray  = array of Longint;
30029:   TDynCardinalArray = array of Cardinal;
30030:   TDynInt64Array    = array of Int64;
30031:   TDynExtendedArray = array of Extended;
30032:   TDynDoubleArray   = array of Double;
30033:   TDynSingleArray   = array of Single;
30034:   TDynFloatArray    = array of Float;
30035:   TDynPointerArray  = array of Pointer;
30036:   TDynStringArray   = array of string;
30037:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
30038:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
30039:   TSynSearchOptions = set of TSynSearchOption;
30040:   TFloat = single
30041:   Float = double
30042:
```

```
30043: /* Project:IFSI_WinForm1puzzle.pas BaseInclude RunTimeLib for maXbox *: pas_includebox.inc
30044: ---------------------------------------------------------------------------
30045: Proc drawPolygon(vPoints: TXYVector; cFrm: TForm);
30046: Proc drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: Int);
30047: Proc SaveCanvas(vCanvas: TCanvas; FileName:Str);
30048: Proc SaveCanvas2(vCanvas: TCanvas; FileName:Str);
30049: Func CheckStringSum(vstring:Str): Int;
30050: Func HexToInt(HexNum:Str): LongInt;
30051: Func IntToBin(Int: Int):Str;
30052: Func BinToInt(Binary:Str): Int;
30053: Func HexToBin(HexNum:Str):Str; external2
30054: Func BinToHex(Binary:Str):Str;
30055: Func IntToFloat(i: Int): double;
30056: Func AddThousandSeparator(S:Str; myChr: Char):Str;
30057: Func Max3(const X,Y,Z: Int): Int;
30058: Proc Swap(var X,Y: char); // faster without inline
30059: Proc ReverseString(var S:Str);
30060: Func CharToHexStr(Value: Char):Str;
30061: Func CharToUniCode(Value: Char):Str;
30062: Func Hex2Dec(Value: Str002): Byte;
30063: Func HexStrCodeToStr(Value:Str):Str;
30064: Func HexToStr(i: Int; value:Str):Str;
30065: Func UniCodeToStr(Value:Str):Str;
30066: Func CRC16(statement:Str):Str;
30067: Func SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2:Str):Str;
30068: Proc SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr:Str);
30069: Proc SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr:Str; offset: Int);
30070: Proc ExecuteCommand(executeFile, paramstring:Str);
30071: Proc ShellExecuteAndWait(executeFile, paramstring:Str);
30072: Func ExecConsoleApp(const AppName, Parameters:Str; AppOutput: TStrings): DWORD;
30073: Proc SearchAndOpenDoc(vfilenamepath:Str);
30074: Proc ShowInterfaces(myFile:Str);
30075: Func Fact2(av: Int): extended;
30076: Func BoolToStr(B:Bool):Str;
30077: Func GCD(x, y : LongInt) : LongInt;
30078: Func LCM(m,n: longint): longint;
30079: Func GetASCII:Str;
30080: Func GetItemHeight(Font: TFont): Int;
30081: Func myPlaySound(s: pchar; flag,syncflag: Int):Bool;
30082: Func myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
30083: Func getHINSTANCE: longword;
30084: Func getHMODULE: longword;
30085: Func GetASCII:Str;
30086: Func ByteIsOk(const AByte:Str; var VB: Byte):Bool;
30087: Func WordIsOk(const AWord:Str; var VW: Word):Bool;
30088: Func TwentyFourBitValueIsOk(const AValue:Str; var VI: Int):Bool;
30089: Func LongIsOk(const ALong:Str; var VC:Card):Bool;
30090: Func SafeStr(const s:Str):Str;
30091: Func ExtractUrlPath(const FileName:Str):Str;
30092: Func ExtractUrlName(const FileName:Str):Str;
30093: Func IsInternet:Bool;
30094: Func RotateLeft1Bit_u32( Value: uint32): uint32;
30095: Proc LinearRegression(const KnownY:array of Double;const KnownX:array of Double;NData:Int;var
       LF:TStLinEst;ErrorStats:Bool);
30096: Proc getEnvironmentInfo;
30097: Proc AntiFreeze;
30098: Func GetCPUSpeed: Double;
30099: Func IsVirtualPcGuest :Bool;
30100: Func IsVmWareGuest :Bool;
30101: Proc StartSerialDialog;
30102: Func IsWoW64:Bool;
30103: Func IsWow64String(var s:Str):Bool;
30104: Proc StartThreadDemo;
30105: Func RGB(R,G,B: Byte): TColor;
30106: Func Sendln(amess:Str):Bool;
30107: Proc maXbox;
30108: Func AspectRatio(aWidth, aHeight: Int):Str;
30109: Func wget(aURL, afile:Str):Bool;
30110: Proc PrintList(Value: TStringList);
30111: Proc PrintImage(aValue: TBitmap; Style: TBitmapStyle);
30112: Proc getEnvironmentInfo;
30113: Proc AntiFreeze;
30114: Func getBitmap(apath:Str): TBitmap;
30115: Proc ShowMessageBig(const aText :Str);
30116: Func YesNoDialog(const ACaption, AMsg:Str):Bool;
30117: Proc SetArrayLength2String(arr: T2StringArray; asize1, asize2: Int);
30118: Proc SetArrayLength2Int(arr: T2IntArray; asize1, asize2: Int);
30119: //Func myStrToBytes(const Value:Str): TBytes;
30120: //Func myBytesToStr(const Value: TBytes):Str;
30121: Func SaveAsExcelFile(AGrid:TStringGrid;ASheetName,AFileName:str;open:boolean):Bool;
30122: Func getBitmap(apath:Str): TBitmap;
30123: Proc ShowMessageBig(const aText :Str);
30124: Func StrToBytes(const Value:Str): TBytes;
30125: Func BytesToStr(const Value: TBytes):Str;
30126: Func SaveAsExcelFile(AGrid: TStringGrid;ASheetName,AFileName:Str;open:boolean):Bool;
30127: Func ReverseDNSLookup(const IPAdrs:str;const DNSServer:str;Timeout,Retris:Int;var HostName:str):Bool;
30128: Func FindInPaths(const fileName, paths :Str) :Str;
30129: Proc initHexArray(var hexn: THexArray);
30130: Func josephusG(n,k: Int; var graphout:Str): Int;
```

```
30131: Func isPowerof2(num: int64):Bool;
30132: Func powerOf2(exponent: Int): int64;
30133: Func getBigPI:Str;
30134: Proc MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
30135:  Func GetASCIILine:Str;
30136: Proc MakeComplexSound(N:Int{stream # to use};freqlist:TStrings;Duration{mSec}: Int;
30137:                          pinknoise:Bool; shape: Int; Volume: TVolumeLevel);
30138: Proc SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:Int);
30139: Proc AddComplexSoundObjectToList(newf,newp,newa,news:Int; freqlist: TStrings);
30140: Func mapfunc(ax, in_min, in_max, out_min, out_max: Int): Int;
30141: Func mapmax(ax, in_min, in_max, out_min, out_max: Int): Int;
30142: Func isKeypressed:Bool;
30143: Func Keypress:Bool;
30144: Proc StrSplitP(const Delimiter: Char; Input:Str; const Strings: TStrings);
30145: Func ReadReg(Base: HKEY; KeyName, ValueName:Str):Str;
30146: Func ReadRegistry(Base: HKEY; KeyName, ValueName:Str):Str;
30147: Func GetOSName:Str;
30148: Func GetOSVersion:Str;
30149: Func GetOSNumber:Str;
30150: Func getEnvironmentString:Str;
30151: Proc StrReplace(var Str:Str; Old, New:Str);
30152: Proc SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
30153: Func getTeamViewerID:Str;
30154: Proc RecurseDirectory(Dir:str; IncludeSubs:boolean; callback:TFileCallbackProcedure);
30155: Proc RecurseDirectory2(Dir :Str; IncludeSubs :Bool);
30156: Proc WinInet_HttpGet(const Url:Str; Stream:TStream);
30157: Proc GetQrCode2(Width,Height: Word;Correct_Level:str;const Data:str;apath:str);
30158: Func StartSocketService:Bool;
30159: Proc StartSocketServiceForm;
30160: Func GetFileList(FileList: TStringlist; apath:Str): TStringlist;
30161: Func GetFileList1(apath:Str): TStringlist;
30162: Proc LetFileList(FileList: TStringlist; apath:Str);
30163: Proc StartWeb(aurl:Str);
30164: Func GetTodayFiles(startdir, amask:Str): TStringlist;
30165: Func PortTCPIsOpen(dwPort : Word; ipAddressStr:Str):Bool;
30166: Func JavahashCode(val:Str): Int;
30167: Proc PostKeyEx32(key: Word; const shift: TShiftState; specialkey:Bool);
30168: Proc SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName:Str);
30169: Proc HideWindowForSeconds(secs: Int);   {//3 seconds}
30170: Proc HideWindowForSeconds2(secs: Int; apphandle, aself: TForm);   {//3 seconds}
30171: Proc ConvertToGray(Cnv: TCanvas);
30172: Func GetFileDate(aFile:str; aWithTime:Boolean):str;
30173: Proc ShowMemory;
30174: Func ShowMemory2:Str;
30175: Func getHostIP:Str;
30176: Proc ShowBitmap(bmap: TBitmap);
30177: Func GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
30178: Func CreateDBGridForm(dblist: TStringList): TListbox;
30179: Func isService:Bool;
30180: Func isApplication:Bool;
30181: Func isTerminalSession:Bool;
30182:  Func SetPrivilege(privilegeName:Str; enable:Bool):Bool;
30183:  Proc getScriptandRunAsk;
30184:  Proc getScriptandRun(ascript:Str);
30185:  Func VersionCheckAct:Str;
30186:  Proc getBox(aurl, extension:Str);
30187:  Func CheckBox:Str;
30188:  Func isNTFS:Bool;
30189:  //Proc doWebCamPic;
30190:  Proc doWebCamPic(picname:Str);
30191:  Func readm:Str;
30192:  Proc getGEOMapandRunAsk;     //APIKey needed
30193:  Func GetMapX(C_form,apath:Str; const Data:Str):Bool;
30194:  Proc GetGEOMap(C_form,apath:Str; const Data:Str);
30195:  Func GetMapXGeoReverse(C_form:Str; const lat,long:Str):Str;
30196:  //Func RoundTo(const AValue:Extended; const ADigit: TRoundToEXRangeExtended):Extended;
30197:  Func GetGeocodeCoord(C_form:Str; const data:str; atxt:Bool):Str;
30198:  Func GetGeoCoord(C_form:Str; const data:str; atxt:Bool):Str;
30199:     ex.: writeln(GetGeoCoord('xml','church cefalu sicily',true))
30200:  Func DownloadFile(SourceFile, DestFile:Str):Bool;
30201:  Func DownloadFileOpen(SourceFile, DestFile:Str):Bool;
30202:  Func OpenMap(const Data:Str):Bool;
30203:  Func GetGeoCode(C_form,apath:Str; const data:Str; sfile:Bool):Str;
30204:  Func getFileCount(amask:Str): Int;
30205:  Func CoordinateStr(Idx: Int; PosInSec: Double; PosLn: TNavPos):Str;
30206:  Proc Debugln(DebugLOGFILE:Str; E:Str);
30207:  Func IntToFloat(i: Int): double;
30208:  Func AddThousandSeparator(S:Str; myChr: Char):Str;
30209:  Func mymciSendString(cmd: PChar; ret: PChar; len: Int; callback: Int):Card;
30210:  Func RoundTime(ADate:Str; Rounding: Int; bRound:Bool):Str;
30211:  Func DynamicDllCallName(Dll:Str; const Name:Str; HasResult:Bool; var Returned:Card; const Parameters:
       array of Int):Bool;;
30212:  Func DynamicDllCall(Dll:Str; const Name:str; HasResult:Boolean; var Returned:Card; const Parameters:
       array of Int):Bool;;
30213:  Func GetSpecialFolderID( const Path :Str) : Int;
30214:  Func GetSpecialFolderPath2( FolderID : Int) :Str;
30215: End C:\maXbook\maxbox3\mX3999\maxbox3\source\IFSI_WinForm1puzzle.pas File loaded
30216:
30217: // News of 3.9.8 up
```

```
30218: Halt-Stop Program in Menu,WebServer2,Stop Event Recompile,
30219: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
30220: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
30221: JvChart - TJvChart Component - 2009 Public
30222: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
30223: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
30224: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
30225: DMath DLL included incl. Demos
30226: Interface Navigator menu/View/Intf Navigator
30227: Unit Explorer menu/Debug/Units Explorer
30228: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
30229: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
30230: Script History to 9 Files WebServer light /Options/Addons/WebServer
30231: Full Text Finder, JVSimLogic Simulator Package
30232: Halt-Stop Program in Menu, WebServer2, Stop Event ,
30233: Conversion Routines, Prebuild Forms, CodeSearch
30234: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
30235: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
30236: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
30237: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
30238: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
30239: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
30240: IDE Reflection API, Session Service Shell S3
30241: additional SynEdit API, isKeyPressed Routine,Bookmarks,OpenToolsAPI Catalog (OTAC)
30242: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
30243: arduino map() function, PMRandom Generator
30244: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
30245: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
30246: REST Test Lib, Multilang Component, Forth Interpreter
30247: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
30248: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
30249: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
30250: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
30251: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
30252: QRCode Service, add more CFunctions like CDateTime of Synapse
30253: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
30254: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
30255: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
30256: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
30257: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
30258: BOLD Package, Indy Package5, maTRIx. MATHEMAX
30259: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
30260: emax layers: system-package-component-unit-class-function-block
30261: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
30262: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
30263: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
30264: OpenGL Game Demo: ..Options/Add Ons/Reversi
30265: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
30266: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
30267: 7% performance gain (hot spot profiling)
30268: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
30269: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
30270: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
30271: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
30272:
30273: add routines in 3.9.7.5
30274: 097: Proc RIRegister_BarCodeScaner_Routines(S: TPSExec);
30275: 996: Proc RIRegister_DBCtrls_Routines(S: TPSExec);
30276: 069: Proc RIRegister_IdStrings_Routines(S: TPSExec);
30277: 516: Proc RIRegister_JclMultimedia_Routines(S: TPSExec);
30278: 215: Proc RIRegister_PNGLoader_Routines(S: TPSExec);
30279: 374: Proc RIRegister_SerDlgs_Routines(S: TPSExec);
30280: 777: Proc RIRegister_LinarBitmap_Routines(S: TPSExec);
30281: 1216 Proc RIRegistger_uPSI_KDialogs, TKBrowseFolderDialog;
30282:
30283: ////////////////////////// TestUnits /////////////////////////
30284: Proc  SelftestPEM;
30285:   SelfTestCFundamentUtils;
30286:   SelfTestCFileUtils;
30287:   SelfTestCDateTime;
30288:   SelfTestCTimer;
30289:   SelfTestCRandom;
30290:   SelftestAES;
30291:   SelfTestASN1;
30292:   SelfTestX509;
30293:   TestDes:Bool;
30294:   Test3Des:Bool;
30295:   TestAes:Bool;
30296:   SelfTestcTLSUtils;
30297:   SelfTestCFundamentUtils;
30298:   SelfTestcHTTPUtils
30299:   SelfTestcXMLFunctions
30300:   SelfTestHugeWord
30301:   SelfTestRSA
30302:
30303:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
30304:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
30305:   //Note: There's no need for installing a client certificate in the
30306:   //       webbrowser. The server asks the webbrowser to send a certificate but
```

```
30307:   //       if nothing is installed the software will work because the server
30308:   //       doesn't check to see if a client certificate was supplied. If you want you can install: file:
      c_cacert.p12  password: c_cakey
30309:
30310:   TKObject = class(TObject)
30311:   private
30312:     FParent: TKObjectList;
30313:     Proc SetParent(const Value: TKObjectList);
30314:   protected
30315:     FUpdateLock: Int;
30316:     Proc CallBeforeUpdate; virtual;
30317:     Proc CallAfterUpdate; virtual;
30318:     Proc ParentChanged; virtual;
30319:   public
30320:     constructor Create; virtual;
30321:     Proc Assign(ASource: TKObject); virtual;
30322:     Func EqualProperties(AValue: TKObject):Bool; virtual;
30323:     Proc LockUpdate; virtual;
30324:     Proc UnLockUpdate; virtual;
30325:     Func UpdateUnlocked:Bool; virtual;
30326:     property Parent: TKObjectList read FParent write SetParent;
30327:   end;
30328:
30329:   TKObjectClass = class of TKObject;
30330:   TKObjectList = class(TObjectList)
30331:   protected
30332:     FUpdateLock: Int;
30333:     Proc CallBeforeUpdate; virtual;
30334:     Proc CallAfterUpdate; virtual;
30335:   public
30336:     constructor Create; virtual;
30337:     Func Add(AObject: TObject): Int;
30338:     Proc Assign(ASource: TKObjectList); virtual;
30339:     Func EqualProperties(AValue: TKObjectList):Bool; virtual;
30340:     Proc Insert(Index: Int; AObject: TObject);
30341:     Proc LockUpdate; virtual;
30342:     Proc UnLockUpdate; virtual;
30343:     Func UpdateUnlocked:Bool; virtual;
30344:   end;
30345:
30346:   TGraphicControl = class(TControl)
30347:   private
30348:     FCanvas: TCanvas;
30349:     Proc WMPaint(var Message: TWMPaint); message WM_PAINT;
30350:   protected
30351:     Proc Paint; virtual;
30352:     property Canvas: TCanvas read FCanvas;
30353:   public
30354:     constructor Create(AOwner: TComponent); override;
30355:     destructor Destroy; override;
30356:   end;
30357:
30358:   TCustomControl = class(TWinControl)
30359:   private
30360:     FCanvas: TCanvas;
30361:     Proc WMPaint(var Message: TWMPaint); message WM_PAINT;
30362:   protected
30363:     Proc Paint; virtual;
30364:     Proc PaintWindow(DC: HDC); override;
30365:     property Canvas: TCanvas read FCanvas;
30366:   public
30367:     constructor Create(AOwner: TComponent); override;
30368:     destructor Destroy; override;
30369:   end;
30370:     RegisterPublishedProperties;
30371:     ('ONCHANGE', 'TNotifyEvent', iptrw);
30372:     ('ONCLICK', 'TNotifyEvent', iptrw);
30373:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
30374:     ('ONENTER', 'TNotifyEvent', iptrw);
30375:     ('ONEXIT', 'TNotifyEvent', iptrw);
30376:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
30377:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
30378:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
30379:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
30380:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
30381: //*********************************************************************
30382:   // To stop the while loop, click on Options/Show Include (boolean switch)!
30383:   Control a loop in a script with a form event:
30384:   IncludeON;   //control the while loop
30385:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
30386:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
30387:   include example:
30388:   {$I .\web\mX47520\flcMaths.inc}
30389:
30390: //-------------------------------------------------------------------------
30391: //**************mX4 ini-file Configuration*************************************
30392: //-------------------------------------------------------------------------
30393:  using config file maxboxdef.ini       menu/Help/Config File
30394:
```

```
30395: //*** Definitions for maXbox mX47 ***
30396: [FORM]
30397: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
30398: FONTSIZE=14
30399: EXTENSION=txt
30400: SCREENX=1386
30401: SCREENY=1077
30402: MEMHEIGHT=350
30403: PRINTFONT=Courier New //GUI Settings
30404: FONTNAME= Courier New //GUI Settings for editor
30405: LINENUMBERS=Y   //line numbers at gutter in editor at left side
30406: EXCEPTIONLOG=Y  //store excepts+success in 2 log files see below! -menu Debug/Show Last Exceptions
30407: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
30408: BOOTSCRIPT=Y    //enabling load a boot script
30409: MEMORYREPORT=Y  //shows memory report on closing maXbox
30410: MACRO=Y         //expand macros (see below) incode e.g. #path:C:\maXbox\works2021\maxbox4\docs\
30411: NAVIGATOR=N     //shows Func list at the right side of editor
30412: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
30413: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
30414: DEP=Y           // activate Data Execution Prevention
30415: INDENT=Y        //shows indent vertical lines! (prepare for code folding) stop it with=N
30416: [WEB]
30417: IPPORT=8080     //for internal webserver - menu /Options/Add Ons/WebServer
30418: IPHOST=192.168.1.53  //run as Administrator!
30419: ROOTCERT=filepathY
30420: SCERT=filepathY
30421: RSAKEY=filepathY
30422: VERSIONCHECK=Y  //set to =N for better starttime and less web traffic!
30423: APP=C:\WINDOWS\System32\calc.exe  //set path to an external app
30424: MYSCRIPT=E:\maxbox4\mXGit39991\examples\330_myclock.txt //start script of menu /View/MyScript
30425:
30426: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
30427: Also possible to set report memory in script to override ini setting
30428: Proc Set_ReportMemoryLeaksOnShutdown(abo:Bool)
30429:
30430: After Change the ini file you can reload the file with ../Help/Config Update
30431: //-----------------------------------------------------------------------------
30432: //**************mX4 maildef.ini ini-file Configuration********************
30433: //-----------------------------------------------------------------------------
30434: //*** Definitions for maXMail ***
30435: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
30436: [MAXMAIL]
30437: HOST=getmail.softwareschule.ch
30438: USER=mailusername
30439: PASS=password
30440: PORT=110
30441: SSL=Y
30442: BODY=Y
30443: LAST=5
30444:
30445: Command Line Interface CLI   -f-s-m-r-t as ParamStr(2); compile with -c
30446: ---------------------------------------------------------
30447: Ex.: S_ShellExecute(ExtractFilePath(ParamStr(0))+'maXbox4.exe',
30448:         myscript + ParamStr(2),seCmdOpen)
30449: Ex.: C:\maXbox\mx4\mx4_v38\mx4\source>maxbox4_7.exe -c examples\440_xml_tutor2.txt
30450:
30451:   if (ParamStr(2) = '-f') then begin
30452:        Application.BringToFront;  //maximize?
30453:        //maxform1.Show;
30454:      end;
30455:      if (ParamStr(2) = '-s') then begin
30456:          maxform1.Show;
30457:      end;
30458:      Compile1Click(self);
30459:      maxform1.memo2.lines.add('CLI Console Call Log at: ' +DateTimeToStr(Now));
30460:      hlog.Add('>>>> Start Console Call Exe: {App_name} v{App_ver}{80@}{now}');
30461:      if (ParamStr(2) = 'm') then begin
30462:        //Compile1Click(self);!
30463:        Application.Minimize;
30464:      end;
30465:     if (ParamStr(2) = 'r') then begin
30466:        Application.run;
30467:      end;
30468:       if (ParamStr(2) = '-t') then begin     //new4
30469:        Application.terminate;
30470:      end;
30471: ---------------------------------------------------------
30472: S_ShellExecute(ExePath+'maXbox4.exe',
30473:  ExePath+'examples\773_streamfibonacci_first.txt t',secmdopen); -terminate
30474: ADO Connection String:
30475: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
30476: \452_dbtreeview2access.txt \452_dbtrv3accessUML2.txt
30477: program TestDbTreeViewMainForm2_ACCESS;
30478:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
30479:          +Exepath+'\examples\detail.mdb;Persist Security Info=False';
30480:  Provider=MSDASQL.1;Persist Security
       Info=False;ExtendedProperties="DSN=FB_EMPLOYEE;Driver=Firebird/InterBase(r)
       driver;Dbname=C:\maXbook\maxbox3\examples\EMPLOYEE.FDB;CHARSET=NONE;UID=SYSDBA;Role=Admin;"'
30481:
```

```
30482: Func CreateAccessDatabase(FileName:Str):Str;
30483: var cat: OLEVariant;
30484: begin
30485:   Result:= '';
30486:   try
30487:     cat:= CreateOleObject('ADOX.Catalog');
30488:     cat.Create('Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + FileName + ';
30489:     cat:= NULL;
30490:   except
30491:     writeln(ExceptionToString(ExceptionType, ExceptionParam));
30492:     writeln('ADOX.Catalog create failed ');
30493:     //on e: Exception do Result := e.message;
30494:   end;
30495: end;
30496:
30497: ADOCommand1.CommandText:= 'CREATE TABLE TABLE1 ('
30498:                            +'ID AUTOINCREMENT,'
30499:                            +'FirstName varchar(255) NOT NULL,'
30500:                            +'LastName varchar(255),'
30501:                            +'Age int )';
30502:
30503: OpenSSL Lib:  unit ssl_openssl_lib;
30504: {$IFDEF CIL}
30505: const
30506:   {$IFDEF LINUX}
30507:   DLLSSLName = 'libssl.so';
30508:   DLLUtilName = 'libcrypto.so';
30509:   {$ELSE}
30510:   DLLSSLName = 'ssleay32.dll';
30511:   DLLUtilName = 'libeay32.dll';
30512:   {$ENDIF}
30513: {$ELSE}
30514: var
30515:   {$IFNDEF MSWINDOWS}
30516:     {$IFDEF DARWIN}
30517:     DLLSSLName:Str = 'libssl.dylib';
30518:     DLLUtilName:Str = 'libcrypto.dylib';
30519:     {$ELSE}
30520:     DLLSSLName:Str = 'libssl.so';
30521:     DLLUtilName:Str = 'libcrypto.so';
30522:     {$ENDIF}
30523:   {$ELSE}
30524:   DLLSSLName:Str = 'ssleay32.dll';
30525:   DLLSSLName2:Str = 'libssl32.dll';
30526:   DLLUtilName:Str = 'libeay32.dll';
30527:   {$ENDIF}
30528: {$ENDIF}
30529:
30530: Proc getHTTP_PNG(vimage: TImage; vURL:Str);
30531: var pngStream: TMemoryStream;
30532: begin
30533:   with TLinearBitmap.Create do
30534:   try
30535:     pngStream:= TMemoryStream.Create;
30536:     try
30537:       HTTPget(vURL, pngStream)
30538:     except
30539:       showmessage(E.message)
30540:     end
30541:     pngStream.Position:= 0;
30542:     LoadFromStream2(pngstream,'JPG');
30543:     vimage.Picture:= NIL;
30544:     AssignTo(vimage.picture);
30545:     //SaveToFile(ExePath+'mX4_open.png');
30546:   finally
30547:     Dispose;
30548:     Free;
30549:     pngStream.Free;
30550:   end;
30551: end;
30552:
30553: Proc ByteArray2Stream(ASource: TBytes; ADest: TStream);
30554: var i: Integer;
30555: begin
30556:    for i:= 0 to Length(ASource)-1 do
30557:       ADest.Write(chr(ASource[i]), 1);
30558: end;
30559:
30560:     buffstr:Str;
30561:     ADest:= TMemoryStream.create;
30562:     ByteArray2Stream([0,23,3,234,8,45,255], aDest);
30563:     SetLength(buffstr,1);
30564:     ADest.Position:= 0;
30565:     for it:= 0 to adest.size-1 do begin
30566:       adest.read(buffstr, 1);
30567:       writeln('decode: '+itoa(ord(buffstr[1])));
30568:     end;
30569:    aDest.free;
30570:
```

```
30571: Proc SearchByteSeq(AHexString:Str; adata: RawByteString);
30572: var dataToSearch: RawByteString;
30573:   nextValueStr:Str;
30574:   nextValueByte: integer; //byte;
30575:   errorPos, index: integer;
30576: begin
30577:   repeat
30578:     AHexString:= Trim(AHexString);
30579:     nextValueStr:= copy(AHexString,1,2);
30580:     delete(AHexString,1,2);
30581:     val('$'+nextValueStr, nextValueByte, errorPos);
30582:     //Func ValBinary( const S :Str; var code : Int) : longint;
30583:     writeln(itoa(errorpos))
30584:     if errorPos = 0 then
30585:       dataToSearch:= datatoSearch + chr(nextValueByte);
30586:   until AHexString = '';
30587:   if length(dataToSearch) > 0 then begin
30588:     index:= pos(dataToSearch, adata)-1;
30589:     if index = 0 then
30590:       writeln('Data not found')
30591:     else
30592:       writeln('Data found at position '+itoa( index));
30593:   end;
30594: end;
30595:
30596: Pictures from:
30597: http://www.softwareschule.ch/images/atomimage1.png
30598: http://www.softwareschule.ch/images/atomimage2.png
30599: http://www.softwareschule.ch/images/maxboxlogo.bmp
30600: http://www.softwareschule.ch/images/maxlindau2.png
30601: http://www.softwareschule.ch/images/citymax.bmp
30602: http://www.softwareschule.ch/images/oposlogo.gif
30603:
30604: //-----------------------------------------------------------------------------
30605: //**************mX4 Macro Tags  ***********************************************
30606: //------------------------------------------------------------ ---------------------
30607: asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
       E:\maxbox\docs\maxbox_extract_funclist399.txt end
30608:
30609: //Tag Macros in ini-file configure
30610:
30611:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
30612:
30613: //Tag Macros
30614: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
30615: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
30616: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
30617: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
30618: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
30619: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
30620: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
30621: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
30622: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
30623:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
30624: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
30625: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
30626:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
30627: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
30628:        [perftime numprocessthreads,getIPAddress(getComputerNameWin),
30629:         timetoStr(time), mbversion]),11);
30630: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
30631:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
30632:
30633: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
30634: //#tech!perf: 0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
30635:
30636: //Replace Macros
30637:   SearchAndCopy(memo1.lines, ' timet14:40:27oStr(time), 6);
30638:   SearchAndCopy(memo1.lines, 'atetoS17/02/2017tr(date), 6);
30639:   SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maXbox3\docs\
30640:   SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maXbox3\
30641:   SearchAndCopy(memo1.lines, 'maxbox_functions.txt
30642:   SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maXbox3\Source
30643:   ref: netcologne.dl.sourceforge.net/project/maxbox/maxbox3.zip
30644:   SearchAndCopy(memo1.lines,'#tech'perf:0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
30645: [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
30646: //#tech!perf: 0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
30647:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
30648:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist476.txt
30649:
30650: //-----------------------------------------------------------------------------
30651: //**************mX4 ToDo List Tags  ../Help/ToDo List*************************
30652: //------------------------------------------------------ -------------------
30653:     while I < sl.Count do begin
30654: //   if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
30655:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
30656:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
30657:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
30658:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
```

```
30659:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
30660:         BreakupToDo(Filename, sl, I, 'TODO', False, True) //only priority info TODO
30661:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
30662:         BreakupToDo(Filename, sl, I, 'DONE', False, True) //only priority info DONE
30663:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
30664:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
30665:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
30666:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
30667:       Inc(I);
30668:     end;
30669:
30670: //-----------------------------------------------------------------------------
30671: //**************mX4 Public  Tools API ******************************************
30672: //-----------------------------------------------------------------------------
30673:  file : unit uPSI_fMain.pas;                {$OTAP} Open Tools API Catalog
30674:  // Those functions concern the editor and preprocessor, all of the IDE
30675:  Example: Call it with maxform1.Info1Click(self)
30676:  Note: Call all Methods with maxForm1., e.g.:
30677:                        maxForm1.ShellStyle1Click(self);
30678:
30679: Proc SIRegister_fMain(CL: TPSPascalCompiler);
30680: begin
30681:  Const('BYTECODE','String 'bytecode.txt'
30682:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
30683:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
30684:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
30685:  Const('PSINC','String PS Includes (*.inc)|*.INC
30686:  Const('DEFFILENAME','String 'firstdemo.txt
30687:  Const('DEFINIFILE','String 'maxboxdef.ini
30688:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
30689:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
30690:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
30691:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
30692:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
30693:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml;
30694:  Const('INCLUDEBOX','String 'pas_includebox.inc
30695:  Const('BOOTSCRIPT','String 'maxbootscript.txt
30696:  Const('MBVERSION','String '4.7.6.20
30697:  Const('VERSION','String'4.7.6.20
30698:  Const('MBVER','String '476
30699:  Const('MBVERI','Int'(476);
30700:  Const('MBVERIALL','Int'(47620);
30701:  Const('EXENAME','String 'maXbox4.exe
30702:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
30703:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
30704:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
30705:  Const('MXINTERNETCHECK','String 'www.ask.com
30706:  Const('MXMAIL','String 'max@kleiner.com
30707:  Const('TAB',Char #$09);
30708:  Const('CODECOMPLETION','String 'bds_delphi.dci
30709:  SIRegister_TMaxForm1(CL);
30710: end;
30711:   with FindClass('TForm'),'TMaxForm1') do begin
30712:     memo2', 'TMemo', iptrw);
30713:     memo1', 'TSynMemo', iptrw);
30714:     CB1SCList', 'TComboBox', iptrw);
30715:     mxNavigator', 'TComboBox', iptrw);
30716:     IPHost', 'string', iptrw);
30717:     IPPort', 'Int', iptrw);
30718:     COMPort', 'Int', iptrw);  //3.9.6.4
30719:     Splitter1', 'TSplitter', iptrw);
30720:     PSScript', 'TPSScript', iptrw);
30721:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
30722:     MainMenu1', 'TMainMenu', iptrw);
30723:     Program1', 'TMenuItem', iptrw);
30724:     Compile1', 'TMenuItem', iptrw);
30725:     Files1', 'TMenuItem', iptrw);
30726:     open1', 'TMenuItem', iptrw);
30727:     Save2', 'TMenuItem', iptrw);
30728:     Options1', 'TMenuItem', iptrw);
30729:     Savebefore1', 'TMenuItem', iptrw);
30730:     Largefont1', 'TMenuItem', iptrw);
30731:     sBytecode1', 'TMenuItem', iptrw);
30732:     Saveas3', 'TMenuItem', iptrw);
30733:     Clear1', 'TMenuItem', iptrw);
30734:     Slinenumbers1', 'TMenuItem', iptrw);
30735:     About1', 'TMenuItem', iptrw);
30736:     Search1', 'TMenuItem', iptrw);
30737:     SynPasSyn1', 'TSynPasSyn', iptrw);
30738:     memo1', 'TSynMemo', iptrw);
30739:     SynEditSearch1', 'TSynEditSearch', iptrw);
30740:     WordWrap1', 'TMenuItem', iptrw);
30741:     XPManifest1', 'TXPManifest', iptrw);
30742:     SearchNext1', 'TMenuItem', iptrw);
30743:     Replace1', 'TMenuItem', iptrw);
30744:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
30745:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
30746:     ShowInclude1', 'TMenuItem', iptrw);
30747:     SynEditPrint1', 'TSynEditPrint', iptrw);
```

```
30748:     Printout1', 'TMenuItem', iptrw);
30749:     mnPrintColors1', 'TMenuItem', iptrw);
30750:     dlgFilePrint', 'TPrintDialog', iptrw);
30751:     dlgPrintFont1', 'TFontDialog', iptrw);
30752:     mnuPrintFont1', 'TMenuItem', iptrw);
30753:     Include1', 'TMenuItem', iptrw);
30754:     CodeCompletionList1', 'TMenuItem', iptrw);
30755:     IncludeList1', 'TMenuItem', iptrw);
30756:     ImageList1', 'TImageList', iptrw);
30757:     ImageList2', 'TImageList', iptrw);
30758:     CoolBar1', 'TCoolBar', iptrw);
30759:     ToolBar1', 'TToolBar', iptrw);
30760:     tbtnLoad', 'TToolButton', iptrw);
30761:     ToolButton2', 'TToolButton', iptrw);
30762:     tbtnFind', 'TToolButton', iptrw);
30763:     tbtnCompile', 'TToolButton', iptrw);
30764:     tbtnTrans', 'TToolButton', iptrw);
30765:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
30766:     toolbtnTutorial', 'TToolButton', iptrw);
30767:     tbtn6res', 'TToolButton', iptrw);
30768:     ToolButton5', 'TToolButton', iptrw); 'ToolButton1', 'TToolButton', iptrw);
30769:     ToolButton3', 'TToolButton', iptrw); 'statusBar1', 'TStatusBar', iptrw);
30770:     SaveOutput1', 'TMenuItem', iptrw); 'ExportClipboard1', 'TMenuItem', iptrw);
30771:     Close1', 'TMenuItem', iptrw); 'Manual1', 'TMenuItem', iptrw);
30772:     About2', 'TMenuItem', iptrw); 'loadLastfile1', 'TMenuItem', iptrw);
30773:     imglogo', 'TImage', iptrw); 'cedebug', 'TPSScriptDebugger', iptrw);
30774:     debugPopupMenu1', 'TPopupMenu', iptrw);
30775:     BreakPointMenu', 'TMenuItem', iptrw);
30776:     Decompile1', 'TMenuItem', iptrw);
30777:     StepInto1', 'TMenuItem', iptrw);
30778:     StepOut1', 'TMenuItem', iptrw);
30779:     Reset1', 'TMenuItem', iptrw);
30780:     DebugRun1', 'TMenuItem', iptrw);
30781:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
30782:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
30783:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
30784:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
30785:     tutorial4', 'TMenuItem', iptrw);
30786:     ExporttoClipboard1', 'TMenuItem', iptrw);
30787:     ImportfromClipboard1', 'TMenuItem', iptrw);
30788:     N4','TMenuItem', iptrw); N5', 'TMenuItem', iptrw);N6', 'TMenuItem', iptrw);
30789:     ImportfromClipboard2', 'TMenuItem', iptrw);
30790:     tutorial1', 'TMenuItem', iptrw);
30791:     N7', 'TMenuItem', iptrw);
30792:     ShowSpecChars1', 'TMenuItem', iptrw);
30793:     OpenDirectory1', 'TMenuItem', iptrw);
30794:     procMess', 'TMenuItem', iptrw);
30795:     tbtnUseCase', 'TToolButton', iptrw);
30796:     ToolButton7', 'TToolButton', iptrw);
30797:     EditFont1', 'TMenuItem', iptrw);
30798:     UseCase1', 'TMenuItem', iptrw);
30799:     tutorial21', 'TMenuItem', iptrw);
30800:     OpenUseCase1', 'TMenuItem', iptrw);
30801:     PSImport_DB1', 'TPSImport_DB', iptrw);
30802:     tutorial31', 'TMenuItem', iptrw);
30803:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
30804:     HTMLSyntax1', 'TMenuItem', iptrw);
30805:     ShowInterfaces1', 'TMenuItem', iptrw);
30806:     Tutorial5', 'TMenuItem', iptrw);
30807:     AllFunctionsList1', 'TMenuItem', iptrw);
30808:     ShowLastException1', 'TMenuItem', iptrw);
30809:     PlayMP31', 'TMenuItem', iptrw);
30810:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
30811:     texSyntax1', 'TMenuItem', iptrw);
30812:     N8', 'TMenuItem', iptrw);
30813:     GetEMails1', 'TMenuItem', iptrw);
30814:     SynCppSyn1', 'TSynCppSyn', iptrw);
30815:     CSyntax1', 'TMenuItem', iptrw);
30816:     Tutorial6', 'TMenuItem', iptrw);
30817:     New1', 'TMenuItem', iptrw);
30818:     AllObjectsList1', 'TMenuItem', iptrw);
30819:     LoadBytecode1', 'TMenuItem', iptrw);
30820:     CipherFile1', 'TMenuItem', iptrw);
30821:     N9', 'TMenuItem', iptrw); 'N10', 'TMenuItem', iptrw);
30822:     Tutorial11', 'TMenuItem', iptrw);
30823:     Tutorial71', 'TMenuItem', iptrw);
30824:     UpdateService1', 'TMenuItem', iptrw);
30825:     PascalSchool1', 'TMenuItem', iptrw);
30826:     Tutorial81', 'TMenuItem', iptrw);
30827:     DelphiSite1', 'TMenuItem', iptrw);
30828:     Output1', 'TMenuItem', iptrw);
30829:     TerminalStyle1', 'TMenuItem', iptrw);
30830:     ReadOnly1', 'TMenuItem', iptrw);
30831:     ShellStyle1', 'TMenuItem', iptrw);
30832:     BigScreen1', 'TMenuItem', iptrw);
30833:     Tutorial91', 'TMenuItem', iptrw);
30834:     SaveOutput2', 'TMenuItem', iptrw);
30835:     N11', 'TMenuItem', iptrw);
30836:     SaveScreenshot', 'TMenuItem', iptrw);
```

```
30837:     Tutorial101', 'TMenuItem', iptrw);
30838:     SQLSyntax1', 'TMenuItem', iptrw);
30839:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
30840:     Console1', 'TMenuItem', iptrw);
30841:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
30842:     XMLSyntax1', 'TMenuItem', iptrw);
30843:     ComponentCount1', 'TMenuItem', iptrw);
30844:     NewInstance1', 'TMenuItem', iptrw);
30845:     toolbtnTutorial', 'TToolButton', iptrw);
30846:     Memory1', 'TMenuItem', iptrw);
30847:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
30848:     JavaSyntax1', 'TMenuItem', iptrw);
30849:     SyntaxCheck1', 'TMenuItem', iptrw);
30850:     Tutorial10Statistics1', 'TMenuItem', iptrw);
30851:     ScriptExplorer1', 'TMenuItem', iptrw);
30852:     FormOutput1', 'TMenuItem', iptrw);
30853:     ArduinoDump1', 'TMenuItem', iptrw);
30854:     AndroidDump1', 'TMenuItem', iptrw);
30855:     GotoEnd1', 'TMenuItem', iptrw);
30856:     AllResourceList1', 'TMenuItem', iptrw);
30857:     ToolButton4', 'TToolButton', iptrw);
30858:     tbtn6res', 'TToolButton', iptrw);
30859:     Tutorial11Forms1', 'TMenuItem', iptrw);
30860:     Tutorial12SQL1', 'TMenuItem', iptrw);
30861:     ResourceExplore1', 'TMenuItem', iptrw);
30862:     Info1', 'TMenuItem', iptrw);
30863:     N12', 'TMenuItem', iptrw);
30864:     CryptoBox1', 'TMenuItem', iptrw);
30865:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
30866:     CipherFile2', 'TMenuItem', iptrw);
30867:     N13', 'TMenuItem', iptrw);
30868:     ModulesCount1', 'TMenuItem', iptrw);
30869:     AddOns2', 'TMenuItem', iptrw);
30870:     N4GewinntGame1', 'TMenuItem', iptrw);
30871:     DocuforAddOns1', 'TMenuItem', iptrw);
30872:     Tutorial14Async1', 'TMenuItem', iptrw);
30873:     Lessons15Review1', 'TMenuItem', iptrw);
30874:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
30875:     PHPSyntax1', 'TMenuItem', iptrw);
30876:     Breakpoint1', 'TMenuItem', iptrw);
30877:     SerialRS2321', 'TMenuItem', iptrw);
30878:     N14', 'TMenuItem', iptrw);
30879:     SynCSSyn1', 'TSynCSSyn', iptrw);
30880:     CSyntax2', 'TMenuItem', iptrw);
30881:     Calculator1', 'TMenuItem', iptrw);
30882:     tbtnSerial', 'TToolButton', iptrw);
30883:     ToolButton8', 'TToolButton', iptrw);
30884:     Tutorial151', 'TMenuItem', iptrw);
30885:     N15', 'TMenuItem', iptrw);
30886:     N16', 'TMenuItem', iptrw);
30887:     ControlBar1', 'TControlBar', iptrw);
30888:     ToolBar2', 'TToolBar', iptrw);
30889:     BtnOpen', 'TToolButton', iptrw);
30890:     BtnSave', 'TToolButton', iptrw);
30891:     BtnPrint', 'TToolButton', iptrw);
30892:     BtnColors', 'TToolButton', iptrw);
30893:     btnClassReport', 'TToolButton', iptrw);
30894:     BtnRotateRight', 'TToolButton', iptrw);
30895:     BtnFullSize', 'TToolButton', iptrw);
30896:     BtnFitToWindowSize', 'TToolButton', iptrw);
30897:     BtnZoomMinus', 'TToolButton', iptrw);
30898:     BtnZoomPlus', 'TToolButton', iptrw);
30899:     Panel1', 'TPanel', iptrw);
30900:     LabelBrettgroesse', 'TLabel', iptrw);
30901:     CB1SCList', 'TComboBox', iptrw);
30902:     ImageListNormal', 'TImageList', iptrw);
30903:     spbtnexplore', 'TSpeedButton', iptrw);
30904:     spbtnexample', 'TSpeedButton', iptrw);
30905:     spbsaveas', 'TSpeedButton', iptrw);
30906:     imglogobox', 'TImage', iptrw);
30907:     EnlargeFont1', 'TMenuItem', iptrw);
30908:     EnlargeFont2', 'TMenuItem', iptrw);
30909:     ShrinkFont1', 'TMenuItem', iptrw);
30910:     ThreadDemo1', 'TMenuItem', iptrw);
30911:     HEXEditor1', 'TMenuItem', iptrw);
30912:     HEXView1', 'TMenuItem', iptrw);
30913:     HEXInspect1', 'TMenuItem', iptrw);
30914:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
30915:     ExporttoHTML1', 'TMenuItem', iptrw);
30916:     ClassCount1', 'TMenuItem', iptrw);
30917:     HTMLOutput1', 'TMenuItem', iptrw);
30918:     HEXEditor2', 'TMenuItem', iptrw);
30919:     Minesweeper1', 'TMenuItem', iptrw);
30920:     N17', 'TMenuItem', iptrw);
30921:     PicturePuzzle1', 'TMenuItem', iptrw);
30922:     sbvclhelp', 'TSpeedButton', iptrw);
30923:     DependencyWalker1', 'TMenuItem', iptrw);
30924:     WebScanner1', 'TMenuItem', iptrw);
30925:     View1', 'TMenuItem', iptrw);
```

```
30926:     mnToolbar1', 'TMenuItem', iptrw);
30927:     mnStatusbar2', 'TMenuItem', iptrw);
30928:     mnConsole2', 'TMenuItem', iptrw);
30929:     mnCoolbar2', 'TMenuItem', iptrw);
30930:     mnSplitter2', 'TMenuItem', iptrw);
30931:     WebServer1', 'TMenuItem', iptrw);
30932:     Tutorial17Server1', 'TMenuItem', iptrw);
30933:     Tutorial18Arduino1', 'TMenuItem', iptrw);
30934:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
30935:     PerlSyntax1', 'TMenuItem', iptrw);
30936:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
30937:     PythonSyntax1', 'TMenuItem', iptrw);
30938:     DMathLibrary1', 'TMenuItem', iptrw);
30939:     IntfNavigator1', 'TMenuItem', iptrw);
30940:     EnlargeFontConsole1', 'TMenuItem', iptrw);
30941:     ShrinkFontConsole1', 'TMenuItem', iptrw);
30942:     SetInterfaceList1', 'TMenuItem', iptrw);
30943:     popintfList', 'TPopupMenu', iptrw);
30944:     intfAdd1', 'TMenuItem', iptrw);
30945:     intfDelete1', 'TMenuItem', iptrw);
30946:     intfRefactor1', 'TMenuItem', iptrw);
30947:     Defactor1', 'TMenuItem', iptrw);
30948:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
30949:     Tutorial20Regex', 'TMenuItem', iptrw);
30950:     N18', 'TMenuItem', iptrw);
30951:     ManualE1', 'TMenuItem', iptrw);
30952:     FullTextFinder1', 'TMenuItem', iptrw);
30953:     Move1', 'TMenuItem', iptrw);
30954:     FractalDemo1', 'TMenuItem', iptrw);
30955:     Tutorial21Android1', 'TMenuItem', iptrw);
30956:     Tutorial0Function1', 'TMenuItem', iptrw);
30957:     SimuLogBox1', 'TMenuItem', iptrw);
30958:     OpenExamples1', 'TMenuItem', iptrw);
30959:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
30960:     JavaScriptSyntax1', 'TMenuItem', iptrw);
30961:     Halt1', 'TMenuItem', iptrw);
30962:     CodeSearch1', 'TMenuItem', iptrw);
30963:     SynRubySyn1', 'TSynRubySyn', iptrw);
30964:     RubySyntax1', 'TMenuItem', iptrw);
30965:     Undo1', 'TMenuItem', iptrw);
30966:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
30967:     LinuxShellScript1', 'TMenuItem', iptrw);
30968:     Rename1', 'TMenuItem', iptrw);
30969:     spdcodesearch', 'TSpeedButton', iptrw);
30970:     Preview1', 'TMenuItem', iptrw);
30971:     Tutorial22Services1', 'TMenuItem', iptrw);
30972:     Tutorial23RealTime1', 'TMenuItem', iptrw);
30973:     Configuration1', 'TMenuItem', iptrw);
30974:     MP3Player1', 'TMenuItem', iptrw);
30975:     DLLSpy1', 'TMenuItem', iptrw);
30976:     SynURIOpener1', 'TSynURIOpener', iptrw);
30977:     SynURISyn1', 'TSynURISyn', iptrw);
30978:     URILinksClicks1', 'TMenuItem', iptrw);
30979:     EditReplace1', 'TMenuItem', iptrw);
30980:     GotoLine1', 'TMenuItem', iptrw);
30981:     ActiveLineColor1', 'TMenuItem', iptrw);
30982:     ConfigFile1', 'TMenuItem', iptrw);
30983:     Sort1Intflist', 'TMenuItem', iptrw);
30984:     Redo1', 'TMenuItem', iptrw);
30985:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
30986:     Tutorial25Configuration1', 'TMenuItem', iptrw);
30987:     IndentSelection1', 'TMenuItem', iptrw);
30988:     UnindentSection1', 'TMenuItem', iptrw);
30989:     SkyStyle1', 'TMenuItem', iptrw);
30990:     N19', 'TMenuItem', iptrw);
30991:     CountWords1', 'TMenuItem', iptrw);
30992:     imbookmarkimages', 'TImageList', iptrw);
30993:     Bookmark11', 'TMenuItem', iptrw);
30994:     N20', 'TMenuItem', iptrw);
30995:     Bookmark21', 'TMenuItem', iptrw);
30996:     Bookmark31', 'TMenuItem', iptrw);
30997:     Bookmark41', 'TMenuItem', iptrw);
30998:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
30999:
31000: Proc IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
31001: Proc IFPS3ClassesPlugin1ExecImport(Sender:TObject;Exec:TPSExec;x:TPSRuntimeClassImporter);
31002:  Proc PSScriptCompile( Sender : TPSScript)
31003:  Proc Compile1Click( Sender : TObject)
31004:  Proc PSScriptExecute( Sender : TPSScript)
31005:  Proc open1Click( Sender : TObject)
31006:  Proc Save2Click( Sender : TObject)
31007:  Proc Savebefore1Click( Sender : TObject)
31008:  Proc Largefont1Click( Sender : TObject)
31009:  Proc FormActivate( Sender : TObject)
31010:  Proc SBytecode1Click( Sender : TObject)
31011:  Proc FormKeyPress( Sender : TObject; var Key : Char)
31012:  Proc Saveas3Click( Sender : TObject)
31013:  Proc Clear1Click( Sender : TObject)
31014:  Proc Slinenumbers1Click( Sender : TObject)
```

```
31015:  Proc About1Click( Sender : TObject)
31016:  Proc Search1Click( Sender : TObject)
31017:  Proc FormCreate( Sender : TObject)
31018:  Proc Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:str;Line,Column:Int;
31019:                                                var Action : TSynReplaceAction)
31020:  Proc Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
31021:  Proc WordWrap1Click( Sender : TObject)
31022:  Proc SearchNext1Click( Sender : TObject)
31023:  Proc Replace1Click( Sender : TObject)
31024:  Func PSScriptNeedFile(Sdr:TObject;const OrginFileName:Str;var FName,Output:Str):Bool;
31025:  Proc ShowInclude1Click( Sender : TObject)
31026:  Proc Printout1Click( Sender : TObject)
31027:  Proc mnuPrintFont1Click( Sender : TObject)
31028:  Proc Include1Click( Sender : TObject)
31029:  Proc FormDestroy( Sender : TObject)
31030:  Proc FormClose( Sender : TObject; var Action : TCloseAction)
31031:  Proc UpdateView1Click( Sender : TObject)
31032:  Proc CodeCompletionList1Click( Sender : TObject)
31033:  Proc SaveOutput1Click( Sender : TObject)
31034:  Proc ExportClipboard1Click( Sender : TObject)
31035:  Proc Close1Click( Sender : TObject)
31036:  Proc Manual1Click( Sender : TObject)
31037:  Proc LoadLastFile1Click( Sender : TObject)
31038:  Proc Memo1Change( Sender : TObject)
31039:  Proc Decompile1Click( Sender : TObject)
31040:  Proc StepInto1Click( Sender : TObject)
31041:  Proc StepOut1Click( Sender : TObject)
31042:  Proc Reset1Click( Sender : TObject)
31043:  Proc cedebugAfterExecute( Sender : TPSScript)
31044:  Proc cedebugBreakpoint(Sender:TObject;const FileName:str;Position,Row,Col:Card)
31045:  Proc cedebugCompile( Sender : TPSScript)
31046:  Proc cedebugExecute( Sender : TPSScript)
31047:  Proc cedebugIdle( Sender : TObject)
31048:  Proc cedebugLineInfo(Sender:TObject;const FileName:str;Position,Row,Col:Card)
31049:  Proc Memo1SpecialLineColors(Sender:TObject;Line:Int;var Speci:Bool;var FG,BG:TColor);
31050:  Proc BreakPointMenuClick( Sender : TObject)
31051:  Proc DebugRun1Click( Sender : TObject)
31052:  Proc tutorial4Click( Sender : TObject)
31053:  Proc ImportfromClipboard1Click( Sender : TObject)
31054:  Proc ImportfromClipboard2Click( Sender : TObject)
31055:  Proc tutorial1Click( Sender : TObject)
31056:  Proc ShowSpecChars1Click( Sender : TObject)
31057:  Proc StatusBar1Dbl1Click( Sender : TObject)
31058:  Proc PSScriptLine( Sender : TObject)
31059:  Proc OpenDirectory1Click( Sender : TObject)
31060:  Proc procMessClick( Sender : TObject)
31061:  Proc tbtnUseCaseClick( Sender : TObject)
31062:  Proc EditFont1Click( Sender : TObject)
31063:  Proc tutorial21Click( Sender : TObject)
31064:  Proc tutorial31Click( Sender : TObject)
31065:  Proc HTMLSyntax1Click( Sender : TObject)
31066:  Proc ShowInterfaces1Click( Sender : TObject)
31067:  Proc Tutorial5Click( Sender : TObject)
31068:  Proc ShowLastException1Click( Sender : TObject)
31069:  Proc PlayMP31Click( Sender : TObject)
31070:  Proc AllFunctionsList1Click( Sender : TObject)
31071:  Proc texSyntax1Click( Sender : TObject)
31072:  Proc GetEMails1Click( Sender : TObject)
31073:  Proc DelphiSite1Click(Sender: TObject);
31074:  Proc TerminalStyle1Click(Sender: TObject);
31075:  Proc ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
31076:  Proc ShellStyle1Click(Sender: TObject);
31077:  Proc Console1Click(Sender: TObject);     //3.2
31078:  Proc BigScreen1Click(Sender: TObject);
31079:  Proc Tutorial91Click(Sender: TObject);
31080:  Proc SaveScreenshotClick(Sender: TObject);
31081:  Proc Tutorial101Click(Sender: TObject);
31082:  Proc SQLSyntax1Click(Sender: TObject);
31083:  Proc XMLSyntax1Click(Sender: TObject);
31084:  Proc ComponentCount1Click(Sender: TObject);
31085:  Proc NewInstance1Click(Sender: TObject);
31086:  Proc CSyntax1Click(Sender: TObject);
31087:  Proc Tutorial6Click(Sender: TObject);
31088:  Proc New1Click(Sender: TObject);
31089:  Proc AllObjectsList1Click(Sender: TObject);
31090:  Proc LoadBytecode1Click(Sender: TObject);
31091:  Proc CipherFile1Click(Sender: TObject);  //V3.5
31092:  Proc NewInstance1Click(Sender: TObject);
31093:  Proc toolbtnTutorialClick(Sender: TObject);
31094:  Proc Memory1Click(Sender: TObject);
31095:  Proc JavaSyntax1Click(Sender: TObject);
31096:  Proc SyntaxCheck1Click(Sender: TObject);
31097:  Proc ScriptExplorer1Click(Sender: TObject);
31098:  Proc FormOutput1Click(Sender: TObject);  //V3.6
31099:  Proc GotoEnd1Click(Sender: TObject);
31100:  Proc AllResourceList1Click(Sender: TObject);
31101:  Proc tbtn6resClick(Sender: TObject);   //V3.7
31102:  Proc Info1Click(Sender: TObject);
31103:  Proc Tutorial10Statistics1Click(Sender: TObject);
```

```
31104:  Proc Tutorial11Forms1Click(Sender: TObject);
31105:  Proc Tutorial12SQL1Click(Sender: TObject);  //V3.8
31106:  Proc ResourceExplore1Click(Sender: TObject);
31107:  Proc Info1Click(Sender: TObject);
31108:  Proc CryptoBox1Click(Sender: TObject);
31109:  Proc ModulesCount1Click(Sender: TObject);
31110:  Proc N4GewinntGame1Click(Sender: TObject);
31111:  Proc PHPSyntax1Click(Sender: TObject);
31112:  Proc SerialRS2321Click(Sender: TObject);
31113:  Proc CSyntax2Click(Sender: TObject);
31114:  Proc Calculator1Click(Sender: TObject);
31115:  Proc Tutorial13Ciphering1Click(Sender: TObject);
31116:  Proc Tutorial14Async1Click(Sender: TObject);
31117:  Proc PHPSyntax1Click(Sender: TObject);
31118:  Proc BtnZoomPlusClick(Sender: TObject);
31119:  Proc BtnZoomMinusClick(Sender: TObject);
31120:  Proc btnClassReportClick(Sender: TObject);
31121:  Proc ThreadDemo1Click(Sender: TObject);
31122:  Proc HEXView1Click(Sender: TObject);
31123:  Proc ExporttoHTML1Click(Sender: TObject);
31124:  Proc Minesweeper1Click(Sender: TObject);
31125:  Proc PicturePuzzle1Click(Sender: TObject); //V3.9
31126:  Proc sbvclhelpClick(Sender: TObject);
31127:  Proc DependencyWalker1Click(Sender: TObject);
31128:  Proc CB1SCListDrawItem(Control:TWinCtrl;Index:Int;aRect:TRect;State:TOwnerDrawState);
31129:  Proc WebScanner1Click(Sender: TObject);
31130:  Proc mnToolbar1Click(Sender: TObject);
31131:  Proc mnStatusbar2Click(Sender: TObject);
31132:  Proc mnConsole2Click(Sender: TObject);
31133:  Proc mnCoolbar2Click(Sender: TObject);
31134:  Proc mnSplitter2Click(Sender: TObject);
31135:  Proc WebServer1Click(Sender: TObject);
31136:  Proc PerlSyntax1Click(Sender: TObject);
31137:  Proc PythonSyntax1Click(Sender: TObject);
31138:  Proc DMathLibrary1Click(Sender: TObject);
31139:  Proc IntfNavigator1Click(Sender: TObject);
31140:  Proc FullTextFinder1Click(Sender: TObject);
31141:  Func AppName:Str;
31142:  Func ScriptName:Str;
31143:  Func LastName:Str;
31144:  Proc FractalDemo1Click(Sender: TObject);
31145:  Proc SimuLogBox1Click(Sender: TObject);
31146:  Proc OpenExamples1Click(Sender: TObject);
31147:  Proc Halt1Click(Sender: TObject);
31148:  Proc Stop;
31149:  Proc CodeSearch1Click(Sender: TObject);
31150:  Proc RubySyntax1Click(Sender: TObject);
31151:  Proc Undo1Click(Sender: TObject);
31152:  Proc LinuxShellScript1Click(Sender: TObject);
31153:  Proc WebScannerDirect(urls:Str);
31154:  Proc WebScanner(urls:Str);
31155:  Proc LoadInterfaceList2;
31156:  Proc DLLSpy1Click(Sender: TObject);
31157:  Proc Memo1DblClick(Sender: TObject);
31158:  Proc URILinksClicks1Click(Sender: TObject);
31159:  Proc GotoLine1Click(Sender: TObject);
31160:  Proc ConfigFile1Click(Sender: TObject);
31161:  Proc Sort1Intflist Click( Sender : TObject)
31162:  Proc Redo1Click( Sender : TObject)
31163:  Proc Tutorial24CleanCode1Click( Sender : TObject)
31164:  Proc IndentSelection1Click( Sender : TObject)
31165:  Proc UnindentSection1Click( Sender : TObject)
31166:  Proc SkyStyle1Click( Sender : TObject)
31167:  Proc CountWords1Click( Sender : TObject)
31168:  Proc Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
31169:  Proc Memo1GutterClick(Send:TObject;Button:TMouseButton;X,Y,Line:Int;Mark:TSynEditMark);
31170:  Proc Bookmark11Click( Sender : TObject)
31171:  Proc Bookmark21Click( Sender : TObject)
31172:  Proc Bookmark31Click( Sender : TObject)
31173:  Proc Bookmark41Click( Sender : TObject)
31174: Proc SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
31175:     'STATMemoryReport', 'boolean', iptrw);
31176:     'IPPort', 'Int', iptrw);
31177:     'COMPort', 'Int', iptrw);
31178:     'lbintflist', 'TListBox', iptrw);
31179:  Func GetStatChange :Bool
31180:  Proc SetStatChange( vstat :Bool)
31181:  Func GetActFileName :Str
31182:  Proc SetActFileName( vname :Str)
31183:  Func GetLastFileName :Str
31184:  Proc SetLastFileName( vname :Str)
31185:  Proc WebScannerDirect( urls :Str)
31186:  Proc LoadInterfaceList2
31187:  Func GetStatExecuteShell :Bool
31188:  Proc DoEditorExecuteCommand( EditorCommand : word)
31189:  Func GetActiveLineColor: TColor
31190:  Proc SetActiveLineColor(acolor: TColor)
31191:  Proc ScriptListbox1Click(Sender: TObject);
31192:  Proc Memo2KeyPress(Sender: TObject; var Key: Char);
```

```
31193:  Proc EnlargeGutter1Click(Sender: TObject);
31194:  Proc Tetris1Click(Sender: TObject);
31195:  Proc ToDoList1Click(Sender: TObject);
31196:  Proc ProcessList1Click(Sender: TObject);
31197:  Proc MetricReport1Click(Sender: TObject);
31198:  Proc ProcessList1Click(Sender: TObject);
31199:  Proc TCPSockets1Click(Sender: TObject);
31200:  Proc ConfigUpdate1Click(Sender: TObject);
31201:  Proc ADOWorkbench1Click(Sender: TObject);
31202:  Proc SocketServer1Click(Sender: TObject);
31203:  Proc FormDemo1Click(Sender: TObject);
31204:  Proc Richedit1Click(Sender: TObject);
31205:  Proc SimpleBrowser1Click(Sender: TObject);
31206:  Proc DOSShell1Click(Sender: TObject);
31207:  Proc SynExport1Click(Sender: TObject);
31208:  Proc ExporttoRTF1Click(Sender: TObject);
31209:  Proc FormCloseQuery(Sender: TObject; var CanClose:Bool);
31210:  Proc SOAPTester1Click(Sender: TObject);
31211:  Proc Sniffer1Click(Sender: TObject);
31212:  Proc AutoDetectSyntax1Click(Sender: TObject);
31213:  Proc FPlot1Click(Sender: TObject);
31214:  Proc PasStyle1Click(Sender: TObject);
31215:  Proc Tutorial183RGBLED1Click(Sender: TObject);
31216:  Proc Reversi1Click(Sender: TObject);
31217:  Proc ManualmaXbox1Click(Sender: TObject);
31218:  Proc BlaisePascalMagazine1Click(Sender: TObject);
31219:  Proc AddToDo1Click(Sender: TObject);
31220:  Proc CreateGUID1Click(Sender: TObject);
31221:  Proc Tutorial27XML1Click(Sender: TObject);
31222:  Proc CreateDLLStub1Click(Sender: TObject);
31223:  Proc Tutorial28DLL1Click(Sender: TObject);;
31224:  Proc ResetKeyPressed;;
31225:  Proc SetKeyPressed;;
31226:  Proc KeyPressedFalse;
31227:  Proc FileChanges1Click(Sender: TObject);;
31228:  Proc OpenGLTry1Click(Sender: TObject);;
31229:  Proc AllUnitList1Click(Sender: TObject);;
31230:  Proc Tutorial29UMLClick(Sender: TObject);
31231:  Proc CreateHeader1Click(Sender: TObject);
31232:  Proc Oscilloscope1Click(Sender: TObject);;
31233:  Proc Tutorial30WOT1Click(Sender: TObject);;
31234:  Proc GetWebScript1Click(Sender: TObject);;
31235:  Proc Checkers1Click(Sender: TObject);;
31236:  Proc TaskMgr1Click(Sender: TObject);;
31237:  Proc WebCam1Click(Sender: TObject);;
31238:  Proc Tutorial31Closure1Click(Sender: TObject);;
31239:  Proc GEOMapView1Click(Sender: TObject);;
31240:  Proc Run1Click(Sender: TObject);
31241:  MaxForm1.GPSSatView1Click, 'GPSSatView1Click;
31242:  MaxForm1.N3DLab1Click, 'N3DLab1Click;
31243:  Proc ExternalApp1Click(Sender: TObject);;
31244:  Proc PANView1Click(Sender: TObject);
31245:  Proc Tutorial39GEOMaps1Click(Sender: TObject);
31246:  Proc UnitConverter1Click(Sender: TObject);
31247:  maxform1.myscript1click(self)
31248:  Proc Terminal1Click(Sender: TObject);
31249:  Proc Tutorial361Click(Sender: TObject);
31250:  Proc TrainingArduino1Click(Sender: TObject);
31251:  Proc Chess41Click(Sender: TObject);
31252:  Proc OrangeStyle1Click(Sender: TObject);
31253:  Proc Darkcolor1Click(Sender: TObject);
31254: //-----------------------------------------------------------------------------
31255: //**************mX4 Editor SynEdit Tools API **********************************
31256: //-----------------------------------------------------------------------------
31257: Proc SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
31258: begin  //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
31259:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
31260:     Constructor Create( AOwner : TComponent)
31261:     SelStart', 'Int', iptrw);
31262:     SelEnd', 'Int', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
31263:     Proc UpdateCaret
31264: Proc AddKey(Command:TSynEditorCommand;Key1:word;SS1:TShiftState;Key2:word;SS2:TShiftState);
31265: Proc AddKey(Command:TSynEditorCommand;Key1:word;SS1:TShiftState;Key2:word;SS2:TShiftState);
31266:   Proc BeginUndoBlock
31267:   Proc BeginUpdate
31268:   Func CaretInView :Bool
31269:   Func CharIndexToRowCol( Index : Int) : TBufferCoord
31270:   Proc Clear
31271:   Proc ClearAll
31272:   Proc ClearBookMark( BookMark : Int)
31273:   Proc ClearSelection
31274:   Proc CommandProcessor(Command:TSynEditorCommand; AChar : char; Data : pointer)
31275:   Proc ClearUndo
31276:   Proc CopyToClipboard
31277:   Proc CutToClipboard
31278:   Proc DoCopyToClipboard( const SText :Str)
31279:   Proc EndUndoBlock
31280:   Proc EndUpdate
31281:   Proc EnsureCursorPosVisible
```

```
31282:   Proc EnsureCursorPosVisibleEx( ForceToMiddle :Bool)
31283:   Proc FindMatchingBracket
31284:   Func GetMatchingBracket : TBufferCoord
31285:   Func GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
31286:   Proc ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
31287:   Func GetBookMark( BookMark : Int; var X, Y : Int) :Bool
31288:   Func GetHighlighterAttriAtRowCol(const XY: TBufferCoord; var Token:str; var Attri
31289:            : TSynHighlighterAttributes) :Bool
31290:   Func GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token :Str;
31291:            var TokenType, Start : Int; var Attri:TSynHighlighterAttributes):boolean
31292:   Func GetPositionOfMouse( out aPos : TBufferCoord) :Bool
31293:   Func GetWordAtRowCol( const XY : TBufferCoord) :Str
31294:   Proc GotoBookMark( BookMark : Int)
31295:   Proc GotoLineAndCenter( ALine : Int)
31296:   Func IdentChars : TSynIdentChars
31297:   Proc InvalidateGutter
31298:   Proc InvalidateGutterLine( aLine : Int)
31299:   Proc InvalidateGutterLines( FirstLine, LastLine : Int)
31300:   Proc InvalidateLine( Line : Int)
31301:   Proc InvalidateLines( FirstLine, LastLine : Int)
31302:   Proc InvalidateSelection
31303:   Func IsBookmark( BookMark : Int) :Bool
31304:   Func IsPointInSelection( const Value : TBufferCoord) :Bool
31305:   Proc LockUndo
31306:   Func BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
31307:   Func DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
31308:   Func LineToRow( aLine : Int) : Int
31309:   Func RowToLine( aRow : Int) : Int
31310:   Func NextWordPos : TBufferCoord
31311:   Func NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
31312:   Proc PasteFromClipboard
31313:   Func WordStart : TBufferCoord
31314:   Func WordStartEx( const XY : TBufferCoord) : TBufferCoord
31315:   Func WordEnd : TBufferCoord
31316:   Func WordEndEx( const XY : TBufferCoord) : TBufferCoord
31317:   Func PrevWordPos : TBufferCoord
31318:   Func PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
31319:   Func PixelsToRowColumn( aX, aY : Int) : TDisplayCoord
31320:   Func PixelsToNearestRowColumn( aX, aY : Int) : TDisplayCoord
31321:   Proc Redo
31322:   Proc RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerDat:pointr);
31323:   Func RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
31324:   Func RowColToCharIndex( RowCol : TBufferCoord) : Int
31325:   Func SearchReplace( const ASearch,AReplace:str; AOptions:TSynSearchOptions): int
31326:   Proc SelectAll
31327:   Proc SetBookMark( BookMark : Int; X : Int; Y : Int)
31328:   Proc SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
31329:   Proc SetDefaultKeystrokes
31330:   Proc SetSelWord
31331:   Proc SetWordBlock( Value : TBufferCoord)
31332:   Proc Undo Proc UnlockUndo
31333:   Proc UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
31334:   Proc AddKeyUpHandler( aHandler : TKeyEvent)
31335:   Proc RemoveKeyUpHandler( aHandler : TKeyEvent)
31336:   Proc AddKeyDownHandler( aHandler : TKeyEvent)
31337:   Proc RemoveKeyDownHandler( aHandler : TKeyEvent)
31338:   Proc AddKeyPressHandler( aHandler : TKeyPressEvent)
31339:   Proc RemoveKeyPressHandler( aHandler : TKeyPressEvent)
31340:   Proc AddFocusControl( aControl : TWinControl)
31341:   Proc RemoveFocusControl( aControl : TWinControl)
31342:   Proc AddMouseDownHandler( aHandler : TMouseEvent)
31343:   Proc RemoveMouseDownHandler( aHandler : TMouseEvent)
31344:   Proc AddMouseUpHandler( aHandler : TMouseEvent)
31345:   Proc RemoveMouseUpHandler( aHandler : TMouseEvent)
31346:   Proc AddMouseCursorHandler( aHandler : TMouseCursorEvent)
31347:   Proc RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
31348:   Proc SetLinesPointer( ASynEdit : TCustomSynEdit)
31349:   Proc RemoveLinesPointer
31350:   Proc HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
31351:   Proc UnHookTextBuffer
31352:   BlockBegin', 'TBufferCoord', iptrw);
31353:   BlockEnd', 'TBufferCoord', iptrw);
31354:   CanPaste', 'Boolean', iptr); 'CanRedo', 'boolean', iptr);
31355:   CanUndo', 'boolean', iptr);  'CaretX', 'Int', iptrw);
31356:   CaretY', 'Int', iptrw); 'CaretXY', 'TBufferCoord', iptrw);
31357:   ActiveLineColor', 'TColor', iptrw);
31358:   DisplayX', 'Int', iptr); 'DisplayY', 'Int', iptr);
31359:   DisplayXY', 'TDisplayCoord', iptr); 'DisplayLineCount', 'Int', iptr);
31360:   CharsInWindow', 'Int', iptr);
31361:   CharWidth', 'Int', iptr);
31362:   Font', 'TFont', iptrw);
31363:   GutterWidth', 'Int', iptr);
31364:   Highlighter', 'TSynCustomHighlighter', iptrw);
31365:   LeftChar', 'Int', iptrw);
31366:   LineHeight', 'Int', iptr);
31367:   LinesInWindow', 'Int', iptr);
31368:   LineText', 'string', iptrw); Lines', 'TStrings', iptrw);
31369:   Marks', 'TSynEditMarkList', iptr);
31370:   MaxScrollWidth', 'Int', iptrw);
```

```
31371:   Modified', 'Boolean', iptrw);
31372:   PaintLock', 'Int', iptr);
31373:   ReadOnly', 'Boolean', iptrw);
31374:   SearchEngine', 'TSynEditSearchCustom', iptrw);
31375:   SelAvail', 'Boolean', iptr); SelLength', 'Int', iptrw);
31376:   SelTabBlock', 'Boolean', iptr);
31377:   SelTabLine', 'Boolean', iptr); SelText', 'string', iptrw);
31378:   StateFlags', 'TSynStateFlags', iptr);
31379:   Text', 'string', iptrw); TopLine', 'Int', iptrw);
31380:   WordAtCursor', 'string', iptr); 'WordAtMouse', 'string', iptr);
31381:   UndoList', 'TSynEditUndoList', iptr);
31382:   RedoList', 'TSynEditUndoList', iptr);
31383:   OnProcessCommand', 'TProcessCommandEvent', iptrw);
31384:   BookMarkOptions', 'TSynBookMarkOpt', iptrw);
31385:   BorderStyle', 'TSynBorderStyle', iptrw);
31386:   ExtraLineSpacing', 'Int', iptrw);
31387:   Gutter', 'TSynGutter', iptrw);
31388:   HideSelection', 'boolean', iptrw);
31389:   InsertCaret', 'TSynEditCaretType', iptrw);
31390:   InsertMode', 'boolean', iptrw); 'IsScrolling', 'Boolean', iptr);
31391:   Keystrokes', 'TSynEditKeyStrokes', iptrw);
31392:   MaxUndo', 'Int', iptrw); 'Options', 'TSynEditorOptions', iptrw);
31393:   OverwriteCaret', 'TSynEditCaretType', iptrw);
31394:   RightEdge', 'Int', iptrw); RightEdgeColor', 'TColor', iptrw);
31395:   ScrollHintColor', 'TColor', iptrw);
31396:   ScrollHintFormat', 'TScrollHintFormat', iptrw);
31397:   ScrollBars', 'TScrollStyle', iptrw);
31398:   SelectedColor', 'TSynSelectedColor', iptrw);
31399:   SelectionMode', 'TSynSelectionMode', iptrw);
31400:   ActiveSelectionMode', 'TSynSelectionMode', iptrw);
31401:   TabWidth', 'Int', iptrw);  WantReturns', 'boolean', iptrw);
31402:   WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
31403:   WordWrapGlyph', 'TSynGlyph', iptrw);
31404:   OnChange', 'TNotifyEvent', iptrw);
31405:   OnClearBookmark', 'TPlaceMarkEvent', iptrw);
31406:   OnCommandProcessed', 'TProcessCommandEvent', iptrw);
31407:   OnContextHelp', 'TContextHelpEvent', iptrw);
31408:   OnDropFiles', 'TDropFilesEvent', iptrw);
31409:   OnGutterClick', 'TGutterClickEvent', iptrw);
31410:   OnGutterGetText', 'TGutterGetTextEvent', iptrw);
31411:   OnGutterPaint', 'TGutterPaintEvent', iptrw);
31412:   OnMouseCursor', 'TMouseCursorEvent', iptrw);
31413:   OnPaint', 'TPaintEvent', iptrw);
31414:   OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
31415:   OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
31416:   OnReplaceText', 'TReplaceTextEvent', iptrw);
31417:   OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
31418:   OnStatusChange', 'TStatusChangeEvent', iptrw);
31419:   OnPaintTransient', 'TPaintTransient', iptrw);
31420:   OnScroll', 'TScrollEvent', iptrw);
31421:  end;
31422:  Proc RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
31423:  Func GetPlaceableHighlighters : TSynHighlighterList
31424:  Func EditorCommandToDescrString( Cmd : TSynEditorCommand) :Str
31425:  Func EditorCommandToCodeString( Cmd : TSynEditorCommand) :Str
31426:  Proc GetEditorCommandValues( Proc : TGetStrProc)
31427:  Proc GetEditorCommandExtended( Proc : TGetStrProc)
31428:  Func IdentToEditorCommand( const Ident :Str; var Cmd : longint) :Bool
31429:  Func EditorCommandToIdent( Cmd : longint; var Ident :Str) :Bool
31430:  Func ConvertCodeStringToExtended( AString :Str) :Str
31431:  Func ConvertExtendedToCodeString( AString :Str) :Str
31432:  Func ConvertExtendedToCommand( AString :Str) : TSynEditorCommand
31433:  Func ConvertCodeStringToCommand( AString :Str) : TSynEditorCommand
31434:  Func IndexToEditorCommand( const AIndex : Int) : Int
31435: TSynEditorOption = (
31436:  eoAltSetsColumnMode,        //Holding down Alt Key will put select mode into columnar format
31437:  eoAutoIndent,         //Will indent caret onnewlines same amountof leading whitespace as
31438:                              // preceding line
31439:  eoAutoSizeMaxScrollWidth,  //Automatically resizes MaxScrollWidth property when insert text
31440:  eoDisableScrollArrows,     //Disables scroll bar arrow buttons when you cant scroll that
31441:                              //direction any more
31442:  eoDragDropEditing,  //Allows to select a textblock and drag in doc to another location
31443:  eoDropFiles,               //Allows the editor accept OLE file drops
31444:  eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
31445:  eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
31446:  eoGroupUndo, //When undoing/redoing actions,handle all cont.changes same kind in onecall
31447:                              //instead undoing/redoing each command separately
31448:  eoHalfPageScroll,   //By scrolling with page-up/page-down commands, scroll half page attime
31449:  eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
31450:  If you have ScrollPastEOL,then it horizontal bar will always there (uses MaxLength instead)
31451:  eoKeepCaretX,          //When moving through lines w/o cursor Past EOL, keeps X posof cursor
31452:  eoNoCaret,                 //Makes it so the caret is never visible
31453:  eoNoSelection,             //Disables selecting text
31454:  eoRightMouseMovesCursor,//When clicking right mouse for popup menu,moves cursor to location
31455:  eoScrollByOneLess,         //Forces scrolling to be one less
31456:  eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
31457:  eoScrollPastEof,           //Allows the cursor to go past the end of file marker
31458:  eoScrollPastEol,           //Allows cursor go past lastchar into white space at end ofline
31459:  eoShowScrollHint,  //Shows a hint of the visible line numbers when scrolling vertically
```

```
31460:  eoShowSpecialChars,         //Shows the special Characters
31461:  eoSmartTabDelete,           //similar to Smart Tabs, but when you delete characters
31462:  eoSmartTabs,                //When tabbing, cursor will goto non-whitespace charof prev line
31463:  eoSpecialLineDefaultFg,//disables foreground textcolor override  OnSpecialLineColor event
31464:  eoTabIndent,      //If active <Tab>and<Shift><Tab> act block indent,unindent iftext select
31465:  eoTabsToSpaces, //Converts a tab character to a specified number of space characters
31466:  eoTrimTrailingSpaces        //Spaces at the end of lines will be trimmed and not saved
31467:
31468:  Change above Options in Editor (for bootscript also):
31469:  for eg. to indent blocks with tab:
31470:
31471:  memo1.options:= memo1.options + [eoTabIndent]
31472:  or to control something:
31473:  memo1.wanttabs:= true;
31474:  if (memo1.wanttabs) and (GetKeyState(VK_CONTROL)) then writeln('wanttabs0^');
31475:
31476: *******************Important Editor Short Cut Cuts shorcut*****************);
31477:  Double click to select a word and count words with highlightning.
31478:  Triple click to select a line.
31479:  CTRL+SHIFT+click to extend a selection.
31480:  Drag with the ALT key down to select columns of text !!!
31481:  Drag and drop is supported.
31482:  With CTRL 3 to jump to the last change  (change tracker)
31483:  Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
31484:  Type CTRL+A to select all.
31485:  Type CTRL+N to set a new line.
31486:  Type CTRL+T to delete a line or token. //Tokenizer
31487:  Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
31488:  Type CTRL+Shift+S to Save as....
31489:  Type CTRL+Alt+M to jump to Memo Terminal.
31490:  Type CTRL+Shift+T to add ToDo in line and list.
31491:  Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
31492:  Type CTRL[0..9] to jump or get to bookmarks.
31493:  Type Home to position cursor at beginning of curr line and End to position at end of line.
31494:  Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of docu.
31495:     Page Up and Page Down work as expected.
31496:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
31497:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
31498:
31499: {$ Short Key Positions Ctrl<A-Z>: }
31500: def
31501:    <A> Select All
31502:    <B> Count Words
31503:    <C> Copy
31504:    <D> Internet Start
31505:    <E> Script List
31506:    <F> Find
31507:    <G> Goto
31508:    <H> Mark Line
31509:    <I> Interface List
31510:    <J> Code Completion
31511:    <K> Console
31512:    <L> Interface List Box
31513:    <M> Font Smaller -
31514:    <N> New Line
31515:    <O> Open File
31516:    <P> Font Larger +
31517:    <Q> Quit
31518:    <R> Replace
31519:    <S> Save!
31520:    <T> Delete Line
31521:    <U> Use Case Editor
31522:    <V> Paste
31523:    <W> URI Links
31524:    <X> Reserved for coding use internal
31525:    <Y> Delete Line
31526:    <Z> Undo
31527:
31528: ref F1 Help to Func List
31529:     F2 Syntax Check
31530:     F3 Search Next
31531:     F4 New Instance
31532:     F5 Line Mark /Breakpoint
31533:     F6 Goto End
31534:     F7 Debug Step Into
31535:     F8 Debug Step Out
31536:     F9 Compile
31537:     F10 Menu
31538:     F11 Word Count Highlight
31539:     F12 Reserved for coding use internal
31540:
31541:    AddRegisteredVariable('Application', 'TApplication;
31542:    AddRegisteredVariable('Screen', 'TScreen;
31543:    AddRegisteredVariable('Self', 'TForm;
31544:    AddRegisteredVariable('Memo1', 'TSynMemo;
31545:    AddRegisteredVariable('memo2', 'TMemo;
31546:    AddRegisteredVariable('maxForm1', 'TMaxform1;  //!!
31547:    AddRegisteredVariable('debugout', 'Tdebugoutput;  //!!
31548:    AddRegisteredVariable('hlog','THotlog;  //!!
```

```
31549:    AddRegisteredVariable('mouse','TMouse;  //!!
31550:    AddRegisteredVariable( it ,Int;  //for closure and each loop!!
31551:    AddRegisteredVariable( sr ,string;  //for closure
31552:    AddRegisteredVariable( bt ,boolean;  //for closure
31553:    AddRegisteredVariable( ft ,double;  //for closure
31554:    AddRegisteredVariable( srlist ,TStringlist;  //for closures
31555:
31556:  def ReservedWords: array[0..86] of string =
31557:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
31558:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
31559:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
31560:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
31561:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
31562:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
31563:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
31564:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
31565:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
31566:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected', xraise
31567:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
31568:    AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3:Bool;
31569: //-----------------------------------------------------------------------------
31570: //**************End of mX4 Public  Tools API ***********************************
31571: //-----------------------------------------------------------------------------
31572:
31573: maXbox 4 Internal Now Inventory INI Nov 2017
31574: Amount of Functions: 18547
31575: Amount of Procedures: 11143
31576: Amount of Constructors: 1730
31577: Totals of Calls: 31420
31578: SHA1: of 4.5.8.10 FFF8300142CE582DFCC8E15489AD7D5EE2E08062
31579: CRC32: A013C5BC:  28465616 bytes
31580: ▯▯▯ mX4 executed: 03/01/2018 17:10:09  Runtime: 0:8:5.816  Memload: 29% use
31581: maXbox 4 Internal Now Inventory INI Dez 2019
31582: Amount of Functions: 18929
31583: Amount of Procedures: 11351
31584: Amount of Constructors: 1771
31585: Amount of Destructors: 12
31586: Totals of Calls: 32063
31587: SHA1: of 4.7.1.80 E372D00CE31BDFC30A29F6C1F77600ABD25AFDB8
31588: CRC32: 3C620310:  28952992 bytes
31589: ▯▯▯ mX4 executed: 03/12/2019 23:15:23  Runtime: 0:9:29.13  Memload: 48% use
31590: ▯▯▯ V 4.7.4.64
31591: Totals of Func Calls: 32633, 75 Tutorials
31592: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
31593: CRC32: 3EB27A87: 28.2 MB (29,608,248) bytes
31594:
31595: ************************************************************
31596:  Doc Short Manual with 70 Tips! Troublshooter - Errors
31597: ************************************************************
31598: - Editor: dont ever load and save a BINARY file in the editor, it saves as CRLF text !
31599: - Install: just save your maxboxdef.ini before and then extract the zip file!
31600: - Toolbar: Click on the red maXbox Sign (right on top) opens your work dir or jump to <Help>
31601: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
31602: - Menu: With <Crtl><F3> you can search for code on examples
31603: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
31604: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
31605: - Menu: Switch or toogle between last 2 scripts in menu File/LoadLast(History is set to 9 files)
31606:
31607: - Inifile: Set memory report in ini:MEMORYREPORT=Y :report on memoryleaks on shutdown by dialog
31608: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
31609: - Inifile: Set VERSIONCHECK=N and MEMORYREPORT=N to speedup boot and shutdown of box
31610:
31611: - Context Menu: You can printout your scripts as a pdf-file or html-export
31612: - Context: You do have a context menu with the right mouse click
31613:
31614: - Menu: With the UseCase Editor you can convert graphic formats too.
31615: - Menu: On menu Options you find Addons as compiled scripts
31616: - IDE: Menu Program: Run Only is faster, after F2 - You dont need a mouse use shortcuts
31617: - Menu: Check Options/ProcessMessages! if something is wrong or you cant see graphics in a time
31618: - IDE: Dragndrop scripts in box or model in use case editor(Cut,Copy,Paste always available)
31619: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface or
       ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
31620:
31621: - Menu: In menu output (console) set output menu in edit mode by unchecking <read only output>
31622: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
31623: - Code: If you code a loop till key-pressed use function: isKeyPressed;
31624: - Code: Macro set the macros #name,dmMaxMaxinistratorth, #file,s.txtmaxbox_functions.txtist5.pdf
31625: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
31626: - Editor: -<F11> Click on Word in Editor search amount of words with highlighting,Dbl Click on Bookmarks
       to delete and Click and mark to drag a bookmark
31627: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
31628: - IDE: A file info with system and script information you find in menu Program/Information
31629: - IDE: After change the config file in help you can update changes in menu Help/Config Update
31630: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
31631: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time start it.
31632: - IDE: With escape or <Ctrl> Q you can also leave box or stopscript in menu program - stop program
31633: - Editor: Set Bookmarks to check your work in app or code
31634: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
31635: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
```

```
31636: - Editor: With <Ctrl W> you set active URL links in code to test availability in Context Menu
31637: - Editor  shortcut Ctrl+Alt+M to jump to output (memo2) console
31638: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
31639: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
31640: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
31641: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
31642: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
31643: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
31644: - Code: if you cant run a Func try the second one, for ex. Voice()-Voice2(),inc() - inc1()
31645: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
31646: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
31647: - IDE menu /Help/Tools/ open the Task Manager
31648: - Editor: set doc section after end. end. in units and interface with begin main end!
31649: - new Style: menu//Output/Darkcolor
31650:   http://www.softwareschule.ch/images/maXbox4_darkcolor.png
31651:   http://www.softwareschule.ch/images/maXbox4_darkcolor3.png
31652:
31653: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
31654: - Add on when no browser is available start /Options/Add ons/Easy Browser
31655: - Add on SOAP Tester with SOP POST File
31656: - Add on IP Protocol Sniffer with List View
31657: - Add on OpenGL mX Robot Demo for android
31658: - Add on Checkers Game, Add on Oscilloscope /View GEO Map View3
31659:
31660: - Menu: Help/Tools as a Tool Section with DOS Opener
31661: - Menu Editor: export the code as RTF File
31662: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
31663: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
31664: - Context: Auto Detect of Syntax depending on file extension
31665: - Code: some Windows API Func start with w in the name like wGetAtomName();
31666: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
31667: - IDE File Check with menu ..View/File Changes/...
31668: - Context: Create a Header with Create Header in Navigator List at right window
31669: - Code: use  SysErrorMessage to get a real Error Description, Ex.
31670:   RemoveDir('c:\NoSuchFold;writeln('System Error Message:'+SysErrorMessage(GetLastError));
31671: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
31672: - Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click on link
31673: - Editor: with <Ctrl+Alt+R> you can write in RTF format with RichEdit link
31674: - Editor: with <Ctrl+Alt+T> you can fill your To Do List.
31675: - Editor: with <Ctrl+Alt+M> you jump to output window below (memo2).
31676: - Menu: Check Help/Tools! you can use richedit, DOS Shell or Explorer
31677: - Menu: Start View/MyScript of ini file maxboxdef.ini [MYSCRIPT]= path of script
31678: - Indent: close you doc lines with //}    for {...}
31679: - Menu: Help/ToDo List add prio and user with {/1.......//}
31680:             Name#Prio# example:
31681:   {/1 TODO (maxname#7#): do the new task one34 then mX4' //}
31682:
31683: - With this code I get an EIdHTTPProtocolException with the message "HTTP/1.1 403 Forbidden".
31684: Set the IdHTTP1.Request.UserAgent to some browser's user agent and you'll get it to work. Server doesn't
       like the Indy's default one and rejects its request.
31685:
31686:    with TIdHTTP.Create(self)  do begin
31687:       Request.UserAgent:= 'maXbox4 compatible';
31688:       get('http://www.softwareschule.ch/examples/robo.txt')
31689:       free
31690:    end;
31691:
31692: use for TRegistry.create constructor TRegistry.Create1
31693: in maxbox_functions.txt add TRegistry.Create1
31694:
31695:  //In the ServiceAfterUninstall method add these lines:
31696:  // begin
31697:   // Delete registry entries for event viewer.
31698:   Key:= '\SYSTEM\CurrentControlSet\Services\Eventlog\Application\' + Self.Name;
31699:   Reg:= TRegistry.Create1(KEY_READ or KEY_WRITE);
31700:   try
31701:     Reg.RootKey := HKEY_LOCAL_MACHINE;
31702:     if Reg.KeyExists(Key) then
31703:       Reg.DeleteKey(Key);
31704:   finally
31705:     Reg.Free;
31706:   end;
31707:
31708: Func AnsiByteArrayStringLen2(Data: TBytes): Integer;
31709: Func StringToAnsiByteArray2(const S:Str): TBytes;
31710: Func AnsiByteArrayToString2(const Data: TBytes; Count: Integer):Str;
31711: Func BytesOf(const Value: Ansistr): TBytes;
31712: Func BytesOfChar(const Value: AnsiChar): TBytes;
31713: Func StringOf(const Bytes: array of Byte): Ansistr;
31714:
31715: - using DLL example in maXbox: //function: {**********************}
31716:  Func GetProcessMemoryInfo(Process:THandle;var
       MemoryCounters:TProcessMemoryCounters;cb:DWORD):BOOL;//stdcall;
31717:     External 'GetProcessMemoryInfo@psapi.dll stdcall';
31718:  Func OpenProcess(dwDesiredAccess:DWORD;bInheritHandle:BOOL;dwProcessId: DWORD):THandle;
31719:     External  'OpenProcess@kernel32.dll stdcall';
31720:
31721: Win API direct DLL with array of char result:
31722:
```

```
31723: //https://docs.microsoft.com/en-us/windows/win32/api/shlobj_core/nf-shlobj_core-shgetfolderpatha
31724:
31725:   function  SHGetFolderPathA(hwnd: HWND; csidl: integer; htoken: THANDLE;
31726:                               dwflags: DWORD; pszPath: string): HRESULT;
31727:    external 'SHGetFolderPathA@Shell32.dll stdcall';
31728:
31729: function GetMyPictures: string;
31730: var pStr: string; //array[0..260] of Char;
31731: begin
31732:   setLength(pstr, 260-1);
31733:   if SHGetFolderPathA(0, CSIDL_MYPICTURES, 0,0, pStr) = S_OK then
31734:     Result:= pStr;
31735: end;
31736:
31737: >>> GetMyPictures path C:\Users\Max\OneDrive\Pictures
31738:
31739: GCC Compile Ex Script
31740: Proc TFormMain_btnCompileClick(Sender: TObject);
31741: begin
31742:   AProcess:= TProcess.Create(Nil);
31743:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
31744:       + ' -o "' + OpenDialog2.FileName + '"';
31745:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
31746:     AProcess.Execute;
31747:     Memo2.Lines.BeginUpdate;
31748:     Memo2.Lines.Clear;
31749:     Memo2.Lines.LoadFromStream(AProcess.Output);
31750:     Memo2.Lines.EndUpdate;
31751:   finally
31752:     AProcess.Free;
31753:   end;
31754: end;
31755:
31756: ref Stopwatch pattern snip
31757: Time1:= Time;
31758:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
31759:   if initAndStartBoard then
31760:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
31761:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
31762:   PrintF('%d %s',[Trunc((Time-Time1)*24),FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
31763:   POST git-receive-pack (chunked)
31764: Pushing to https://github.com/maxkleiner/maXbox3.git
31765: To https://github.com/maxkleiner/maXbox3.git f127d21..c6a98da  masterbox2 -> masterbox2
31766: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
31767:
31768: ---------------------------------------------------------------------------
31769: History Shell Hell - Walk the Talk - CoolCode
31770: PCT Precompile Technology , mX4 ScriptStudio
31771: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
31772: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
31773: emax layers: system-package-component-unit-class-function-block
31774: new keywords def ref using maXCalcF UML xraise: use case act class state seq pac comp dep - lib lab
31775: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
31776: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
31777: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
31778: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
31779: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
31780: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
31781: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
31782: SendBuffer, Color+Caption hack,ComboSet,SetPrivilege,WakeOnLAN,ParaDice 3DCube Polygraph,OCR
31783: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
31784: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
31785: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
31786: Inno Install and Setup Routines Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
31787: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
31788: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
31789: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
31790: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
31791: Func Menu/View/GEO Map View, DownloadFile, wgetX, sensors
31792: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
31793: ByteCode2, IPUtils2, GEOCode, CGI-Powtils, GPS_2, External App, Unit Converter
31794: Add 16 Units, 1 Slide,Tutor, Big Numbers (Decimals, TInt), ModBusTCP, TGEOInfo
31795: Add 36 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, ACM, CDS, XMLDoc, DDE
31796: Add 77 Units, 12 Tutors, ChangeTracker, CPP+,OLEUtils2,xmldom,Chess4,3DFrame,XMLRPC,X509
31797: Add 15 Units, 1 Tutor, Pipe Libraray2, KLog, FPlot42, KDialogs, NumEdit, KControls, Kronos
31798: add 12 units and 125 functions - Class Helper - KMemo RTF
31799: add 16 units and 365 functions- WMI Script Type Library - webbox -restbox
31800: V4.6.2.10 Jan 2018:Tutor 56 Neural Network-Python Checker- 3 more Units PascalCoin
31801: V4.7.1.10 Sep 2019:Tutor 57-70, VState Machine,CGI,MachineLearning +20 Units +Tensorflow dll
31802: V4.7.1.82 Dec 2019:Tutor-72,EKON23Fixing,WebSocks/Spider,PHP_CGI,JS,OpenOffice,7+overbyte Units
31803: V4.7.4.64 June 2020: EKON24 Fixing, uPSI_SimpleRSS, Json Base Prometheus, neural CAI, Dendron
31804: V4.7.5.20 Jan 2021: few Fundamentals 5.00, JCL fixes, GraphMathLibrary, StringBuilder
31805: V4.7.5.80 July 2021: RSS+, WDCC, P4D_Beta (Python4Delphi), Wininet Threads<br>
31806: V4.7.5.90 October 2021: FLC Vectors, CAI Neuralnet, SingleListClass<br>
31807: V4.7.6.20 October 2022: Threadslist,CAI Neuralnet3,P4D+,EKON25 fixing,klib,HTTPClient2, -c CLI, PSProc<br>
31808: Ref:
31809: https://unibe-ch.academia.edu/MaxKleiner
31810: http://www.slideshare.net/maxkleiner1
31811: http://www.scribd.com/max_kleiner
```

```
31812: http://www.delphiforfun.org/Programs/Utilities/index.htm
31813: http://www.slideshare.net/maxkleiner1
31814: http://s3.amazonaws.com/PreviewLinks/22959.html
31815: http://www.softwareschule.ch/arduino_training.pdf
31816: http://www.jrsoftware.org/isinfo.php
31817: http://www.be-precision.com/products/precision-builder/express/
31818: http://www.blaisepascal.eu/
31819: http://www.delphibasics.co.uk/
31820: http://www.youtube.com/watch?v=av89HAbqAsI
31821: http://www.angelfire.com/hi5/delphizeus/modal.html
31822: http://www.retroarchive.org/garbo/pc/turbopas/index.html
31823: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
31824: http://delphi.org/2014/01/every-android-api-for-delphi/
31825: https://en.wikipedia.org/wiki/User:Maxkleiner
31826: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
31827: https://bitbucket.org/max_kleiner/maxbox3
31828: https://bitbucket.org/max_kleiner/maxbox3/downloads
31829: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
31830: http://www.slideshare.net/maxkleiner1/codereview-topics
31831: http://www.abaecker.biz/images/abakus/images/TAbCompass.gif
31832: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
31833: https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
31834: https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
31835: http://max.kleiner.com/boxart.htm
31836: https://maxbox4.wordpress.com
31837:
31838: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
31839: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
       =renderBasicSearchNarrative&q=%s';
31840: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
31841: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
31842:
31843: SAMPLES Snippets Blocks and Error or Errors or Bugs or Bug List or Trouble Bug Fix Errorlist bugfix codes
31844: ------------------------------------------------  Troubleshooter
31845:
31846: [Error] C:\maXbox\TestApp2\sonysavefeb2014\TestApp\tetris\175_pas_anti_tetris2.txt(1:1): Variable Expected
31847: dont inc an longint in plus Inc(Score+5); ---> solution: Inc(Score);
31848:
31849: Editor: index out of bounds error exception in editor after open or modified
31850: 03/12/2019 20:10:50 V:4.7.1.20 [Max] MAXBOX10 List index out of bounds (147).
31851:
31852: This is due of Editor Indent  ../Options/Show Indent
31853: 4 Workarounds:
31854: 1 simply move cursor to right or left and save (Ctrl+S)
31855: 2 Goto to end or line (147) or what it says and return new line or tab
31856: 3 Disable <Options Show Indent> in menu
31857: 4 Set INDENT=N in maxboxdef.ini for runtime systems
31858:
31859: can not see the Form , Form not visible ---> add Show;
31860: frm:= TmyForm.create(self)
31861: with frm do begin
31862:   Left := 245
31863:   Top := 108
31864:   BorderStyle := bsDialog
31865:   Caption := 'Suggestions'
31866:   ClientHeight := 219
31867:   ClientWidth := 297
31868:   Color := clBtnFace
31869:   //TextHeight := 20
31870:   Show; ------------------------->
31871:
31872: Exception Handling Raise and XRaise
31873: type EPowerException = Exception;
31874:   ...end else
31875:        Xraise( EPowerException.Create('Infinite Result'));
31876:      end;
31877:      if (X > 0) then
31878:      try
31879:        result := exp(N * ln(X));
31880:        exit;
31881:      except
31882:        raise EPowerException.Create('Overflow/Underflow Result');
31883:      end;
31884:
31885: - Object with an Record inside get first a record instance and second you get the attribute
31886:
31887: Proc testprimesPerformance;   // Performance Test
31888: var count, beg, j, r: integer;
31889: var f :boolean;
31890: begin
31891:    processmessagesOFF
31892:    Println(' prime time performance check ?????: ');
31893:    count:= 0;
31894:    beg:= Random(1000000000)+2;
31895:    for it:=beg to beg+5000 do begin
31896:      f:= True;
31897:      j:= 2;
31898:      r:= Round(Sqrt(it));
31899:      while f and (j<=r) do
```

```
31900:      if it mod j = 0 then f:= False
31901:      else inc(j);
31902:      if f then begin
31903:        Print(itoa(it)+' ');
31904:        count:= count+ 1;
31905:        if count mod 8 = 0 then
31906:        Println('');
31907:      end;
31908:    end;
31909:    processmessagesON;
31910: end;
31911:
31912: RSS-FEED Rss Feed
31913: IIndyHTTP:=TIdHTTP.create(self);
31914:     with TSimpleRSS.create(self) do begin
31915:       XMLType:= xtRDFrss;
31916:       IndyHTTP:= iindyhttp;
31917:       LoadFromHTTP(RSS_NewsFeed)
31918:       for it:= 0 to items.count-1 do begin
31919:          //Items[it].description
31920:          writeln(itoa(it)+': '+Items[it].title+'  :'{+items[it].link});
31921:          println(GetSentimentStream7(URLSentimentAPI2,Items[it].title));
31922:       end;
31923:       free;
31924:       IIndyHTTP.Free;
31925:     end; //}
31926:
31927: Ver: 4.7.4.64 (474). Workdir: C:\maXbox\mX47464\maxbox4
31928: ▯▯▯ mX4 executed: 05/11/2020 17:57:05  Runtime: 0:0:39.224  Memload: 60% use
31929:
31930: Func OpenMap(const Data:Str):Bool;
31931: var encURL:Str;
31932: begin
31933:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
31934:   try  //HttpGet(EncodedURL, mapStream);   //WinInet
31935:   Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
31936:    //OpenDoc(Exepath+'openmapx.html;
31937:    S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
31938:   finally encURL:= '';
31939:   end;
31940: end;
31941:
31942: Proc GetGEOMap(C_form,apath:Str; const Data:Str);
31943: var encodedURL:Str; mapStream: TMemoryStream;
31944: begin //encodedURL:= Format(UrlGoogleQrCode,[Width,Height,C_Level, HTTPEncode(Data)]);
31945:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
31946:   mapStream:= TMemoryStream.create;
31947:   try
31948:     Wininet_HttpGet(EncodedURL, mapStream);   //WinInet
31949:     mapStream.Position:= 0;
31950:     mapStream.Savetofile(apath); // OpenDoc(apath);
31951:     S_ShellExecute(apath,'',seCmdOpen);
31952:   finally
31953:     mapStream.Free;
31954:   end;
31955: end;
31956:
31957: Web Request API HTTPSend:
31958:
31959: //http://synapse.ararat.cz/doc/help/httpsend.THTTPSend.html#@Description
31960:
31961: Const aURL='https://api.metadefender.com/v4/file/bzIwMDYxNi1TSW42NDBPVlprTWw3YjRBMQ';
31962:
31963: Begin  //@main
31964:     srlist:= TStringlist.create;
31965:     with THTTPSend.create() do begin
31966:       Headers.Add('apikey: 6b337c92c792174a54acd715ab1aae64');
31967:       writeln(botostr(HTTPMethod('GET',aURl)));
31968:       with TJson.create() do begin
31969:         Parse(StreamtoString3(Document));
31970:         Split(Stringify,'{',srlist)
31971:       end;
31972:       writeln(itoa(resultCode)+':'+resultString+CRLF+srlist.text);
31973:       clear;free;
31974:     end;
31975:     srlist.Free;
31976:
31977:
31978: Proc BtnFactory(a,b,c,d:smallint; title,apic:str;
31979:                              var abtn:TBitBtn; anEvent:TNotifyEvent; afrm:TForm);
31980: begin
31981:   abtn:= TBitBtn.create(afrm);
31982:   with abtn do begin
31983:     parent:= afrm;
31984:     setBounds(a,b,c,d)
31985:     font.size:= 12;
31986:     glyph.LoadFromResourceName(HINSTANCE, apic);
31987:     mXButton(5,5,width, height,12,12,handle);
31988:     caption:= title;
```

```
31989:     onClick:= anEvent As TNotifyEvent;
31990:   end;
31991: end;
31992:
31993: Func GetSystemDirectory2(var S:Str):Bool;
31994: var Len: Int;
31995: begin
31996:   Len:= {Windows.}GetSystemDirectory('Nil', 0);
31997:   if Len > 0 then begin
31998:     SetLength(S, Len);  //writeln(itoa(len))
31999:     Len:= {Windows.}GetSystemDirectory(PChar(S), Len);
32000:     SetLength(S, Len);  //writeln(itoa(len))
32001:     Result:= Len > 0;
32002:   end else
32003:     Result:= False;
32004: end;
32005:
32006: Func RecordsetToXML2(const Recordset: variant):Str;
32007: var
32008:   RS: Variant;
32009:   Stream: TStringStream;
32010: begin
32011:   Result := '';
32012:   //if Recordset = unassigned then Exit;
32013:   Stream:= TStringStream.Create(';
32014:   try
32015:     RS:= CreateOleObject('ADODB.Recordset;
32016:     RS:= Recordset;
32017:     RS.Save(TStreamAdapter.Create(stream,soReference) as IUnknown, adPersistXML);
32018:     Stream.Position := 0;
32019:     Result:= Stream.DataString;
32020:   finally
32021:     Stream.Free;
32022:   end;
32023: end;
32024:
32025: convert JPEG to Bitmap
32026: if version = '4.7.6.20' then begin
32027:   JPGImage:= TJPEGImage.Create;
32028:   try
32029:     JPGImage.LoadFromStream(TResourceStream.Create(hInstance,'Everest','JPEG'));
32030:     with TImage.create(form1) do begin
32031:       setbounds(50,500,500,200)
32032:       parent:= form1;
32033:       Picture.Graphic:= JPGImage;
32034:       stretch:= true;
32035:     end;
32036:   finally
32037:     JPGImage.Free;
32038:   end; //}
32039: end;
32040:
32041: Func BigFactSum(n: integer):Str;
32042:   var tbig1, tbig2, tbig3: TInteger;
32043:   begin
32044:     result:= '0'
32045:     tbig1:= TInteger.create(1);  //facultry
32046:     tbig2:= TInteger.create(0);  //assign
32047:     tbig3:= Tinteger.create(0);
32048:     for it:= 1 to n do begin
32049:       tbig2.assign1(it)
32050:      tbig1.mult(tbig2)
32051:     tbig3.add(tbig1);
32052:   end;
32053:     result:= tbig3.tostring(false)
32054:     tbig3.free;
32055:     tbig2.free;
32056:     tbig1.free;
32057:   end;
32058:
32059: No helper class possible or predict:
32060: USES Classes,StrUtils;
32061:
32062: TYPE
32063: TStringListHelper = CLASS HELPER FOR TStrings
32064: Func ToSQL :Str;
32065: END;
32066: Func TStringListHelper.ToSQL :Str;
32067: VAR S :Str;
32068:
32069: Func QuotedStr(CONST S :Str) :Str;
32070: BEGIN
32071: Result:='''''+ReplaceStr(S,'''','''''')+''''
32072: END;
32073:
32074: BEGIN
32075: Result:='';
32076: FOR S IN Self DO BEGIN
32077: IF Result='' THEN Result:='(' ELSE Result:=Result+',';
```

```
32078: Result:=Result+QuotedStr(S)
32079: END;
32080: IF Result<>'' THEN Result:=Result+')'
32081: END;
32082:
32083: SL:=TStringList.Create;
32084: SL.Add('One');
32085: SL.Add('Two');
32086: SL.Add('Number Three');
32087: SL.Add('It''s number 4');
32088: WRITELN('SELECT * FROM TABLE WHERE FIELD IN '+SL.ToSQL);
32089: SELECT * FROM TABLE WHERE FIELD IN ('One','Two','Number Three','It''s number 4'}
32090:
32091: RSA Crypto Demo
32092: 965_RSA_IBZ_PrivatePublicKey_Form.pas
32093: const _n = '95163118457906561534997167608470014334413 57'; //# p*q = modulus
32094:       _e = '65537';
32095:       _d = '56178431878449531703084636222302833762986 85';
32096:   abig:=TInteger.create(1);
32097:   abig.Assignhex(ASCIItohex(message));
32098:   writeln('big plain text integer: '+abig.Tostring(normal))
32099:   s_encrypt:= BigPowMod(abig.Tostring(normal),_e, _n)     //bigmulu(ap,aq));
32100:   writeln('encrypted text integer '+s_encrypt)
32101:   s_decrypt:= BigPowMod(s_encrypt,_d,_n)                  //bigmulu(ap,aq))
32102:   writeln('decrypted text integer '+s_decrypt)
32103:
32104: Func ConvertHex(HexString :Str; Len : integer) :Str;
32105: begin
32106: if trim(HexString) = '' then Result := '0' else
32107:   Result := IntToStr(StrToInt('$' + trim(HexString)));
32108:    while length(Result) < Len do Result := '0' + Result;
32109: end;
32110:  Proc SetBit(const a,b:Card);
32111:   var d: Int64; // absolute dest;
32112:  begin
32113:     d:= $F000000000+(Int64(a) shl 21)+b shl 6;
32114:  end;
32115:
32116: var alabel: TComponent;   //Exception: A component named lblCell0 already exists.
32117:     lc: integer;
32118: begin
32119: //object frMain: TfrMain
32120:     if assigned(TLabel(FindComponent('lblCell'+intToStr(lc)))) then begin
32121:       write('exist&Nil ')
32122:      TLabel(FindComponent('lblCell'+ IntToStr(lc))).Name:= '';
32123:      alabel:= TLabel(FindComponent('lblCell'+ IntToStr(lc)));
32124:      alabel:= Nil;
32125:      end;
32126:    end;
32127:
32128: //Discuss the Recursion of findfile!
32129: Proc FindFilePattern3(root:str; pattern:str);
32130: var SR: TFindRec;
32131: begin
32132:   root:=IncludeTrailingPathDelimiter(root);
32133:   if FindFirst3(root+'*.*',SR) = TRue then begin
32134:   repeat
32135:    Application.ProcessMessages;
32136:      if ((sr.attributes and faDirectory)= SR.Attributes) and
32137:        (pos('.',SR.name)=0) then
32138:         FindFilePattern3(root+SR.Name,pattern)
32139:      else begin
32140:       if pos(pattern,SR.Name)>0 then
32141:         //Form1.ListBox1.Items.Add(Root+SR.Name);
32142:         writeln(Root+SR.Name);
32143:      end;
32144:   until FindNext3(SR) = False;
32145:     FindClose3(SR);
32146:   end;
32147: end;
32148:
32149: Encoding Decoding
32150: const tname= 'Subject: =?iso-8859-1?Q?Die_besten_W=FCnsche_zum_Neuen_Jahr_2021?=';
32151:
32152:   writeln('mimecharset '+CharsetConversion(tn,ISO_8859_1, UTF_8));
32153:   writeln('mimecharset '+CharsetConversion(tname,ISO_8859_1, win_1252));
32154:   writeln( MimeCharsetToCharsetString(ISO_8859_1) );
32155:
32156:   solution: writeln(DecodeHeader(tname)); or writeln(ForceDecodeHeader(tname));
32157:   -----> Subject: Die besten Wünsche zum Neuen Jahr 2021
32158:   writeln(UTF8toAnsi(ALUTF8HTMLdecode('Kaufbest&auml;tigung')))
32159:   -----> Kaufbestätigung
32160:
32161: RegEx by example:
32162: The Regexlibrary is inbuilt so it works independent of Operating System:
32163:
32164: Proc RegEX_ExtractDemo;
32165: var regEx: TRegExpr; //or OleVariant if HISUtils;
32166: InStr, ResStr:Str;
```

```
32167: begin
32168:    ResStr:='';
32169:    InStr:= 'Please e-mail us at support@mycompany.com or to sales@mycompany.com';
32170:    // Create a regular expression
32171:    regEx:= TRegExpr.create; //HISUtils.RegExpr;
32172:    // Set regular expression pattern that specifies an e-mail address
32173:    regEx.Expression:='\w+(\.[\w-]+)*@([\w-]+\.)+[a-zA-Z]{2,6}';
32174:    // Execute search
32175:    if regEx.Exec(InStr) then Repeat
32176:      ResStr:= ResStr + regEx.Match[0] + '; ';
32177:    Until not regEx.ExecNext;
32178:
32179:    //Log.Message('These e-mails were extracted: '+ResStr);
32180:    writeln('These e-mails were extracted: '+ResStr);
32181:    // Posts the following to the log:
32182:    //'These e-mails were extracted: support@mycompany.com; sales@mycompany.com;'
32183: end;
32184:
32185: Proc LoadFilesByMask(lst: TStrings; const SpecDir, WildCard:Str);
32186: var SearchRec: TFindRec;
32187: begin lst.Clear;
32188:   if FindFirst3(SpecDir + WildCard, SearchRec) then
32189:   repeat
32190:     lst.Add(SpecDir + SearchRec.Name);
32191:   until FindNext3(SearchRec) = false;
32192:   FindClose3(SearchRec);
32193: end;
32194:
32195: Proc regExtest;     //SynRegExpr unit     RegEx Snippet
32196:  var RegEx: TRegExpr; // This is a class, need to be freed!
32197:    //Match: TMatch;
32198:    i: Integer;
32199:  begin
32200:    with TRegExpr.Create do begin
32201:      Expression:= ('^v(ol\.?|olume)?\s*([0-9]+)$');
32202:      if Exec('vol.3456') then
32203:        for i:=0 to SubExprMatchCount do
32204:           PrintF('Group %d  : %s', [i, Match[i]]);
32205:       //Until not regex.ExecNext;
32206:      Free;
32207:    end
32208:  end;
32209:
32210:  //(?i) ---> case insensitive
32211:  //(?i) case-insensitive mode ON
32212:  //(?-i) case-insensitive mode OFF
32213:  writeln(getMatchString('(?i)G[a-b].*', 'this GB string'));
32214:  PrintF('CountPos: %d',[CountPos(uppercase('Selftest'),uppercase(sr))])
32215:
32216:  Set Mapping
32217:  There are some differences in the use of set operators in PascalScript and Delphi.
32218:  In DelphiScript the [...] set operator is not supported or the in operator not always works.
32219:  Set mapping with ord(CS) works:
32220:  type  TMimeCharset = (
32221:          CS_DEFAULT1,
32222:          CS_NOTMAPPED,
32223:          UTF_8,
32224:          WIN_1250,
32225:          WIN_1254,
32226:          WIN_1255,
32227:          WIN_1257,
32228:          WIN_1258,
32229:          ISO_8859_1,
32230:          ISO_8859_2)
32231:
32232:     Cs: TMimeCharSet;
32233:     CS:= CS_NOTMAPPED;
32234:     I := Ord(CS);
32235:     CharsetInfos[I].MimeCharset    := CS_NOTMAPPED;
32236:     //CharsetInfos[I].CodePage     := ERR_CP_NOTMAPPED;
32237:     CharsetInfos[I].MimeName       := ('');
32238:     CharsetInfos[I].FriendlyName   := 'sUnicodeUTF844';
32239:
32240:   EnvironmentString:
32241:   envlist:= Tstringlist.create;
32242:   StrtoList(GetEnvironmentString, envlist, CR)
32243:      writeln(envlist.Text);
32244:      writeln('envlist.count: '+itoa(envlist.count));
32245:   envlist.Free;
32246:
32247: Note arrays in PascalScript cannot be initialized statically. For instance, the following code raises an
       error:
32248: var I : array[0..1] of Integer = (3,4) // <-- Error!!!
32249:
32250: To solve problem, initialize array elements in your code. There are several ways to do this:
32251:     Initializing each element individually:
32252:     DelphiScript
32253:     var a : array [0..3] of string;
32254:     begin
```

```
32255:       a[0] := 'Element 0';
32256:       a[1] := 'Element 1';
32257:       a[2] := 'Element 2';
32258:       a[3] := 'Element 3';
32259:       ...
32260:     end;
32261:
32262: Initializing elements in a loop:
32263: DelphiScript
32264: var a : array [0..3] of string;
32265: begin
32266:   for i := 0 to 3 do
32267:     a[i] := 'Element ' + IntToStr(i);
32268:   ...
32269: end;
32270:
32271: var array with 2 dimensions:
32272: Var frMain     : TfrMain;
32273:     iXPos      : array [1..3] of array [ 1..3] of Integer;
32274:     iOPos      : array [1..3] of array [ 1..3] of Integer;
32275:
32276: Constructor with override do most have a second one:
32277:
32278: var Images : TObjectList;
32279:     //Create TObject list with AOwnsObjects set to True means that destroying
32280:     //the object list will also destroy all of the objects it contains
32281:     //NOTE: On ARC compiler destroying TObjectList will only remove the reference
32282:     //to the objects and they will be destroyed only if thir reference count
32283:     //drops to 0
32284:
32285:     Images := TObjectList.Create;
32286:     Images := TObjectList.Create1(true);   //override
32287:     a1 := TStatArray.Create1(0); // Growing array.
32288:
32289: Return Values
32290: Use ArcTan3 instead of ArcTan
32291: Func ArcTan3(const X : Extended) : Extended');  //bugfix of ArcTan()
32292:
32293: To return a value from a function, use Result instead of Func name. Using Func name will cause an error:
32294:
32295: Func Calc(x, y) : Integer;
32296: begin
32297:   Calc := (x - y) * (x + y); // Error !!!
32298:   Result := (x - y) * (x + y); // Correct variant
32299: end;
32300:
32301: Proc TStringsExchange(Index1, Index2: Integer; stlist: TStringlist);
32302: var
32303:   TempObject: TObject;
32304:   TempString:Str;
32305: begin
32306:   stlist.BeginUpdate;
32307:   try
32308:     TempString := stlist.Strings[Index1];
32309:     TempObject := stlist.Objects[Index1];
32310:     stlist.Strings[Index1] := stlist.Strings[Index2];
32311:     stlist.Objects[Index1] := stlist.Objects[Index2];
32312:     stlist.Strings[Index2] := TempString;
32313:     stlist.Objects[Index2] := TempObject;
32314:   finally
32315:     stlist.EndUpdate;
32316:   end;
32317: end;    //*)
32318:
32319: Integer Overflow of a DLL import:
32320: https://forum.lazarus.freepascal.org/index.php?topic=26116.0
32321: usually you set stdcall convention for a dll
32322: Func  csoundgetversion: Longint;
32323:      external 'csoundGetVersion@csound64.dll stdcall';
32324:
32325: but for procedures or void try with cdecl:
32326:   Proc  csoundSleep(duration: word);
32327:      external 'csoundSleep@csound64.dll cdecl';  //no stdcall (int overflow)
32328:
32329: [Error]..\maXbox4\examples\558_hirestimer2test2.txt(12:1): program is not allowed at this position cause:
       strange typecasts like: TStream(Nil) or TString(null)
32330:
32331: set a DLL Template: dll+Ctrl J
32332:
32333: //https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-internetsetoptiona
32334:
32335: Func  InternetSetOption(hinternet:HINTERNET; dwoption: DWord;
32336:                          buffer: Ansistr; bufflen: DWord):Bool;
32337:     external 'InternetSetOptionA@wininet.dll stdcall';
32338:
32339:  { BOOLAPI InternetSetOptionA(
32340:     HINTERNET hInternet,
32341:     DWORD     dwOption,
32342:     LPVOID    lpBuffer,
```

```
32343:     DWORD     dwBufferLength
32344:    );  }
32345:
32346: Proc testReverse_Proxy;
32347:  var  ConnectionHandle: HINTERNET;
32348:   UserName: Ansistr;
32349:   res:Bool;
32350:   EmptyParam: OleVariant;
32351: begin
32352:   // Init ConnectionHandle here
32353:   ... res:= InternetSetOption(ConnectionHandle,
32354:                           INTERNET_OPTION_PROXY_USERNAME,
32355:                           UserName, Length(UserName) + 1);
32356:
32357: Proc Str() doesnt return a value : use intToStr()
32358:
32359: TBytes = array of byte : call of the library problem
32360: Proc SaveBytesToFile(const Data: TBytes; const FileName:Str);  //const OK!
32361: Func ShiftByteArray(var A: TBytes): Byte;      // var Not OK!
32362: in script
32363: Func ByteToString(var Value: TBytes):Str;  // var and const OK!
32364: var I: integer;
32365:     S :Str;
32366:     Letra: char;
32367: begin
32368:     S := '';
32369:     for I := Length(Value)-1 Downto 0 do begin
32370:         letra := Chr(Value[I] + 48);
32371:         S := letra + S;
32372:     end;
32373:     Result := S;
32374: end;
32375:
32376: ------------------------------ Func wrapper ---------------------------
32377: Its better to call a Func wrapper than the Func itself:
32378:
32379: Proc QuickSortX(StringList : TStringList; L, R: Integer);
32380: Var
32381: I, J, P : Integer;
32382: Begin
32383:   P := StringList.Count;
32384:   If (L >= P) or (R >= P) Then Exit;
32385: {……}
32386:
32387: Proc SortList(StringList : TStringList); //Func wrapper hides paras
32388: Begin
32389:   QuickSortX(StringList, 0, StringList.Count - 1)
32390: End;
32391: ------------------_
32392: tst:= TStringlist.create;
32393: tst.add('7')
32394: tst.add('9')
32395: writeln(objtostr(tst))
32396: SortList(tst) //call to Func wrapper
32397:
32398: Hashmap or Dictionary or stringhash or map
32399:
32400: with TStringHashMap.create(CaseSensitiveTraits, 50) do begin
32401:       //Add( const s :Str; const p);
32402:      data['this is']:= '345'
32403:      data['this is2']:= '3456'
32404:      data['this is25']:= '3456677'
32405:      writeln(botostr(Has('this is2')))
32406:      setlength(sr,120)
32407:      stri:= pchar(data['this is'][10])
32408:      writeln(itoa(length(stri)))
32409:      writeln(stri)
32410:      writeln(itoa(Count));
32411:      writeln(itoa(HashSize))
32412:      clear;
32413:      free;
32414:     end;
32415:
32416: Proc stringHashtest;
32417:  var hash : TStringHash2;
32418:   //Here's unit with those TIntegerHash2 and TStringHash2, TObjectHash2 described @
32419:  basehash: THash2;
32420:  begin
32421:    hash:= TStringHash2.Create;
32422:    try
32423:      hash['one']:= 'viens value';
32424:      hash['two']:= 'divi value';
32425:      //ShowMessage(hash['one']);
32426:      //ShowMessage(hash['two']);
32427:      if hash.exists('one') then
32428:        writeln(hash['one']);
32429:      writeln(hash['two']);
32430:      hash.Rename('one', 'onerename');
32431:      writeln(hash['onerename']);
```

```
32432:    finally
32433:      hash.Free;
32434:    end;
32435:  end;
32436:
32437:  // word frequency snippet
32438:  for it:= 1 to wordlist.count-1 do begin
32439:    //writeln(itoa(it)+' '+Utf8ToAnsi(wordlist[it]));
32440:    aword:= lowercase(Utf8ToAnsi(wordlist[it]))
32441:    regex.RegEx:= '\w+';
32442:    regex.Subject:= aword;
32443:    if regex.match then aword:= regex.groups[0];  //word filter
32444:    if aword <> '' then
32445:    if dict.exists(aword) then
32446:      dict[aword]:= inttostr(strtoint(dict[aword])+1) else
32447:         dict[aword]:= '1';
32448:  end;
32449:   // iterate through all values
32450:   dict.Restart
32451:   for it:= 1 to (dict.itemcount-1) do
32452:     if dict.next then
32453:       writeln(dict.currentkey+':  '+dict.items[dict.currentkey]) ;  //}
32454:
32455: for each method - for in metheod
32456: theres no for each operator but we can map it:
32457:
32458: //for c := 'a' to 'z' do
32459:    for c:= ord('a') to ord('z') do
32460:
32461: //for ch in adata do begin
32462:   for chi:= low(adata) to high(adata) do begin
32463:      ch:= adata[chi]
32464:
32465:   c: AnsiChar; Dec: Ansistr
32466:   //for c in Dec do
32467:   for it:= 1 to length(dec) do begin
32468:     c:= dec[it]
32469:
32470:   for in and set or include charset
32471:   var
32472:     include: charset; ex:str;
32473:   //if ex[i] in ['a'..'z'] then include:=include+[ex[i]]; -type mismatch
32474:   alternatives:
32475:   1. if (ex[i] > 'a') and (ex[i] < 'z') then //in ['a'..'z']
32476:   2. if ex[i] in StrToCharSet(lowercase(LETTERSET))
32477:                then include:= include+strtoCharset(ex[i]);
32478:
32479:   use strtocharset and chartostrset as helper Func with sets
32480:   stest[2]:= '^';
32481:     if stest[2] in
32482:        StrToCharSet(lowercase(LETTERSET)+'_'+'?'+' '+'-'+'*'+'/'+'ö'+'^') then
32483:              writeln('inside') else writeln('outside');
32484:   >>> inside
32485:   writeln(charsettostr(strtocharset('*'+'?')))
32486:
32487:   function IsSTRLegal2(c: string): boolean;
32488:   begin
32489:     //result:= (Length(c)=1) and (c[1] in StrToCharSet('+'+ '-'+ '.'+ DIGISET));
32490:     result:= (Length(c)=1) and (c[1] in StrToCharSet('+-.,:'+DIGISET));
32491:   end;
32492:
32493: Func IsInvalid(s:Str):Bool;
32494: var c: char;
32495:   //leters: set of char;
32496:   leters: Charset;
32497:   firstE:Bool;
32498: begin
32499:   //writeln(itoa(IndexStr(s,['k'])));
32500:   //Result := (Length(s) < 3) or (s.IndexOf('k') = -1) or (s.Length > 9);
32501:   Result:= (Length(s) < 3) or (pos('k',s) = 0) or (Length(s) > 9);
32502:   if not Result then begin
32503:     leters:= ['d', 'e', 'g', 'k', 'l', 'n', 'o', 'w'];
32504:     firstE:= true;
32505:     //for c in s do
32506:     for it:= 1 to length(s) do begin
32507:       c:= s[it];
32508:       if c in leters then
32509:         if (c = 'e') and (firstE) then
32510:           firstE:= false
32511:         else
32512:           //Exclude(leters, AnsiChar(c))
32513:           leters:= leters-strtoChars(c)
32514:       else begin
32515:         result:= true
32516:         exit;
32517:       end;
32518:     end;
32519:   end;
32520: end;
```

```
32521:
32522:   Func TrimRight7(Str:Str):Str;
32523:    var i: Integer;
32524:    begin
32525:        i:= Length(Str);
32526:        //while (i > 0) and (Str[i] in [' ', #9]) do
32527:        while (i > 0) and (Str[i] in strtocharset(' '+#9)) do
32528:            i:= i - 1;
32529:        Result:= Copy(Str,1,i);
32530:    end;
32531:
32532:   if MatchesMask('abc.xyz', 'a*.*',true) then
32533:     writeln('File name is OK.')
32534:   else
32535:     writeln('Invalid file name.');
32536:
32537:   Find all files of a defined date:
32538:   8 C:\>dir /s /o N *.dcu | find "16/06/2020" > C:\maXbox\maxboxunitalldisk_sort.txt
32539:
32540:  Operator Overloading Surrogate:
32541:  type complexreal = record
32542:                re : real;
32543:                im : real;
32544:             end;
32545:   { TFR }
32546:   { frequency response }
32547:   TFR = record
32548:     M, phi: extended; { magnitude and phase }
32549:     F: complexreal;   { complex FR (F = M(omega) * exp(i * phi(omega)) }
32550:   end;
32551:
32552:  Func cexp (z : complexreal) : complexreal;
32553:     { exponantial : r := exp(z) }
32554:     { exp(x + iy) = exp(x).exp(iy) = exp(x).[cos(y) + i sin(y)] }
32555:     var expz : real;
32556:     begin
32557:        expz := exp(z.re);
32558:        cexp.re := expz * cos(z.im);
32559:        cexp.im := expz * sin(z.im);
32560:     end;
32561:  Func complexmulu(z1 : complexreal; r : real): complexreal;
32562:     { multiplication : z := z1 * r }
32563:     begin
32564:        result.re := z1.re * r;
32565:        result.im := z1.im * r;
32566:     end;
32567:   FFr.F:= FFr.M * cexp(i * FFr.phi); { M and phi encoded in polar coordinates }
32568:   //complextemp:= complexmulu(i,FFr.phi);
32569:   FFr.F:= complexmulu(cexp(complexmulu(i,FFr.phi)), FFr.M); { M and phi encoded in polar }
32570:
32571: replace move with iteration to move()
32572:
32573: // big note to move the move to an iteration
32574:    For i:= BListLenght downto 1 do BList[i]:= BList[i-1]; //or
32575:
32576:    move(Blist[0],Blist[1],BListLenght*SizeOf(Blist[0]));
32577:
32578:  Array of const
32579:  you cant set an array like :
32580:  {CONST} VAR t : TA;
32581: (* =
32582:                 ((0,3,1,7,5,9,8,6,4,2),
32583:                  (7,0,9,2,1,5,4,8,6,3),
32584:                  (4,2,0,6,8,7,1,3,5,9),
32585:                  (1,7,5,0,9,8,3,4,2,6),
32586:                  (6,1,2,3,0,4,5,9,7,8),
32587:                  (3,6,7,4,2,0,9,5,8,1),
32588:                  (5,8,6,9,7,2,0,1,3,4),
32589:                  (8,9,4,5,3,6,2,0,1,7),
32590:                  (9,4,3,8,6,1,7,2,0,5),
32591:                  (2,5,8,1,4,3,6,7,9,0));  *)
32592:
32593:   but you can convert:
32594:
32595:   const arrstr=  '0,3,1,7,5,9,8,6,4,2'+
32596:                  '7,0,9,2,1,5,4,8,6,3'+
32597:                  '4,2,0,6,8,7,1,3,5,9'+
32598:                  '1,7,5,0,9,8,3,4,2,6'+
32599:                  '6,1,2,3,0,4,5,9,7,8'+
32600:                  '3,6,7,4,2,0,9,5,8,1'+
32601:                  '5,8,6,9,7,2,0,1,3,4'+
32602:                  '8,9,4,5,3,6,2,0,1,7'+
32603:                  '9,4,3,8,6,1,7,2,0,5'+
32604:                  '2,5,8,1,4,3,6,7,9,0';
32605:
32606: procedure InitTable4;
32607: var i,j,k: integer; repl: string;
32608: begin
32609:  repl:= StringReplace(arrstr,',','',[rfReplaceAll]);
```

```
32610:  k:= 0
32611:  for i:= 0 to 9 do
32612:    for j:= 0 to 9 do begin
32613:        inc(k)
32614:        t[i][j]:= strtoint(repl[k])
32615:    end;
32616: end;
32617:
32618: Convert Static Array to Dynamic Array
32619:
32620: procedure TForm1Button1Click(Sender: TObject);
32621: var
32622:   ADate: TDateTime;
32623:   days: array[1..7] of string;
32624: begin
32625:   days[1]:= 'Sunday';
32626:   days[2]:= 'Monday';
32627:   days[3]:= 'Tuesday';
32628:   days[4]:= 'Wednesday';
32629:   days[5]:= 'Thursday';
32630:   days[6]:= 'Friday';
32631:   days[7]:= 'Saturday';
32632:   ADate := StrToDate('31/01/1981');
32633:   ShowMessage('31/01/1981' + ' was a ' + days[{SysUtils.}DayOfWeek(ADate)]);
32634: end;
32635:
32636: //Incompatible types: 'Dynamic array' and 'array of string' to open array
32637:
32638: procedure TForm1Button1ClickDynArray(Sender: TObject);
32639: var aDate: TDateTime;
32640:     days: array of string;
32641: begin
32642:   setLength(days,7);
32643:   days:=['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'];
32644:   aDate:= StrToDate('31/01/1981');
32645:   ShowMessage('31/01/1981'+' was a '+ days[{SysUtils.}DayOfWeek(aDate)-1]);
32646: end;
32647: // with SplitStr
32648: begin
32649:   aDate:= StrToDate('31/01/1981');
32650:   days:= SplitStr('Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday',',');
32651:   writeln('31/01/1981'+' was a '+ days[{SysUtils.}DayOfWeek(aDate)-1]);
32652: end;
32653:
32654:  AssignFile versus TStringlist
32655:  ---------------------------------
32656:  In addition to the old style file handling routines mentioned above, a new system exists that uses
       concept of streams (- of data) at a higher abstraction level. Means data can be read from or written to
       any location (disk, memory, hardware ports etc.) by one uniform interface.
32657:  In addition, most string handling classes have the ability to load and save content from/to a file. These
       methods are usually named SaveToFile and LoadFromFile. in // old style assign file
32658:
32659:     filelist:= TStringlist.create;
32660:     // assignfile(f,opendialog1.filename);
32661:     filelist.loadfromfile( filename)
32662:     // reset(f);
32663:     // while not eof(f) do
32664:     for it:= 0 to filelist.count-1 do begin
32665:       //readln(f,line);
32666:       line:= filelist.strings[it];
32667:       writeln(line)
32668:       if (length(line)>2) and (copy(line,1,2)='ID') then {skip Id records}
32669:       else begin {it's a data record}
32670:         if totcount=0 then begin
32671:           savedecimalseparator:=decimalseparator;
32672:           if pos(',',line)>0 then setdecimalseparator(',')
32673:           else setdecimalseparator('.');
32674:         end;
32675:          errcode:=0;
32676:         try
32677:           x:=strtofloat(trim(line));
32678:           except errcode:=1;
32679:         end;
32680:         if (errcode=0) and (x>0)and(x<=amaxtime)
32681:         then begin
32682:          inc(samps[round(maxindex*x)]); {increment the count for the bar for x}
32683:          inc(totcount);
32684:         end;
32685:       end;
32686:     end;
32687:     // closefile(f);
32688:     filelist.Free;
32689:
32690: Random Generator:
32691: https://github.com/TurboPack/SysTools/blob/master/source/StRandom.pas
32692:
32693:  strand:= TStRandomSystem.create(42); //if 0 then randomseed, else randomstate
32694:   sleep(200)
32695:   writeln(floattostr(strand.asFloat))
```

```
32696:   writeln(itoa(strand.asInt(10000)))
32697:   writeln(itoa(strand.asInt($E15)))
32698:   UniformityTest2(strand, ChiSquare, DegsFreedom)
32699:   writeln('ChiSquare: '+floattostr(ChiSquare)+' ,degsfreedom:'+itoa(degsfreedom));
32700:   UniformityTest3(strand, ChiSquare, DegsFreedom)
32701:   writeln('ChiSquare: '+floattostr(ChiSquare)+' ,degsfreedom:'+itoa(degsfreedom));
32702:  strand.Free;
32703:
32704: Https Post API: http://www.softwareschule.ch/download/maxbox_starter94.pdf
32705: translator for different countries:
32706: var httpReq,hr: Olevariant;
32707: try
32708:   hr:= httpReq.Open('POST','https://libretranslate.pussthecat.org/translate', false);
32709:   httpReq.setRequestheader('user-agent',
32710: 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0');
32711:   httpReq.setRequestheader('content-type','application/x-www-form-urlencoded');
32712: //httpReq.setRequestheader('X-RapidAPI-Host','nlp-translation.p.rapidapi.com');
32713:   if hr= S_OK then HttpReq.Send('q='+HTTPEncode(feedstream)+
32714:                                 '&source=es'+'&target=en');
32715:   If HttpReq.Status = 200 Then
32716:        result:= HttpReq.responseText
32717:     Else result:= 'Failed at getting response:'+itoa(HttpReq.Status)+HttpReq.responseText;
32718:      //writeln('debug response '+HttpReq.GetAllResponseHeaders);
32719:   finally
32720:     httpreq:= unassigned;
32721:
32722: File Size Reference:  filegetsize()
32723: Func FileGetSizeX(const FileName:Str): Int64;
32724: {$ifndef fpc}
32725: var
32726:   FileInfo: TSearchRec;
32727: {$endif}
32728: begin
32729: {$ifdef fpc}
32730:   Result:= FileUtil.FileSize(FileName);
32731: {$else}
32732:   // from LCL FileUtil code
32733:   FileInfo.Name:= Filename;
32734:   FileInfo.FindHandle:= {Windows.}FindFirstFile({Windows.}{LPTSTR}(FileInfo.Name)
32735:                                                 ,FileInfo.FindData);
32736:   if FileInfo.FindHandle={Windows.}Invalid_Handle_value then begin
32737:     Result:=-1;
32738:     Exit;
32739:   end;
32740:   Result:= (int64(FileInfo.FindData.nFileSizeHigh)
32741:                     Shl 32)+ FileInfo.FindData.nFileSizeLow;
32742:   {Windows.}FindCloseW(FileInfo.FindHandle);
32743: {$endif}
32744: end;
32745: { this func is a wrapper for platform-specific file fns
32746:   IE a way to get rid of those portability warnings
32747:   and a place to put the equivalent linux fns }
32748:
32749: RTL OSLibrary+
32750: Func MySoundcard: Longint; external 'waveOutGetNumDevs@winmm.dll stdcall';
32751: Func isSound:Bool; begin result:= mySoundcard > 0 end;
32752: Func StringtoHex(Data:Str):Str;
32753: Func GetAnsistrRefcount(const S:Str):Card;
32754: Func WideStringToString(const ws: WideString; codePage: Word): Ansistr;
32755: Func StringToWideString(const s: Ansistr; codePage: Word): WideString;
32756: Proc FreeObjectList(List: TObjectList);
32757: Func SecondToTime(const Seconds:Card): Double;
32758: Func CopyDir2(const fromDir, toDir:Str):Bool;;
32759: Func MoveDir(const fromDir, toDir:Str):Bool;;
32760: Func DelDir(dir:Str):Bool;;
32761: Proc DeleteScansRect(Src, Dest: TBitmap; rs, rd: TRect);
32762: Proc FadeIn(ImageFileName: TFileName; aForm1: TForm);
32763: Proc FadeOut(ImageFileName: TFileName);
32764: Proc FadeOut32(const Bmp: TImage; Pause: Int);
32765: Func CheckBDEInstalled:Bool;   //IsBDE
32766: Proc SaveStringToStream( const Str :Str; Stream : TStream);
32767: Func LoadStringFromStream( Stream : TStream) :Str;
32768: Func WaitForSyncObject(SyncObject:THandle;Timeout:Card;BlockInput:Bool):Card;
32769: Func ProcessMessage :Bool;
32770: Proc ProcessMessages( Timeout : DWORD);  //without application!
32771: Func GetNumberOfEventLogRecords(hEventLog:THandle;var NumberOfRecords:DWORD): BOOL;
32772: Func GetOldestEventLogRecord(hEventLog:THandle; var OldestRecord : DWORD): BOOL;
32773: ex.:EventLog:=RegisterEventSource('0',('maxbox4.exe'));if GetNumberOfEventLogRecords(eventlog,recs)
32774: Proc InitializeLocaleSupport); Proc TerminateLocaleSupport;
32775: Func StrReplaceRegEx(const Subject,Pattern:Ansistr; Args:array of const : Ansistr;
32776: Func TickCountToDateTime( Ticks :Card) : TDateTime;
32777: Proc OutputDebugStringmax( const S :Str);
32778: Proc OutputDebugFormat( const FmtStr :Str; Args : array of const);
32779: Func IsAppRunningInDelphi :Bool;
32780: Func RunExecutable( const AFileName :Str; AWaitForIt :Bool) : DWORD;
32781: Func SystemCodePage : Int;
32782: Func CPUSpd:Str;
32783: Func CPUSpeed:Str;
32784: Func BigFib(n: Int):Str;  //BigFibo
```

```
32785: Func BigFac(n: Int):Str;  //BigFact
32786: Func UpTime:Str;
32787: Func NetLogon(const Server,User,Password:WideString; out ErrorMessage:str) :Bool;
32788: Func NetLogoff( const Server, User, Password : WideString) :Bool;
32789: Proc ErrorNamedPipe( const Message :Str);
32790: Proc BroadcastChange; //method that broadcasts the necessary message WM_SETTINGCHANGE
32791: Func WriteFile2(hFile:THandle;const Buffer,nNumberOfBytesToWrite:DWORD; var lpNumberOfBytesWritten:DWORD;
       lpOverlapped : Tobject):BOOL;
32792: Func ReadFile2(hFile:THandle;var Buffer,nNumberOfBytesToRead:DWORD;var
       lpNumberOfBytesRead:DWORD;lpOverlapped : Tobject):BOOL;
32793: // Security helper functions
32794: Proc InitializeSecurity(var SA: TSecurityAttributes);
32795: Proc FinalizeSecurity(var SA: TSecurityAttributes);
32796: // Pipe helper functions
32797: Proc CheckPipeName(Value:Str);
32798: Proc ClearOverlapped(var Overlapped: TOverlapped; ClearEvent:Bool);
32799: Proc CloseHandleClear(var Handle: THandle);
32800: Func ComputerName2:Str;
32801: Proc DisconnectAndClose(Pipe: HPIPE; IsServer:Bool = True);
32802: Func EnumConsoleWindows(Window: HWND; lParam: Int): BOOL; stdcall;
32803: Proc FlushMessages;
32804: Func IsHandle(Handle: THandle):Bool;
32805: Proc RaiseWindowsError;
32806: Func UpTime:Str; //OS uptime
32807: Func CharSetToCP(ACharSet: TFontCharSet): Int;
32808: Func CPToCharSet(ACodePage: Int): TFontCharSet;
32809: Func TwipsToPoints(AValue: Int): Int;
32810: Func PointsToTwips(AValue: Int): Int;
32811: Proc LoadGraphicFromResource(Graphic:TGraphic; const ResName:str; ResType:PChar);
32812: Func SplitStr(sInput:str; Delimiter:str): TStringArray;
32813: Func GetDataFromFile2(sFileName: Ansistr): Ansistr;
32814: Func ExtractFileNameWithoutExt(const FileName:Str):Str;
32815: Func SubstringCount (Substring :Str; Str :Str): Int;
32816: Func loadForm(vx, vy: smallint): TForm;   //alias getForm()
32817: Func getForm(vx, vy: smallint): TForm;   //alias getForm()
32818: Func getForm2(vx,vy:smallint;acolor:TColor;aname:str):TForm; //sizeable!
32819: Proc paintProcessingstar2(ppform: TForm);
32820: Func ContentTypeGetExtn(Const Content:Str; var CLSID:Str):Str;
32821: Func ContentTypeFromExtn(Const Extension:Str):Str;
32822: Func DateTimeDiff2(Start, Stop : TDateTime) : int64;
32823: Func checkSystem:Str;;
32824: Func getSystemReport:Str;;
32825: Func SystemCheck:Str;;
32826: Func SetThreadDesktop( hDesktop : HDESK) : BOOL;
32827: Func RSAEncrypt(aval, apow, amod:Str):Str;;
32828: Func RSADecrypt(aval, apow, amod:Str):Str;;
32829: Func EncryptRSA(aval, apow, amod:Str):Str;;
32830: Func DecryptRSA(aval, apow, amod:Str):Str;;
32831: Func SwitchToThread : BOOL;
32832: Func CloseDesktop( hDesktop : HDESK) : BOOL;
32833: Func GetThreadDesktop( dwThreadId : DWORD) : HDESK;
32834: Func SetSyscallHook():Bool;;
32835: Func SetSwapcontextHook():Bool;;
32836: Func UnhookAll():Bool;;
32837: Func getWorld:Str;;
32838: Func getIPConfigAll:Str;;
32839: Func getIPConfig:Str;;
32840: Func WinsockEnabled:Bool;
32841: Func HTTPEncode2(const AStr:Str):Str;
32842: RSAEncryptStr(const EncryptionType:TRSAEncryptionType;const PublicKey:TRSAPublicKey;const Plain:Ansistr):
       Ansistr;
32843:
32844: ////////////////////////////////////////////////////////////////////////////
32845: // Object instance functions
32846: ////////////////////////////////////////////////////////////////////////////
32847: Func AllocateHWnd(Method: TWndMethod): HWND;
32848: Proc DeallocateHWnd(Wnd: HWND);
32849: Func DeleteDC( DC : HDC) : BOOL;
32850: Func DeleteMetaFile( p1 : HMETAFILE) : BOOL;
32851: Func DeleteObject( p1 : HGDIOBJ) : BOOL;
32852: Func FormInstanceCount2(AFormClass: TFormClass): Int;;
32853: Func FormInstanceCount(const AFormClassName:Str): Int;;
32854:  ClassN(CL.FindClass('Class of TForm'),'TFormClass;
32855: Func ObjToStr( const O : TObject) :Str
32856: MessageBox(0,PChar('CPU speedis '+CPUSpd+' MHz'),'CPU Speed Check',MB_IconInformation+MB_OK);
32857: TKLogEvent', 'Proc (Sender : TObject; Code : TKLogType; const Text :Str);
32858: AddRegisteredVariable('FormatSettings','TFormatSettings;  //at sysutils!
32859: AddRegisteredVariable('mouse','TMouse;  //at controls HPIPE', 'THandle;
32860:
32861: Func ListIdentical2(l1,l2:TStringList):Bool;
32862: begin  Result:= False;
32863:   if l1.count = l2.count then begin
32864:     for it:= 0 to l1.count-1 do
32865:       if (l1[it] <> l2[it]) then Exit;
32866:     Result:= True;
32867:   end;
32868: end;
32869:
32870: // Converts String To Hexadecimal Maybe usefull for a hex-editor
```

```
32871: // For example: Input = 'ABCD' Output = '41 42 43 44'
32872: Func StringtoHex(Data:Str):Str;
32873: var i, i2: Int; s:Str;
32874: begin
32875:   i2:= 1;
32876:   for i:= 1 to Length(Data) do begin
32877:     Inc(i2);
32878:     if i2 = 2 then
32879:       s:= s + ' ';  i2 := 1;
32880:       s:= s + IntToHex(Ord(Data[i]), 2);
32881:   end;
32882:   Result:= s;
32883: end;
32884:
32885: Func XRTLIsInMainThread:Bool;
32886: begin
32887:   Result:= GetCurrentThreadID = MainThreadID;
32888: end;
32889:
32890: computers:= TStringList.Create;
32891:   FindComputers(computers)
32892:   for it:= 0 to computers.count-1 do
32893:           writeln(computers.strings[it]); computers.free;
32894:
32895:   s:= (Sender AS TStringGrid).Cells[col,row];
32896:   Sender AS TStringGrid).Canvas.Font.Color := clBlack;
32897:   twidth:=  (Sender AS TStringGrid).Canvas.TextWidth('X;
32898:   (Sender AS TStringGrid).Canvas.TextRect(Rect, Rect.Left + twidth div 2,
32899:                 YCenter(Rect, (Sender AS TStringGrid).Canvas,s),s)
32900:
32901: Proc | EXceptionsOnmaXbox;  Exception Handling EXceptions On maXbox
32902: var filename,emsg:str;
32903: begin
32904:  filename:= '';
32905:  try
32906:    if filename = '' then
32907:      RaiseException(erCustomError, 'Exception: File name cannot be blank;
32908:  except
32909:    emsg:= ExceptionToString(ExceptionType, ExceptionParam);
32910:      //do act with the exception message i.e. email it or save to a log etc
32911:    writeln(emsg)
32912:  end;
32913: end;
32914:
32915: XRaise raise an exception without except block:
32916: if (DLLHandle < HINSTANCE_ERROR) then
32917:    Xraise(Exception.Create(DLLName + ' library can not be loaded or not found.'+
32918:                              SysErrorMessage(GetLastError)));
32919:
32920: Proc CallTicker(msecs: Int);
32921: begin
32922:  with TStopWatch.create do begin
32923:    start
32924:      | //Delay2StopWatch(msecs)
32925:    stop
32926:    writeln('Stop Watch CPU Tester: '+getValueStr)
32927:  end;
32928: end;
32929:
32930: MessageDlg Example:
32931: mr:=messagedlg('Save clock first?',mtconfirmation,[mbYes,mbNo,mbCancel],0);
32932:     if mr=mryes then writeln('yes')
32933:
32934: Func getFilefrom
32935: if not (Fileexists('1004unixdict.txt') or fileexists('unixdict.txt')) then
32936:  if IsInternet then begin
32937:    writeln(dictURL)
32938:    wGetX2(dictUrl,Exepath+'1004unixdict.txt');
32939:    openFile(Exepath+'1004unixdict.txt')
32940:  end; //}
32941:
32942: Func IsASCIIDigit(const Ch: Char):Bool;
32943: begin //Result := Ord(Ch) in [Ord('0')..Ord('9')];
32944:   Result:= (Ord(Ch) >= Ord('0')) And (ord(ch) <= Ord('9'));
32945: end;
32946:
32947: Func RemainingBatteryPercent: Int;
32948: var Stat: {Windows.}TSystemPowerStatus;
32949: begin
32950:   {Windows.}GetSystemPowerStatus(Stat);
32951:   Result:= Stat.BatteryLifePercent;
32952:   if (Result < 0) Or (Result > 100) then
32953:     Result:= -1;
32954: end;
32955: // ************************************************************
32956: // TIdBaseComponent is the base class for all Indy components.
32957: // ************************************************************
32958: type
32959:   TIdBaseComponent = class(TComponent)
```

```
32960:   public
32961:     Func GetVersion:Str;
32962:     property Version:Str read GetVersion;
32963:   published
32964:   end;
32965:
32966:  if IsInternet then
32967:    wGetX('http://www.softwareschule.ch/download/maxbox_starter4.pdf','mywgettestpdf.pdf;
32968:     RegEx tester sr := '123.456.789-00$p'
32969:     writeln(ReplaceRegExpr('\D',sr,'',true))
32970:     >>> 12345678900  //\D is a non-digit, and \W is a non-word character, both should work)
32971:     ref: lambda: a+b --> a.+(b) in preparations
32972:     namedpipe1:= namedpipe as TNamedpipe;
32973:     printF('this is %.18f ',[maxCalc('ln(2)^e')]); this is 0.369248502937272178
32974:     printF('this is Area of r=1  %.18f ',[maxCalc('PI*(1^2)')]);
32975:     printF('this is Area of d=2r %.18f ',[maxCalc('PI/4*(2^2)')]);
32976:     println(ReplaceRegExpr('\d\d{4,12}[},]',
32977:              GetSentimentStream7(URLSentimentAPI2,Items[it].title),' ',false))
32978:     {"probability": {"neg": 0.5180  "neutral": 0.5201  "pos": 0.4819 , "label": "neutral"}
32979:  begin
32980:   initCRCarray2;
32981:   aStringStream := TStringStream.Create(filetoString(exepath+testFile));
32982:   try
32983:    //aStringStream.loadfromfile
32984:     writeln('CRC32_: '+IntToHex(CalculateCRC32X(aStringStream), 8));
32985:   finally
32986:     aStringStream.Free;
32987:   end;
32988:  end;
32989:
32990: Proc TForm1Findwindow_Remote_Notepad;
32991: var wnd: HWND; i: Int; s:Str;
32992: begin
32993:   wnd := FindWindow('notepad', ';
32994:   if wnd <> 0 then begin
32995:     wnd := FindWindowEx(wnd, 0, 'Edit', ';
32996:     // Write Text in Notepad.
32997:     s := 'Hello maXPad 3 ecotronics'+#13#10+'second line:';
32998:     for i := 1 to Length(s) do
32999:       SendMessage(wnd, WM_CHAR, Ord(s[i]), 0);
33000:     // Simulate Return Key.
33001:     PostMessage(wnd, WM_KEYDOWN, VK_RETURN, 0);
33002:     // Simulate Space.
33003:     PostMessage(wnd, WM_KEYDOWN, VK_SPACE, 0);
33004:     SendMessage(wnd, WM_CHAR, 5, 0);
33005:     //send time of F5 command in notepad!
33006:     PostMessage(wnd, WM_KEYDOWN, VK_F5, 0);
33007:   end;
33008: end;
33009:
33010: Proc XPATHTest; var XML: OLEVariant;
33011: begin
33012:   XML:= CreateOleObject('MSXML2.DOMDocument.3.0') ;
33013:   XML.async:= false;
33014:   XML.SetProperty('SelectionLanguage','XPath');
33015:   XML:= unassigned;
33016: end;
33017:
33018: with adoQuery do begin //recordset hack
33019:   cachesize:= 500;
33020:   //commandtext:= 'INSERT INTO Table1 (FirstName, LastName, Phone)'+
33021:                   'VALUES (''MaXML'', ''Box42810'', ''031-3337788'')';
33022:   commandText:= 'SELECT * FROM Table1';
33023:   connectionString:='Provider=MSDASQL.1;Persist Security Info=False;Data Source=mX4base';
33024:   Open;
33025:   Recordset20(recordset).save(Exepath+'recordset6.xml',adPersistXML)
33026:   writeln(GetISODateString(1995,12,24))
33027:   writeln(botostr(IsValidISODateTimeString(GetISODateString(1995,12,12), true)))
33028:   writeln(MicrosoftCodePageToMIMECharset(1252))writeln(GetISOTimeString(12,45,33, true));
33029:   writeln(botostr(IsValidISOTimeString(GetISOTimeString(12,45,33, true), true)))
33030:   IDoc:= LoadXMLDocument3(BOOKSXML); //('.\books.xml');
33031:   INode:= SelectXMLNode(IDoc.getDocumentElement,'/bookstore/book[2]/title');
33032:   writeln(FormatXMLFile2String(BOOKSXML));
33033:   writeln(DataSetToXML(TDataset(adoquery)));
33034:
33035: Proc EnableDEP2;     // DEP
33036: const PROCESS_DEP_ENABLE: DWORD=$00000001;
33037: var SetProcessDEPPolicy: function(dwFlags: DWORD): BOOL; stdcall;
33038: begin  SetProcessDEPPolicy := GetProcAddress(GetModuleHandle(kernel32),'SetProcessDEPPolicy');
33039:   if Assigned(SetProcessDEPPolicy) then begin
33040:     //don't bother checking for errors since we don't need to know if it fails
33041:     SetProcessDEPPolicy(PROCESS_DEP_ENABLE);
33042:   end;
33043: end;
33044:
33045: unit uPSI_UTGraphSearch; chNGE tnode to tnode2
33046:
33047: procedure SIRegister_TNode2(CL: TPSPascalCompiler);
33048: begin
```

```
33049:   //with RegClassS(CL,'tObject', 'TNode2') do
33050:   with CL.AddClassN(CL.FindClass('tObject'),'TNode') do begin
33051:     RegisterProperty('adjacents', 'array of TEdge', iptrw);
33052:     RegisterProperty('nbradjacents', 'integer', iptrw);
33053:     RegisterProperty('index', 'integer', iptrw);
33054:     RegisterProperty('x', 'integer', iptrw);
33055:     RegisterProperty('y', 'integer', iptrw);
33056:     RegisterMethod('Constructor create');
33057:   end;
33058: end;
33059:
33060: TList.count and capacity!
33061: RegisterProperty('Q', 'TList', iptrw);
33062: type  __Pointer; as TObject cause of NIL
33063: FreeAndNilStream (Output); Output : TStream;
33064: FreeAndNilBmp (Output); Output : TBitmap;
33065: sign: About!
33066:
33067: Convert a dfm to string: var abt:Bool;
33068: srlist:= TStringlist.create;
33069: if LoadDFMFile2Strings('C:\maXbox\mX42810\backgroundworker\Demo\Primes\Main.dfm',srlist,abt)= 0 then
       writeln(srlist.text);  srlist.Free;    *) Assert Test: _outd( a1.StandardDeviation );
33070:    Assert(trunc(a1.StandardDeviation)=46, 'must = 46 ');
33071:  isloc:= WMIStart;  iserv:= WMIConnect(isloc,'localhost','','');
33072:  wmi_dom:= 'Win32_NetworkAdapterConfiguration';
33073:
33074: V 4.7.5.90 VI  +  4.7.6.10
33075: Amount of Functions: 20549 Amount of Procedures: 12307
33076: Amount of Constructors: 1937 Amount of Destructors: 14
33077: Totals of Calls: 34807 , Size: 31,974,216 bytes
33078: SHA1: of 4.7.5.90 96DCDE2028125E00B67E42A801721AC513A5EAFC CRC32: BBC3A7E5
33079: ▯▯▯ mX4 executed: 03/11/2021 15:25:47  Runtime: 0:12:33.807 Memload: 39% use
33080: ▯▯▯ 4.7.6.10 III Totals of Calls: 35218
33081: SHA1: of 4.7.6.10 6647E6200E46702944A6B8175A53EDC1ABEF8E14
33082: CRC32: 1A2922DA:  32,221,000 bytes
33083: ▯▯▯ mX4 executed: 12/12/2021 10:03:11  Runtime: 0:12:50.630  Memload: 45% use
33084: ▯▯▯ 4.7.6.10 IV Total of Function Calls: 35351
33085: SHA1: of 4.7.6.10  A9C0D35723F9D2E72E9270207640F938FF58555B
33086: CRC32: 8230EA82 31.3 MB (32,917,832 bytes)
33087: ▯▯▯ 4.7.6.10 VII
33088: Amount of Functions: 20994 Amount of Procedures: 12492
33089: Amount of Constructors: 1984  Amount of Destructors: 14
33090: Totals of Calls:35484 SHA1:of 4.7.6.10 EC20D0A371A57AA078762D6D45F7229A370549A1
33091: CRC32: 82E481A4:  33273160 bytes
33092: ▯▯▯ mX4 executed: 14/03/2022 15:21:38  Runtime: 0:13:41.253  Memload: 42% use
33093: ▯▯▯
33094: 17/03/2022  14:08      194,307 IFSI_WinForm1puzzle.dcu
33095: 17/03/2022  20:30       46,854 uPSC_classes.dcu
33096: 17/03/2022  19:48        5,226 uPSC_std.dcu
33097: 17/03/2022  18:49       53,416 uPSR_classes.dcu
33098: 17/03/2022  20:30       19,528 uPSI_SynEditHighlighter.dcu
33099: 17/03/2022  16:03       11,784 uPSI_SynEditMiscProcs.dcu
33100: 17/03/2022  21:15      173,341 uPSI_TeEngine.dcu
33101: 17/03/2022  21:16       30,854 MathsLib.dcu
33102: 17/03/2022  20:13       77,627 uPSI_neuralvolume.dcu
33103: 17/03/2022  13:10       13,188 uPSI_UcomboV2.dcu
33104: 17/03/2022  13:26       14,813 uPSI_UTGraphSearch.dcu
33105: 17/03/2022  13:22       21,041 UTGraphSearch.dcu
33106:
33107: Release Notes maXbox 4.7.6.10 VIII August 2022 mX476
33108: Amount of Functions: 21179
33109: Amount of Procedures: 12591
33110: Amount of Constructors: 1987
33111: Amount of Destructors: 14
33112: Totals of Calls: 35771
33113:
33114: Release Notes maXbox 4.7.6.20 Oct 2022 mX476
33115: Amount of Functions: 21309
33116: Amount of Procedures: 12640
33117: Amount of Constructors: 1989
33118: Amount of Destructors: 14
33119: Totals of Calls: 35952
33120: SHA1: of 4.7.6.20 C0459A5DD0CCC12CE9F3CDEC82E3E0C920D38D9C
33121: CRC32: 498B62AF:  33690392 bytes
33122: ▯▯▯ mX4 executed: 09/10/2022 20:59:18  Runtime: 0:14:21.810  Memload: 39% use
33123: ▯▯▯ maXbox4.exe now available as a NFT
33124: sha256: 09eb95c4cc3b7fe60772405b170af4705085da89d190213228c65313c8733499
33125: https://lawrencebarsanti.wordpress.com/2009/11/28/introduction-to-pascal-script/
33126:
33127: Total of Function Calls: 35952
33128: SHA1: 4.7.6.20 B2D3DB10079C63D9C4E3C3350844174E259F4FEB
33129: CRC32: 0C3FCCE9 32.2 MB (33,776,920 bytes)
33130: Compilation Timestamp 2022-10-13 18:54:15 UTC
33131: Entry Point 24770844 - Contained Sections 10
33132: ▯▯▯
33133: sha1: b2d3db10079c63d9c4e3c3350844174e259f4feb
33134: sha256: 5d229931d7b91a8e1d79ee2b1f19279a0dd2685c7c99974b0da9df749347fc58
33135:
33136: Amount of Functions: 21314
```

```
33137: Amount of Procedures: 12647
33138: Amount of Constructors: 1981
33139: Amount of Destructors: 14
33140: Totals of Calls: 35956
33141: SHA1: of 4.7.6.20 545D9B5F888B2D147968BC88F40358ACBA898F4A
33142: CRC32: B42723A6:  33788184 bytes
33143: ▯▯▯ mX4 executed: 15/10/2022 20:54:58  Runtime: 0:14:27.700  Memload: 38% use
33144: ▯▯▯
33145: sha1: 545d9b5f888b2d147968bc88f40358acba898f4a
33146: sha256: c6e8c0828c61b0a05cb26ebdafe6985983eb8f6b4d92fb99ad2fa62d2ba50bed
33147: Compilation Timestamp 2022-10-15 15:57:18 UTC
33148: Entry Point 24783132 - Contained Sections 10
33149:
33150: https://www.hybrid-analysis.com/
33151:
33152: Suspicious Indicators
33153: General
33154: Found a potential E-Mail address in binary/memory
33155: Network Related
33156: Found potential IP address in binary/memory
33157: Remote Access Related
33158: Contains indicators of bot communication commands
33159: Contains references to WMI/WMIC
33160: Reads terminal service related keys (often RDP related)
33161: Spyware/Information Retrieval
33162: Touches files in program files directory
33163: Unusual Characteristics
33164: Input file contains API references not part of its Import Address Table (IAT)
33165: Installs hooks/patches the running process
33166:
33167:
33168: ************************************************************
33169:  unit List asm internal end units  all unitlist unitslist allunits
33170: ************************************************************
33171: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
33172: 02 unit SIRegister_IdStrings                   //Indy Sockets
33173: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
33174: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
33175: 05 unit IFSI_WinForm1puzzle;                   //maXbox
33176: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
33177: 07 unit RegisterDateTimeLibrary_R(exec);       //Delphi
33178: 08 unit RIRegister_MathMax_Routines(exec);     //Jedi & Delphi
33179: 09 unit RIRegister_IdGlobal_Routines(exec);    //Indy Sockets
33180: 10 unit RIRegister_SysUtils_Routines(Exec);    //Delphi
33181: 11 unit uPSI_IdTCPConnection;                  //Indy some functions
33182: 12 unit uPSCompiler.pas;                       //PS kernel functions
33183: 13 unit uPSI_DBCommon;                         //DB Common_Routines and Types
33184: 14 unit uPSI_Printers.pas                      //Delphi VCL
33185: 15 unit uPSI_MPlayer.pas                       //Delphi VCL
33186: 16 unit uPSC_comobj;                           //COM Functions
33187: 17 unit uPSI_Clipbrd;                          //Delphi VCL
33188: 18 unit Filectrl in IFSI_SysUtils_max;         //VCL Runtime
33189: 19 unit uPSI_SqlExpr;                          //DBX3
33190: 20 unit uPSI_ADODB;                            //ADODB
33191: 21 unit uPSI_StrHlpr;                          //String Helper Routines
33192: 22 unit uPSI_DateUtils;                        //Expansion to DateTimeLib
33193: 23 unit uPSI_FileUtils;                        //Expansion to Sys/File Utils
33194: 24 unit JUtils / gsUtils;                      //Jedi / Metabase
33195: 25 unit JvFunctions_max;                       //Jedi Functions
33196: 26 unit HTTPParser;                            //Delphi VCL
33197: 27 unit HTTPUtil;                              //Delphi VCL
33198: 28 unit uPSI_XMLUtil;                          //Delphi VCL
33199: 29 unit uPSI_SOAPHTTPClient;                   //Delphi VCL SOAP WebService V3.5
33200: 30 unit uPSI_Contnrs;                          //Delphi RTL Container of Classes
33201: 31 unit uPSI_MaskUtils;                        //RTL Edit and Mask functions
33202: 32 unit uPSI_MyBigInt;                         //big Int class with Math
33203: 33 unit uPSI_ConvUtils;                        //Delphi VCL Conversions engine
33204: 34 unit Types_Variants;                        //Delphi\Win32\rtl\sys
33205: 35 unit uPSI_IdHashSHA1;                       //Indy Crypto Lib
33206: 36 unit uPSI_IdHashMessageDigest               //Indy Crypto;
33207: 37 unit uPSI_IdASN1Util;                       //Indy ASN1Utility Routines;
33208: 38 unit uPSI_IdLogFile;                        //Indy Logger from LogBase
33209: 39 unit uPSI_IdIcmpClient;                     //Indy Ping ICMP
33210: 40 unit uPSI_IdHashMessageDigest_max           //Indy Crypto &OpenSSL;
33211: 41 unit uPSI_FileCtrl;                         //Delphi RTL
33212: 42 unit uPSI_Outline;                          //Delphi VCL
33213: 43 unit uPSI_ScktComp;                         //Delphi RTL
33214: 44 unit uPSI_Calendar;                         //Delphi VCL
33215: 45 unit uPSI_VListView                         //VListView;
33216: 46 unit uPSI_DBGrids;                          //Delphi VCL
33217: 47 unit uPSI_DBCtrls;                          //Delphi VCL
33218: 48 unit ide_debugoutput;                       //maXbox
33219: 49 unit uPSI_ComCtrls;                         //Delphi VCL
33220: 50 unit uPSC_stdctrls+;                        //Delphi VCL
33221: 51 unit uPSI_Dialogs;                          //Delphi VCL
33222: 52 unit uPSI_StdConvs;                         //Delphi RTL
33223: 53 unit uPSI_DBClient;                         //Delphi RTL
33224: 54 unit uPSI_DBPlatform;                       //Delphi RTL
33225: 55 unit uPSI_Provider;                         //Delphi RTL
```

```
33226: 56 unit uPSI_FMTBcd;                          //Delphi RTL
33227: 57 unit uPSI_DBCGrids;                        //Delphi VCL
33228: 58 unit uPSI_CDSUtil;                         //MIDAS
33229: 59 unit uPSI_VarHlpr;                         //Delphi RTL
33230: 60 unit uPSI_ExtDlgs;                         //Delphi VCL
33231: 61 unit sdpStopwatch;                         //maXbox
33232: 62 unit uPSI_JclStatistics;                   //JCL
33233: 63 unit uPSI_JclLogic;                        //JCL
33234: 64 unit uPSI_JclMiscel;                       //JCL
33235: 65 unit uPSI_JclMath_max;                     //JCL RTL
33236: 66 unit uPSI_uTPLb_StreamUtils;               //LockBox 3
33237: 67 unit uPSI_MathUtils;                       //BCB
33238: 68 unit uPSI_JclMultimedia;                   //JCL
33239: 69 unit uPSI_WideStrUtils;                    //Delphi API/RTL
33240: 70 unit uPSI_GraphUtil;                       //Delphi RTL
33241: 71 unit uPSI_TypeTrans;                       //Delphi RTL
33242: 72 unit uPSI_HTTPApp;                         //Delphi VCL
33243: 73 unit uPSI_DBWeb;                           //Delphi VCL
33244: 74 unit uPSI_DBBdeWeb;                        //Delphi VCL
33245: 75 unit uPSI_DBXpressWeb;                     //Delphi VCL
33246: 76 unit uPSI_ShadowWnd;                       //Delphi VCL
33247: 77 unit uPSI_ToolWin;                         //Delphi VCL
33248: 78 unit uPSI_Tabs;                            //Delphi VCL
33249: 79 unit uPSI_JclGraphUtils;                   //JCL
33250: 80 unit uPSI_JclCounter;                      //JCL
33251: 81 unit uPSI_JclSysInfo;                      //JCL
33252: 82 unit uPSI_JclSecurity;                     //JCL
33253: 83 unit uPSI_JclFileUtils;                    //JCL
33254: 84 unit uPSI_IdUserAccounts;                  //Indy
33255: 85 unit uPSI_IdAuthentication;                //Indy
33256: 86 unit uPSI_uTPLb_AES;                       //LockBox 3
33257: 87 unit uPSI_IdHashSHA1;                      //LockBox 3
33258: 88 unit uTPLb_BlockCipher;                    //LockBox 3
33259: 89 unit uPSI_ValEdit.pas;                     //Delphi VCL
33260: 90 unit uPSI_JvVCLUtils;                      //JCL
33261: 91 unit uPSI_JvDBUtil;                        //JCL
33262: 92 unit uPSI_JvDBUtils;                       //JCL
33263: 93 unit uPSI_JvAppUtils;                      //JCL
33264: 94 unit uPSI_JvCtrlUtils;                     //JCL
33265: 95 unit uPSI_JvFormToHtml;                    //JCL
33266: 96 unit uPSI_JvParsing;                       //JCL
33267: 97 unit uPSI_SerDlgs;                         //Toolbox
33268: 98 unit uPSI_Serial;                          //Toolbox
33269: 99 unit uPSI_JvComponent;                     //JCL
33270: 100 unit uPSI_JvCalc;                         //JCL
33271: 101 unit uPSI_JvBdeUtils;                     //JCL
33272: 102 unit uPSI_JvDateUtil;                     //JCL
33273: 103 unit uPSI_JvGenetic;                      //JCL
33274: 104 unit uPSI_JclBase;                        //JCL
33275: 105 unit uPSI_JvUtils;                        //JCL
33276: 106 unit uPSI_JvStrUtil;                      //JCL
33277: 107 unit uPSI_JvStrUtils;                     //JCL
33278: 108 unit uPSI_JvFileUtil;                     //JCL
33279: 109 unit uPSI_JvMemoryInfos;                  //JCL
33280: 110 unit uPSI_JvComputerInfo;                 //JCL
33281: 111 unit uPSI_JvgCommClasses;                 //JCL
33282: 112 unit uPSI_JvgLogics;                      //JCL
33283: 113 unit uPSI_JvLED;                          //JCL
33284: 114 unit uPSI_JvTurtle;                       //JCL
33285: 115 unit uPSI_SortThds; unit uPSI_ThSort;     //maXbox
33286: 116 unit uPSI_JvgUtils;                       //JCL
33287: 117 unit uPSI_JvExprParser;                   //JCL
33288: 118 unit uPSI_HexDump;                        //Borland
33289: 119 unit uPSI_DBLogDlg;                       //VCL
33290: 120 unit uPSI_SqlTimSt;                       //RTL
33291: 121 unit uPSI_JvHtmlParser;                   //JCL
33292: 122 unit uPSI_JvgXMLSerializer;               //JCL
33293: 123 unit uPSI_JvJCLUtils;                     //JCL
33294: 124 unit uPSI_JvStrings;                      //JCL
33295: 125 unit uPSI_uTPLb_IntUtils;                 //TurboPower
33296: 126 unit uPSI_uTPLb_HugeCardinal;             //TurboPower
33297: 127 unit uPSI_uTPLb_HugeCardinalUtils;        //TurboPower
33298: 128 unit uPSI_SynRegExpr;                     //SynEdit
33299: 129 unit uPSI_StUtils;                        //SysTools4
33300: 130 unit uPSI_StToHTML;                       //SysTools4
33301: 131 unit uPSI_StStrms;                        //SysTools4
33302: 132 unit uPSI_StFIN;                          //SysTools4
33303: 133 unit uPSI_StAstroP;                       //SysTools4
33304: 134 unit uPSI_StStat;                         //SysTools4
33305: 135 unit uPSI_StNetCon;                       //SysTools4
33306: 136 unit uPSI_StDecMth;                       //SysTools4
33307: 137 unit uPSI_StOStr;                         //SysTools4
33308: 138 unit uPSI_StPtrns;                        //SysTools4
33309: 139 unit uPSI_StNetMsg;                       //SysTools4
33310: 140 unit uPSI_StMath;                         //SysTools4
33311: 141 unit uPSI_StExpEng;                       //SysTools4
33312: 142 unit uPSI_StCRC;                          //SysTools4
33313: 143 unit uPSI_StExport,                       //SysTools4
33314: 144 unit uPSI_StExpLog,                       //SysTools4
```

```
33315: 145 unit uPSI_ActnList;                          //Delphi VCL
33316: 146 unit uPSI_jpeg;                              //Borland
33317: 147 unit uPSI_StRandom;                          //SysTools4
33318: 148 unit uPSI_StDict;                            //SysTools4
33319: 149 unit uPSI_StBCD;                             //SysTools4
33320: 150 unit uPSI_StTxtDat;                          //SysTools4
33321: 151 unit uPSI_StRegEx;                           //SysTools4
33322: 152 unit uPSI_IMouse;                            //VCL
33323: 153 unit uPSI_SyncObjs;                          //VCL
33324: 154 unit uPSI_AsyncCalls;                        //Hausladen
33325: 155 unit uPSI_ParallelJobs;                      //Saraiva
33326: 156 unit uPSI_Variants;                          //VCL
33327: 157 unit uPSI_VarCmplx;                          //VCL Wolfram
33328: 158 unit uPSI_DTDSchema;                         //VCL
33329: 159 unit uPSI_ShLwApi;                           //Brakel
33330: 160 unit uPSI_IBUtils;                           //VCL
33331: 161 unit uPSI_CheckLst;                          //VCL
33332: 162 unit uPSI_JvSimpleXml;                       //JCL
33333: 163 unit uPSI_JclSimpleXml;                      //JCL
33334: 164 unit uPSI_JvXmlDatabase;                     //JCL
33335: 165 unit uPSI_JvMaxPixel;                        //JCL
33336: 166 unit uPSI_JvItemsSearchs;                    //JCL
33337: 167 unit uPSI_StExpEng2;                         //SysTools4
33338: 168 unit uPSI_StGenLog;                          //SysTools4
33339: 169 unit uPSI_JvLogFile;                         //Jcl
33340: 170 unit uPSI_CPort;                             //ComPort Lib v4.11
33341: 171 unit uPSI_CPortCtl;                          //ComPort
33342: 172 unit uPSI_CPortEsc;                          //ComPort
33343: 173 unit BarCodeScaner;                          //ComPort
33344: 174 unit uPSI_JvGraph;                           //JCL
33345: 175 unit uPSI_JvComCtrls;                        //JCL
33346: 176 unit uPSI_GUITesting;                        //D Unit
33347: 177 unit uPSI_JvFindFiles;                       //JCL
33348: 178 unit uPSI_StSystem;                          //SysTools4
33349: 179 unit uPSI_JvKeyboardStates;                  //JCL
33350: 180 unit uPSI_JvMail;                            //JCL
33351: 181 unit uPSI_JclConsole;                        //JCL
33352: 182 unit uPSI_JclLANMan;                         //JCL
33353: 183 unit uPSI_IdCustomHTTPServer;                //Indy
33354: 184 unit IdHTTPServer                            //Indy
33355: 185 unit uPSI_IdTCPServer;                       //Indy
33356: 186 unit uPSI_IdSocketHandle;                    //Indy
33357: 187 unit uPSI_IdIOHandlerSocket;                 //Indy
33358: 188 unit IdIOHandler;                            //Indy
33359: 189 unit uPSI_cutils;                            //Bloodshed
33360: 190 unit uPSI_BoldUtils;                         //boldsoft
33361: 191 unit uPSI_IdSimpleServer;                    //Indy
33362: 192 unit uPSI_IdSSLOpenSSL;                      //Indy
33363: 193 unit uPSI_IdMultipartFormData;               //Indy
33364: 194 unit uPSI_SynURIOpener;                      //SynEdit
33365: 195 unit uPSI_PerlRegEx;                         //PCRE
33366: 196 unit uPSI_IdHeaderList;                      //Indy
33367: 197 unit uPSI_StFirst;                           //SysTools4
33368: 198 unit uPSI_JvCtrls;                           //JCL
33369: 199 unit uPSI_IdTrivialFTPBase;                  //Indy
33370: 200 unit uPSI_IdTrivialFTP;                      //Indy
33371: 201 unit uPSI_IdUDPBase;                         //Indy
33372: 202 unit uPSI_IdUDPClient;                       //Indy
33373: 203 unit uPSI_utypes;                            //for DMath.DLL
33374: 204 unit uPSI_ShellAPI;                          //Borland
33375: 205 unit uPSI_IdRemoteCMDClient;                 //Indy
33376: 206 unit uPSI_IdRemoteCMDServer;                 //Indy
33377: 207 unit IdRexecServer;                          //Indy
33378: 208 unit IdRexec; (unit uPSI_IdRexec;)           //Indy
33379: 209 unit IdUDPServer;                            //Indy
33380: 210 unit IdTimeUDPServer;                        //Indy
33381: 211 unit IdTimeServer;                           //Indy
33382: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)     //Indy
33383: 213 unit uPSI_IdIPWatch;                         //Indy
33384: 214 unit uPSI_IdIrcServer;                       //Indy
33385: 215 unit uPSI_IdMessageCollection;               //Indy
33386: 216 unit uPSI_cPEM;                              //Fundamentals 4
33387: 217 unit uPSI_cFundamentUtils;                   //Fundamentals 4
33388: 218 unit uPSI_uwinplot;                          //DMath
33389: 219 unit uPSI_xrtl_util_CPUUtils;                //ExtentedRTL
33390: 220 unit uPSI_GR32_System;                       //Graphics32
33391: 221 unit uPSI_cFileUtils;                        //Fundamentals 4
33392: 222 unit uPSI_cDateTime; (timemachine)           //Fundamentals 4
33393: 223 unit uPSI_cTimers; (high precision timer)    //Fundamentals 4
33394: 224 unit uPSI_cRandom;                           //Fundamentals 4
33395: 225 unit uPSI_ueval;                             //DMath
33396: 226 unit uPSI_xrtl_net_URIUtils;                 //ExtendedRTL
33397: 227 unit xrtl_net_URIUtils;                      //ExtendedRTL
33398: 228 unit uPSI_ufft;  (FFT)                       //DMath
33399: 229 unit uPSI_DBXChannel;                        //Delphi
33400: 230 unit uPSI_DBXIndyChannel;                    //Delphi Indy
33401: 231 unit uPSI_xrtl_util_COMCat;                  //ExtendedRTL
33402: 232 unit uPSI_xrtl_util_StrUtils;                //ExtendedRTL
33403: 233 unit uPSI_xrtl_util_VariantUtils;            //ExtendedRTL
```

```
33404: 234 unit uPSI_xrtl_util_FileUtils;          //ExtendedRTL
33405: 235 unit xrtl_util_Compat;                  //ExtendedRTL
33406: 236 unit uPSI_OleAuto;                      //Borland
33407: 237 unit uPSI_xrtl_util_COMUtils;           //ExtendedRTL
33408: 238 unit uPSI_CmAdmCtl;                     //Borland
33409: 239 unit uPSI_ValEdit2;                     //VCL
33410: 240 unit uPSI_GR32;  //Graphics32           //Graphics32
33411: 241 unit uPSI_GR32_Image;                   //Graphics32
33412: 242 unit uPSI_xrtl_util_TimeUtils;          //ExtendedRTL
33413: 243 unit uPSI_xrtl_util_TimeZone;           //ExtendedRTL
33414: 244 unit uPSI_xrtl_util_TimeStamp;          //ExtendedRTL
33415: 245 unit uPSI_xrtl_util_Map;                //ExtendedRTL
33416: 246 unit uPSI_xrtl_util_Set;                //ExtendedRTL
33417: 247 unit uPSI_CPortMonitor;                 //ComPort
33418: 248 unit uPSI_StIniStm;                     //SysTools4
33419: 249 unit uPSI_GR32_ExtImage;                //Graphics32
33420: 250 unit uPSI_GR32_OrdinalMaps;             //Graphics32
33421: 251 unit uPSI_GR32_Rasterizers;             //Graphics32
33422: 252 unit uPSI_xrtl_util_Exception;          //ExtendedRTL
33423: 253 unit uPSI_xrtl_util_Value;              //ExtendedRTL
33424: 254 unit uPSI_xrtl_util_Compare;            //ExtendedRTL
33425: 255 unit uPSI_FlatSB;                       //VCL
33426: 256 unit uPSI_JvAnalogClock;                //JCL
33427: 257 unit uPSI_JvAlarms;                     //JCL
33428: 258 unit uPSI_JvSQLS;                       //JCL
33429: 259 unit uPSI_JvDBSecur;                    //JCL
33430: 260 unit uPSI_JvDBQBE;                      //JCL
33431: 261 unit uPSI_JvStarfield;                  //JCL
33432: 262 unit uPSI_JVCLMiscal;                   //JCL
33433: 263 unit uPSI_JvProfiler32;                 //JCL
33434: 264 unit uPSI_JvDirectories,                //JCL
33435: 265 unit uPSI_JclSchedule,                  //JCL
33436: 266 unit uPSI_JclSvcCtrl,                   //JCL
33437: 267 unit uPSI_JvSoundControl,               //JCL
33438: 268 unit uPSI_JvBDESQLScript,               //JCL
33439: 269 unit uPSI_JvgDigits,                    //JCL>
33440: 270 unit uPSI_ImgList;                      //TCustomImageList
33441: 271 unit uPSI_JclMIDI;                      //JCL>
33442: 272 unit uPSI_JclWinMidi;                   //JCL>
33443: 273 unit uPSI_JclNTFS;                      //JCL>
33444: 274 unit uPSI_JclAppInst;                   //JCL>
33445: 275 unit uPSI_JvRle;                        //JCL>
33446: 276 unit uPSI_JvRas32;                      //JCL>
33447: 277 unit uPSI_JvImageDrawThread,            //JCL>
33448: 278 unit uPSI_JvImageWindow,                //JCL>
33449: 279 unit uPSI_JvTransparentForm;            //JCL>
33450: 280 unit uPSI_JvWinDialogs;                 //JCL>
33451: 281 unit uPSI_JvSimLogic,                   //JCL>
33452: 282 unit uPSI_JvSimIndicator,               //JCL>
33453: 283 unit uPSI_JvSimPID,                     //JCL>
33454: 284 unit uPSI_JvSimPIDLinker,               //JCL>
33455: 285 unit uPSI_IdRFCReply;                   //Indy
33456: 286 unit uPSI_IdIdent;                      //Indy
33457: 287 unit uPSI_IdIdentServer;                //Indy
33458: 288 unit uPSI_JvPatchFile;                  //JCL
33459: 289 unit uPSI_StNetPfm;                     //SysTools4
33460: 290 unit uPSI_StNet;                        //SysTools4
33461: 291 unit uPSI_JclPeImage;                   //JCL
33462: 292 unit uPSI_JclPrint;                     //JCL
33463: 293 unit uPSI_JclMime;                      //JCL
33464: 294 unit uPSI_JvRichEdit;                   //JCL
33465: 295 unit uPSI_JvDBRichEd;                   //JCL
33466: 296 unit uPSI_JvDice;                       //JCL
33467: 297 unit uPSI_JvFloatEdit;                  //JCL 3.9.8
33468: 298 unit uPSI_JvDirFrm;                     //JCL
33469: 299 unit uPSI_JvDualList;                   //JCL
33470: 300 unit uPSI_JvSwitch;                     ////JCL
33471: 301 unit uPSI_JvTimerLst;                   ////JCL
33472: 302 unit uPSI_JvMemTable;                   //JCL
33473: 303 unit uPSI_JvObjStr;                     //JCL
33474: 304 unit uPSI_StLArr;                       //SysTools4
33475: 305 unit uPSI_StWmDCpy;                     //SysTools4
33476: 306 unit uPSI_StText;                       //SysTools4
33477: 307 unit uPSI_StNTLog;                      //SysTools4
33478: 308 unit uPSI_xrtl_math_Int;                //ExtendedRTL
33479: 309 unit uPSI_JvImagPrvw;                   //JCL
33480: 310 unit uPSI_JvFormPatch;                  //JCL
33481: 311 unit uPSI_JvPicClip;                    //JCL
33482: 312 unit uPSI_JvDataConv;                   //JCL
33483: 313 unit uPSI_JvCpuUsage;                   //JCL
33484: 314 unit uPSI_JclUnitConv_mX2;              //JCL
33485: 315 unit JvDualListForm;                    //JCL
33486: 316 unit uPSI_JvCpuUsage2;                  //JCL
33487: 317 unit uPSI_JvParserForm;                 //JCL
33488: 318 unit uPSI_JvJanTreeView;                //JCL
33489: 319 unit uPSI_JvTransLED;                   //JCL
33490: 320 unit uPSI_JvPlaylist;                   //JCL
33491: 321 unit uPSI_JvFormAutoSize;               //JCL
33492: 322 unit uPSI_JvYearGridEditForm;           //JCL
```

```
33493: 323 unit uPSI_JvMarkupCommon;                    //JCL
33494: 324 unit uPSI_JvChart;                           //JCL
33495: 325 unit uPSI_JvXPCore;                          //JCL
33496: 326 unit uPSI_JvXPCoreUtils;                     //JCL
33497: 327 unit uPSI_StatsClasses;                      //mX4  a1: TStatArray;
33498: 328 unit uPSI_ExtCtrls2;                         //VCL
33499: 329 unit uPSI_JvUrlGrabbers;                     //JCL
33500: 330 unit uPSI_JvXmlTree;                         //JCL
33501: 331 unit uPSI_JvWavePlayer;                      //JCL
33502: 332 unit uPSI_JvUnicodeCanvas;                   //JCL
33503: 333 unit uPSI_JvTFUtils;                         //JCL
33504: 334 unit uPSI_IdServerIOHandler;                 //Indy
33505: 335 unit uPSI_IdServerIOHandlerSocket;           //Indy
33506: 336 unit uPSI_IdMessageCoder;                    //Indy
33507: 337 unit uPSI_IdMessageCoderMIME;                //Indy
33508: 338 unit uPSI_IdMIMETypes;                       //Indy
33509: 339 unit uPSI_JvConverter;                       //JCL
33510: 340 unit uPSI_JvCsvParse;                        //JCL
33511: 341 unit uPSI_umath;  unit uPSI_ugamma;          //DMath
33512: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport)   //JCL
33513: 343 unit uPSI_JvDBGridExport;                    //JCL
33514: 344 unit uPSI_JvgExport;                         //JCL
33515: 345 unit uPSI_JvSerialMaker;                     //JCL
33516: 346 unit uPSI_JvWin32;                           //JCL
33517: 347 unit uPSI_JvPaintFX;                         //JCL
33518: 348 unit uPSI_JvOracleDataSet; (beta)            //JCL
33519: 349 unit uPSI_JvValidators; (preview)            //JCL
33520: 350 unit uPSI_JvNTEventLog;                      //JCL
33521: 351 unit uPSI_ShellZipTool;                      //mX4
33522: 352 unit uPSI_JvJoystick;                        //JCL
33523: 353 unit uPSI_JvMailSlots;                       //JCL
33524: 354 unit uPSI_JclComplex;                        //JCL
33525: 355 unit uPSI_SynPdf;                            //Synopse
33526: 356 unit uPSI_Registry;                          //VCL
33527: 357 unit uPSI_TlHelp32;                          //VCL
33528: 358 unit uPSI_JclRegistry;                       //JCL
33529: 359 unit uPSI_JvAirBrush;                        //JCL
33530: 360 unit uPSI_mORMotReport;                      //Synopse
33531: 361 unit uPSI_JclLocales;                        //JCL
33532: 362 unit uPSI_SynEdit;                           //SynEdit
33533: 363 unit uPSI_SynEditTypes;                      //SynEdit
33534: 364 unit uPSI_SynMacroRecorder;                  //SynEdit
33535: 365 unit uPSI_LongIntList;                       //SynEdit
33536: 366 unit uPSI_devcutils;                         //DevC
33537: 367 unit uPSI_SynEditMiscClasses;                //SynEdit
33538: 368 unit uPSI_SynEditRegexSearch;                //SynEdit
33539: 369 unit uPSI_SynEditHighlighter;                //SynEdit
33540: 370 unit uPSI_SynHighlighterPas;                 //SynEdit
33541: 371 unit uPSI_JvSearchFiles;                     //JCL
33542: 372 unit uPSI_SynHighlighterAny;                 //Lazarus
33543: 373 unit uPSI_SynEditKeyCmds;                    //SynEdit
33544: 374 unit uPSI_SynEditMiscProcs,                  //SynEdit
33545: 375 unit uPSI_SynEditKbdHandler                  //SynEdit
33546: 376 unit uPSI_JvAppInst,                         //JCL
33547: 377 unit uPSI_JvAppEvent;                        //JCL
33548: 378 unit uPSI_JvAppCommand;                      //JCL
33549: 379 unit uPSI_JvAnimTitle;                       //JCL
33550: 380 unit uPSI_JvAnimatedImage;                   //JCL
33551: 381 unit uPSI_SynEditExport;                     //SynEdit
33552: 382 unit uPSI_SynExportHTML;                     //SynEdit
33553: 383 unit uPSI_SynExportRTF;                      //SynEdit
33554: 384 unit uPSI_SynEditSearch;                     //SynEdit
33555: 385 unit uPSI_fMain_back                         //maXbox;
33556: 386 unit uPSI_JvZoom;                            //JCL
33557: 387 unit uPSI_PMrand;                            //PM
33558: 388 unit uPSI_JvSticker;                         //JCL
33559: 389 unit uPSI_XmlVerySimple;                     //mX4
33560: 390 unit uPSI_Services;                          //ExtPascal
33561: 391 unit uPSI_ExtPascalUtils;                    //ExtPascal
33562: 392 unit uPSI_SocketsDelphi;                     //ExtPascal
33563: 393 unit uPSI_StBarC;                            //SysTools
33564: 394 unit uPSI_StDbBarC;                          //SysTools
33565: 395 unit uPSI_StBarPN;                           //SysTools
33566: 396 unit uPSI_StDbPNBC;                          //SysTools
33567: 397 unit uPSI_StDb2DBC;                          //SysTools
33568: 398 unit uPSI_StMoney;                           //SysTools
33569: 399 unit uPSI_JvForth;                           //JCL
33570: 400 unit uPSI_RestRequest;                       //mX4
33571: 401 unit uPSI_HttpRESTConnectionIndy;            //mX4
33572: 402 unit uPSI_JvXmlDatabase; //update            //JCL
33573: 403 unit uPSI_StAstro;                           //SysTools
33574: 404 unit uPSI_StSort;                            //SysTools
33575: 405 unit uPSI_StDate;                            //SysTools
33576: 406 unit uPSI_StDateSt;                          //SysTools
33577: 407 unit uPSI_StBase;                            //SysTools
33578: 408 unit uPSI_StVInfo;                           //SysTools
33579: 409 unit uPSI_JvBrowseFolder;                    //JCL
33580: 410 unit uPSI_JvBoxProcs;                        //JCL
33581: 411 unit uPSI_urandom; (unit uranuvag;)          //DMath
```

```
33582: 412 unit uPSI_usimann; (unit ugenalg;)          //DMath
33583: 413 unit uPSI_JvHighlighter;                    //JCL
33584: 414 unit uPSI_Diff;                             //mX4
33585: 415 unit uPSI_SpringWinAPI;                     //DSpring
33586: 416 unit uPSI_StBits;                           //SysTools
33587: 417 unit uPSI_TomDBQue;                         //mX4
33588: 418 unit uPSI_MultilangTranslator;              //mX4
33589: 419 unit uPSI_HyperLabel;                       //mX4
33590: 420 unit uPSI_Starter;                          //mX4
33591: 421 unit uPSI_FileAssocs;                       //devC
33592: 422 unit uPSI_devFileMonitorX;                  //devC
33593: 423 unit uPSI_devrun;                           //devC
33594: 424 unit uPSI_devExec;                          //devC
33595: 425 unit uPSI_oysUtils;                         //devC
33596: 426 unit uPSI_DosCommand;                       //devC
33597: 427 unit uPSI_CppTokenizer;                     //devC
33598: 428 unit uPSI_JvHLParser;                       //devC
33599: 429 unit uPSI_JclMapi;                          //JCL
33600: 430 unit uPSI_JclShell;                         //JCL
33601: 431 unit uPSI_JclCOM;                           //JCL
33602: 432 unit uPSI_GR32_Math;                        //Graphics32
33603: 433 unit uPSI_GR32_LowLevel;                    //Graphics32
33604: 434 unit uPSI_SimpleHl;                         //mX4
33605: 435 unit uPSI_GR32_Filters,                     //Graphics32
33606: 436 unit uPSI_GR32_VectorMaps;                  //Graphics32
33607: 437 unit uPSI_cXMLFunctions;                    //Fundamentals 4
33608: 438 unit uPSI_JvTimer;                          //JCL
33609: 439 unit uPSI_cHTTPUtils;                       //Fundamentals 4
33610: 440 unit uPSI_cTLSUtils;                        //Fundamentals 4
33611: 441 unit uPSI_JclGraphics;                      //JCL
33612: 442 unit uPSI_JclSynch;                         //JCL
33613: 443 unit uPSI_IdTelnet;                         //Indy
33614: 444 unit uPSI_IdTelnetServer,                   //Indy
33615: 445 unit uPSI_IdEcho,                           //Indy
33616: 446 unit uPSI_IdEchoServer,                     //Indy
33617: 447 unit uPSI_IdEchoUDP,                        //Indy
33618: 448 unit uPSI_IdEchoUDPServer,                  //Indy
33619: 449 unit uPSI_IdSocks,                          //Indy
33620: 450 unit uPSI_IdAntiFreezeBase;                 //Indy
33621: 451 unit uPSI_IdHostnameServer;                 //Indy
33622: 452 unit uPSI_IdTunnelCommon,                   //Indy
33623: 453 unit uPSI_IdTunnelMaster,                   //Indy
33624: 454 unit uPSI_IdTunnelSlave,                    //Indy
33625: 455 unit uPSI_IdRSH,                            //Indy
33626: 456 unit uPSI_IdRSHServer,                      //Indy
33627: 457 unit uPSI_Spring_Cryptography_Utils;        //Spring4Delphi
33628: 458 unit uPSI_MapReader,                        //devC
33629: 459 unit uPSI_LibTar,                           //devC
33630: 460 unit uPSI_IdStack;                          //Indy
33631: 461 unit uPSI_IdBlockCipherIntercept;           //Indy
33632: 462 unit uPSI_IdChargenServer;                  //Indy
33633: 463 unit uPSI_IdFTPServer,                      //Indy
33634: 464 unit uPSI_IdException,                      //Indy
33635: 465 unit uPSI_utexplot;                         //DMath
33636: 466 unit uPSI_uwinstr;                          //DMath
33637: 467 unit uPSI_VarRecUtils;                      //devC
33638: 468 unit uPSI_JvStringListToHtml,               //JCL
33639: 469 unit uPSI_JvStringHolder,                   //JCL
33640: 470 unit uPSI_IdCoder;                          //Indy
33641: 471 unit uPSI_SynHighlighterDfm;                //Synedit
33642: 472 unit uHighlighterProcs; in 471              //Synedit
33643: 473 unit uPSI_LazFileUtils,                     //LCL
33644: 474 unit uPSI_IDECmdLine;                       //LCL
33645: 475 unit uPSI_lazMasks;                         //LCL
33646: 476 unit uPSI_ip_misc;                          //mX4
33647: 477 unit uPSI_Barcode;                          //LCL
33648: 478 unit uPSI_SimpleXML;                        //LCL
33649: 479 unit uPSI_JclIniFiles;                      //JCL
33650: 480 unit uPSI_D2XXUnit;  {$X-}                  //FTDI
33651: 481 unit uPSI_JclDateTime;                      //JCL
33652: 482 unit uPSI_JclEDI;                           //JCL
33653: 483 unit uPSI_JclMiscel2;                       //JCL
33654: 484 unit uPSI_JclValidation;                    //JCL
33655: 485 unit uPSI_JclAnsistrs; {-PString}        //JCL
33656: 486 unit uPSI_SynEditMiscProcs2;                //Synedit
33657: 487 unit uPSI_JclStreams;                       //JCL
33658: 488 unit uPSI_QRCode;                           //mX4
33659: 489 unit uPSI_BlockSocket;                      //ExtPascal
33660: 490 unit uPSI_Masks,Utils                       //VCL
33661: 491 unit uPSI_synautil;                         //Synapse!
33662: 492 unit uPSI_JclMath_Class;                    //JCL RTL
33663: 493 unit ugamdist; //Gamma Func             //DMath
33664: 494 unit uibeta, ucorrel; //IBeta              //DMath
33665: 495 unit uPSI_SRMgr;                            //mX4
33666: 496 unit uPSI_HotLog;                           //mX4
33667: 497 unit uPSI_DebugBox;                         //mX4
33668: 498 unit uPSI_ustrings;                         //DMath
33669: 499 unit uPSI_uregtest;                         //DMath
33670: 500 unit uPSI_usimplex;                         //DMath
```

```
33671: 501 unit uPSI_uhyper;                          //DMath
33672: 502 unit uPSI_IdHL7;                           //Indy
33673: 503 unit uPSI_IdIPMCastBase,                   //Indy
33674: 504 unit uPSI_IdIPMCastServer;                 //Indy
33675: 505 unit uPSI_IdIPMCastClient;                 //Indy
33676: 506 unit uPSI_unlfit; //nlregression           //DMath
33677: 507 unit uPSI_IdRawHeaders;                    //Indy
33678: 508 unit uPSI_IdRawClient;                     //Indy
33679: 509 unit uPSI_IdRawFunctions;                  //Indy
33680: 510 unit uPSI_IdTCPStream;                     //Indy
33681: 511 unit uPSI_IdSNPP;                          //Indy
33682: 512 unit uPSI_St2DBarC;                        //SysTools
33683: 513 unit uPSI_ImageWin;  //FTL                //VCL
33684: 514 unit uPSI_CustomDrawTreeView; //FTL       //VCL
33685: 515 unit uPSI_GraphWin;  //FTL                //VCL
33686: 516 unit uPSI_actionMain;  //FTL               //VCL
33687: 517 unit uPSI_StSpawn;                         //SysTools
33688: 518 unit uPSI_CtlPanel;                        //VCL
33689: 519 unit uPSI_IdLPR;                           //Indy
33690: 520 unit uPSI_SockRequestInterpreter;          //Indy
33691: 521 unit uPSI_ulambert;                        //DMath
33692: 522 unit uPSI_ucholesk;                        //DMath
33693: 523 unit uPSI_SimpleDS;                        //VCL
33694: 524 unit uPSI_DBXSqlScanner;                   //VCL
33695: 525 unit uPSI_DBXMetaDataUtil;                 //VCL
33696: 526 unit uPSI_Chart;                           //TEE
33697: 527 unit uPSI_TeeProcs;                        //TEE
33698: 528 unit mXBDEUtils;                           //mX4
33699: 529 unit uPSI_MDIEdit;                         //VCL
33700: 530 unit uPSI_CopyPrsr;                        //VCL
33701: 531 unit uPSI_SockApp;                         //VCL
33702: 532 unit uPSI_AppEvnts;                        //VCL
33703: 533 unit uPSI_ExtActns;                        //VCL
33704: 534 unit uPSI_TeEngine;                        //TEE
33705: 535 unit uPSI_CoolMain; //browser             //VCL
33706: 536 unit uPSI_StCRC;                           //SysTools
33707: 537 unit uPSI_StDecMth2;                       //SysTools
33708: 538 unit uPSI_frmExportMain;                   //Synedit
33709: 539 unit uPSI_SynDBEdit;                       //Synedit
33710: 540 unit uPSI_SynEditWildcardSearch;           //Synedit
33711: 541 unit uPSI_BoldComUtils;                    //BOLD
33712: 542 unit uPSI_BoldIsoDateTime;                 //BOLD
33713: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils     //BOLD
33714: 544 unit uPSI_BoldXMLRequests;                 //BOLD
33715: 545 unit uPSI_BoldStringList;                  //BOLD
33716: 546 unit uPSI_BoldFileHandler;                 //BOLD
33717: 547 unit uPSI_BoldContainers;                  //BOLD
33718: 548 unit uPSI_BoldQueryUserDlg;                //BOLD
33719: 549 unit uPSI_BoldWinINet;                     //BOLD
33720: 550 unit uPSI_BoldQueue;                       //BOLD
33721: 551 unit uPSI_JvPcx;                           //JCL
33722: 552 unit uPSI_IdWhois;                         //Indy
33723: 553 unit uPSI_IdWhoIsServer;                   //Indy
33724: 554 unit uPSI_IdGopher;                        //Indy
33725: 555 unit uPSI_IdDateTimeStamp;                 //Indy
33726: 556 unit uPSI_IdDayTimeServer;                 //Indy
33727: 557 unit uPSI_IdDayTimeUDP;                    //Indy
33728: 558 unit uPSI_IdDayTimeUDPServer;              //Indy
33729: 559 unit uPSI_IdDICTServer;                    //Indy
33730: 560 unit uPSI_IdDiscardServer;                 //Indy
33731: 561 unit uPSI_IdDiscardUDPServer;              //Indy
33732: 562 unit uPSI_IdMappedFTP;                     //Indy
33733: 563 unit uPSI_IdMappedPortTCP;                 //Indy
33734: 564 unit uPSI_IdGopherServer;                  //Indy
33735: 565 unit uPSI_IdQotdServer;                    //Indy
33736: 566 unit uPSI_JvRgbToHtml;                     //JCL
33737: 567 unit uPSI_JvRemLog,                        //JCL
33738: 568 unit uPSI_JvSysComp;                       //JCL
33739: 569 unit uPSI_JvTMTL;                          //JCL
33740: 570 unit uPSI_JvWinampAPI;                     //JCL
33741: 571 unit uPSI_MSysUtils;                       //mX4
33742: 572 unit uPSI_ESBMaths;                        //ESB
33743: 573 unit uPSI_ESBMaths2;                       //ESB
33744: 574 unit uPSI_uLkJSON;                         //Lk
33745: 575 unit uPSI_ZURL;  //Zeos                   //Zeos
33746: 576 unit uPSI_ZSysUtils;                       //Zeos
33747: 577 unit unaUtils internals                    //UNA
33748: 578 unit uPSI_ZMatchPattern;                   //Zeos
33749: 579 unit uPSI_ZClasses;                        //Zeos
33750: 580 unit uPSI_ZCollections;                    //Zeos
33751: 581 unit uPSI_ZEncoding;                       //Zeos
33752: 582 unit uPSI_IdRawBase;                       //Indy
33753: 583 unit uPSI_IdNTLM;                          //Indy
33754: 584 unit uPSI_IdNNTP;                          //Indy
33755: 585 unit uPSI_usniffer; //PortScanForm        //mX4
33756: 586 unit uPSI_IdCoderMIME;                     //Indy
33757: 587 unit uPSI_IdCoderUUE;                      //Indy
33758: 588 unit uPSI_IdCoderXXE;                      //Indy
33759: 589 unit uPSI_IdCoder3to4;                     //Indy
```

```
33760: 590 unit uPSI_IdCookie;                          //Indy
33761: 591 unit uPSI_IdCookieManager;                   //Indy
33762: 592 unit uPSI_WDosSocketUtils;                   //WDos
33763: 593 unit uPSI_WDosPlcUtils;                      //WDos
33764: 594 unit uPSI_WDosPorts;                         //WDos
33765: 595 unit uPSI_WDosResolvers;                     //WDos
33766: 596 unit uPSI_WDosTimers;                        //WDos
33767: 597 unit uPSI_WDosPlcs;                          //WDos
33768: 598 unit uPSI_WDosPneumatics;                    //WDos
33769: 599 unit uPSI_IdFingerServer;                    //Indy
33770: 600 unit uPSI_IdDNSResolver;                     //Indy
33771: 601 unit uPSI_IdHTTPWebBrokerBridge;             //Indy
33772: 602 unit uPSI_IdIntercept;                       //Indy
33773: 603 unit uPSI_IdIPMCastBase;                     //Indy
33774: 604 unit uPSI_IdLogBase;                         //Indy
33775: 605 unit uPSI_IdIOHandlerStream;                 //Indy
33776: 606 unit uPSI_IdMappedPortUDP;                   //Indy
33777: 607 unit uPSI_IdQOTDUDPServer;                   //Indy
33778: 608 unit uPSI_IdQOTDUDP;                         //Indy
33779: 609 unit uPSI_IdSysLog;                          //Indy
33780: 610 unit uPSI_IdSysLogServer;                    //Indy
33781: 611 unit uPSI_IdSysLogMessage;                   //Indy
33782: 612 unit uPSI_IdTimeServer;                      //Indy
33783: 613 unit uPSI_IdTimeUDP;                         //Indy
33784: 614 unit uPSI_IdTimeUDPServer;                   //Indy
33785: 615 unit uPSI_IdUserAccounts;                    //Indy
33786: 616 unit uPSI_TextUtils;                         //mX4
33787: 617 unit uPSI_MandelbrotEngine;                  //mX4
33788: 618 unit uPSI_delphi_arduino_Unit1;              //mX4
33789: 619 unit uPSI_DTDSchema2;                        //mX4
33790: 620 unit uPSI_fplotMain;                         //DMath
33791: 621 unit uPSI_FindFileIter;                      //mX4
33792: 622 unit uPSI_PppState;  (JclStrHashMap)         //PPP
33793: 623 unit uPSI_PppParser;                         //PPP
33794: 624 unit uPSI_PppLexer;                          //PPP
33795: 625 unit uPSI_PCharUtils;                        //PPP
33796: 626 unit uPSI_uJSON;                             //WU
33797: 627 unit uPSI_JclStrHashMap;                     //JCL
33798: 628 unit uPSI_JclHookExcept;                     //JCL
33799: 629 unit uPSI_EncdDecd;                          //VCL
33800: 630 unit uPSI_SockAppReg;                        //VCL
33801: 631 unit uPSI_PJFileHandle;                      //PJ
33802: 632 unit uPSI_PJEnvVars;                         //PJ
33803: 633 unit uPSI_PJPipe;                            //PJ
33804: 634 unit uPSI_PJPipeFilters;                     //PJ
33805: 635 unit uPSI_PJConsoleApp;                      //PJ
33806: 636 unit uPSI_UConsoleAppEx;                     //PJ
33807: 637 unit uPSI_DbxSocketChannelNative,            //VCL
33808: 638 unit uPSI_DbxDataGenerator,                  //VCL
33809: 639 unit uPSI_DBXClient;                         //VCL
33810: 640 unit uPSI_IdLogEvent;                        //Indy
33811: 641 unit uPSI_Reversi;                           //mX4
33812: 642 unit uPSI_Geometry;                          //mX4
33813: 643 unit uPSI_IdSMTPServer;                      //Indy
33814: 644 unit uPSI_Textures;                          //mX4
33815: 645 unit uPSI_IBX;                               //VCL
33816: 646 unit uPSI_IWDBCommon;                        //VCL
33817: 647 unit uPSI_SortGrid;                          //mX4
33818: 648 unit uPSI_IB;                                //VCL
33819: 649 unit uPSI_IBScript;                          //VCL
33820: 650 unit uPSI_JvCSVBaseControls;                 //JCL
33821: 651 unit uPSI_Jvg3DColors;                       //JCL
33822: 652 unit uPSI_JvHLEditor;  //beat                //JCL
33823: 653 unit uPSI_JvShellHook;                       //JCL
33824: 654 unit uPSI_DBCommon2                          //VCL
33825: 655 unit uPSI_JvSHFileOperation;                 //JCL
33826: 656 unit uPSI_uFilexport;                        //mX4
33827: 657 unit uPSI_JvDialogs;                         //JCL
33828: 658 unit uPSI_JvDBTreeView;                      //JCL
33829: 659 unit uPSI_JvDBUltimGrid;                     //JCL
33830: 660 unit uPSI_JvDBQueryParamsForm;               //JCL
33831: 661 unit uPSI_JvExControls;                      //JCL
33832: 662 unit uPSI_JvBDEMemTable;                     //JCL
33833: 663 unit uPSI_JvCommStatus;                      //JCL
33834: 664 unit uPSI_JvMailSlots2;                      //JCL
33835: 665 unit uPSI_JvgWinMask;                        //JCL
33836: 666 unit uPSI_StEclpse;                          //SysTools
33837: 667 unit uPSI_StMime;                            //SysTools
33838: 668 unit uPSI_StList;                            //SysTools
33839: 669 unit uPSI_StMerge;                           //SysTools
33840: 670 unit uPSI_StStrS;                            //SysTools
33841: 671 unit uPSI_StTree,                            //SysTools
33842: 672 unit uPSI_StVArr;                            //SysTools
33843: 673 unit uPSI_StRegIni;                          //SysTools
33844: 674 unit uPSI_urkf;                              //DMath
33845: 675 unit uPSI_usvd;                              //DMath
33846: 676 unit uPSI_DepWalkUtils;                      //JCL
33847: 677 unit uPSI_OptionsFrm;                        //JCL
33848: 678 unit yuvconverts;                            //mX4
```

```
33849: 679 uPSI_JvPropAutoSave;                        //JCL
33850: 680 uPSI_AclAPI;                                //alcinoe
33851: 681 uPSI_AviCap;                                //alcinoe
33852: 682 uPSI_ALAVLBinaryTree;                       //alcinoe
33853: 683 uPSI_ALFcnMisc;                             //alcinoe
33854: 684 uPSI_ALStringList;                          //alcinoe
33855: 685 uPSI_ALQuickSortList;                       //alcinoe
33856: 686 uPSI_ALStaticText;                          //alcinoe
33857: 687 uPSI_ALJSONDoc;                             //alcinoe
33858: 688 uPSI_ALGSMComm;                             //alcinoe
33859: 689 uPSI_ALWindows;                             //alcinoe
33860: 690 uPSI_ALMultiPartFormDataParser;             //alcinoe
33861: 691 uPSI_ALHttpCommon;                          //alcinoe
33862: 692 uPSI_ALWebSpider,                           //alcinoe
33863: 693 uPSI_ALHttpClient;                          //alcinoe
33864: 694 uPSI_ALFcnHTML;                             //alcinoe
33865: 695 uPSI_ALFTPClient;                           //alcinoe
33866: 696 uPSI_ALInternetMessageCommon;               //alcinoe
33867: 697 uPSI_ALWininetHttpClient;                   //alcinoe
33868: 698 uPSI_ALWinInetFTPClient;                    //alcinoe
33869: 699 uPSI_ALWinHttpWrapper;                      //alcinoe
33870: 700 uPSI_ALWinHttpClient;                       //alcinoe
33871: 701 uPSI_ALFcnWinSock;                          //alcinoe
33872: 702 uPSI_ALFcnSQL;                              //alcinoe
33873: 703 uPSI_ALFcnCGI;                              //alcinoe
33874: 704 uPSI_ALFcnExecute;                          //alcinoe
33875: 705 uPSI_ALFcnFile;                             //alcinoe
33876: 706 uPSI_ALFcnMime;                             //alcinoe
33877: 707 uPSI_ALPhpRunner;                           //alcinoe
33878: 708 uPSI_ALGraphic;                             //alcinoe
33879: 709 uPSI_ALIniFiles;                            //alcinoe
33880: 710 uPSI_ALMemCachedClient;                     //alcinoe
33881: 711 unit uPSI_MyGrids;                          //mX4
33882: 712 uPSI_ALMultiPartMixedParser                 //alcinoe
33883: 713 uPSI_ALSMTPClient                           //alcinoe
33884: 714 uPSI_ALNNTPClient;                          //alcinoe
33885: 715 uPSI_ALHintBalloon;                         //alcinoe
33886: 716 unit uPSI_ALXmlDoc;                         //alcinoe
33887: 717 unit uPSI_IPCThrd;                          //VCL
33888: 718 unit uPSI_MonForm;                          //VCL
33889: 719 unit uPSI_TeCanvas;                         //Orpheus
33890: 720 unit uPSI_Ovcmisc;                          //Orpheus
33891: 721 unit uPSI_ovcfiler;                         //Orpheus
33892: 722 unit uPSI_ovcstate;                         //Orpheus
33893: 723 unit uPSI_ovccoco;                          //Orpheus
33894: 724 unit uPSI_ovcrvexp;                         //Orpheus
33895: 725 unit uPSI_OvcFormatSettings;                //Orpheus
33896: 726 unit uPSI_OvcUtils;                         //Orpheus
33897: 727 unit uPSI_ovcstore;                         //Orpheus
33898: 728 unit uPSI_ovcstr;                           //Orpheus
33899: 729 unit uPSI_ovcmru;                           //Orpheus
33900: 730 unit uPSI_ovccmd;                           //Orpheus
33901: 731 unit uPSI_ovctimer;                         //Orpheus
33902: 732 unit uPSI_ovcintl;                          //Orpheus
33903: 733 uPSI_AfCircularBuffer;                      //AsyncFree
33904: 734 uPSI_AfUtils;                               //AsyncFree
33905: 735 uPSI_AfSafeSync;                            //AsyncFree
33906: 736 uPSI_AfComPortCore;                         //AsyncFree
33907: 737 uPSI_AfComPort;                             //AsyncFree
33908: 738 uPSI_AfPortControls;                        //AsyncFree
33909: 739 uPSI_AfDataDispatcher;                      //AsyncFree
33910: 740 uPSI_AfViewers;                             //AsyncFree
33911: 741 uPSI_AfDataTerminal;                        //AsyncFree
33912: 742 uPSI_SimplePortMain;                        //AsyncFree
33913: 743 unit uPSI_ovcclock;                         //Orpheus
33914: 744 unit uPSI_o32intlst;                        //Orpheus
33915: 745 unit uPSI_o32ledlabel;                      //Orpheus
33916: 746 unit uPSI_AlMySqlClient;                    //alcinoe
33917: 747 unit uPSI_ALFBXClient;                      //alcinoe
33918: 748 unit uPSI_ALFcnSQL;                         //alcinoe
33919: 749 unit uPSI_AsyncTimer;                       //mX4
33920: 750 unit uPSI_ApplicationFileIO;                //mX4
33921: 751 unit uPSI_PsAPI;                            //VCLé
33922: 752 uPSI_ovcuser;                               //Orpheus
33923: 753 uPSI_ovcurl;                                //Orpheus
33924: 754 uPSI_ovcvlb;                                //Orpheus
33925: 755 uPSI_ovccolor;                              //Orpheus
33926: 756 uPSI_ALFBXLib,                              //alcinoe
33927: 757 uPSI_ovcmeter;                              //Orpheus
33928: 758 uPSI_ovcpeakm;                              //Orpheus
33929: 759 uPSI_O32BGSty;                              //Orpheus
33930: 760 uPSI_ovcBidi;                               //Orpheus
33931: 761 uPSI_ovctcary;                              //Orpheus
33932: 762 uPSI_DXPUtils;                              //mX4
33933: 763 uPSI_ALMultiPartBaseParser;                 //alcinoe
33934: 764 uPSI_ALMultiPartAlternativeParser;          //alcinoe
33935: 765 uPSI_ALPOP3Client;                          //alcinoe
33936: 766 uPSI_SmallUtils;                            //mX4
33937: 767 uPSI_MakeApp;                               //mX4
```

```
33938: 768 uPSI_O32MouseMon;                          //Orpheus
33939: 769 uPSI_OvcCache;                             //Orpheus
33940: 770 uPSI_ovccalc;                              //Orpheus
33941: 771 uPSI_Joystick,                             //OpenGL
33942: 772 uPSI_ScreenSaver;                          //OpenGL
33943: 773 uPSI_XCollection,                          //OpenGL
33944: 774 uPSI_Polynomials,                          //OpenGL
33945: 775 uPSI_PersistentClasses, //9.86             //OpenGL
33946: 776 uPSI_VectorLists;                          //OpenGL
33947: 777 uPSI_XOpenGL,                             //OpenGL
33948: 778 uPSI_MeshUtils;                            //OpenGL
33949: 779 unit uPSI_JclSysUtils;                     //JCL
33950: 780 unit uPSI_JclBorlandTools;                 //JCL
33951: 781 unit JclFileUtils_max;                     //JCL
33952: 782 uPSI_AfDataControls,                       //AsyncFree
33953: 783 uPSI_GLSilhouette;                         //OpenGL
33954: 784 uPSI_JclSysUtils_class;                    //JCL
33955: 785 uPSI_JclFileUtils_class;                   //JCL
33956: 786 uPSI_FileUtil;                             //JCL
33957: 787 uPSI_changefind;                           //mX4
33958: 788 uPSI_cmdIntf;                              //mX4
33959: 789 uPSI_fservice;                             //mX4
33960: 790 uPSI_Keyboard;                             //OpenGL
33961: 791 uPSI_VRMLParser,                           //OpenGL
33962: 792 uPSI_GLFileVRML,                           //OpenGL
33963: 793 uPSI_Octree;                               //OpenGL
33964: 794 uPSI_GLPolyhedron,                         //OpenGL
33965: 795 uPSI_GLCrossPlatform;                      //OpenGL
33966: 796 uPSI_GLParticles;                          //OpenGL
33967: 797 uPSI_GLNavigator;                          //OpenGL
33968: 798 uPSI_GLStarRecord;                         //OpenGL
33969: 799 uPSI_GLTextureCombiners;                   //OpenGL
33970: 800 uPSI_GLCanvas;                             //OpenGL
33971: 801 uPSI_GeometryBB;                           //OpenGL
33972: 802 uPSI_GeometryCoordinates;                  //OpenGL
33973: 803 uPSI_VectorGeometry;                       //OpenGL
33974: 804 uPSI_BumpMapping;                          //OpenGL
33975: 805 uPSI_TGA;                                  //OpenGL
33976: 806 uPSI_GLVectorFileObjects;                  //OpenGL
33977: 807 uPSI_IMM;                                  //VCL
33978: 808 uPSI_CategoryButtons;                      //VCL
33979: 809 uPSI_ButtonGroup;                          //VCL
33980: 810 uPSI_DbExcept;                             //VCL
33981: 811 uPSI_AxCtrls;                              //VCL
33982: 812 uPSI_GL_actorUnit1;                        //OpenGL
33983: 813 uPSI_StdVCL;                               //VCL
33984: 814 unit CurvesAndSurfaces;                    //OpenGL
33985: 815 uPSI_DataAwareMain;                        //AsyncFree
33986: 816 uPSI_TabNotBk;                             //VCL
33987: 817 uPSI_udwsfiler;                            //mX4
33988: 818 uPSI_synaip;                               //Synapse!
33989: 819 uPSI_synacode;                             //Synapse
33990: 820 uPSI_synachar;                             //Synapse
33991: 821 uPSI_synamisc;                             //Synapse
33992: 822 uPSI_synaser;                              //Synapse
33993: 823 uPSI_synaicnv;                             //Synapse
33994: 824 uPSI_tlntsend;                             //Synapse
33995: 825 uPSI_pingsend;                             //Synapse
33996: 826 uPSI_blcksock;                             //Synapse
33997: 827 uPSI_asn1util;                             //Synapse
33998: 828 uPSI_dnssend;                              //Synapse
33999: 829 uPSI_clamsend;                             //Synapse
34000: 830 uPSI_ldapsend;                             //Synapse
34001: 831 uPSI_mimemess;                             //Synapse
34002: 832 uPSI_slogsend;                             //Synapse
34003: 833 uPSI_mimepart;                             //Synapse
34004: 834 uPSI_mimeinln;                             //Synapse
34005: 835 uPSI_ftpsend,                              //Synapse
34006: 836 uPSI_ftptsend;                             //Synapse
34007: 837 uPSI_httpsend;                             //Synapse
34008: 838 uPSI_sntpsend;                             //Synapse
34009: 839 uPSI_smtpsend;                             //Synapse
34010: 840 uPSI_snmpsend;                             //Synapse
34011: 841 uPSI_imapsend;                             //Synapse
34012: 842 uPSI_pop3send;                             //Synapse
34013: 843 uPSI_nntpsend;                             //Synapse
34014: 844 uPSI_ssl_cryptlib;                         //Synapse
34015: 845 uPSI_ssl_openssl;                          //Synapse
34016: 846 uPSI_synhttp_daemon;                       //Synapse
34017: 847 uPSI_NetWork;                              //mX4
34018: 848 uPSI_PingThread;                           //Synapse
34019: 849 uPSI_JvThreadTimer;                        //JCL
34020: 850 unit uPSI_wwSystem;                        //InfoPower
34021: 851 unit uPSI_IdComponent;                     //Indy
34022: 852 unit uPSI_IdIOHandlerThrottle;             //Indy
34023: 853 unit uPSI_Themes;                          //VCL
34024: 854 unit uPSI_StdStyleActnCtrls;               //VCL
34025: 855 unit uPSI_UDDIHelper;                      //VCL
34026: 856 unit uPSI_IdIMAP4Server;                   //Indy
```

```
34027: 857 uPSI_VariantSymbolTable,                    //VCL //3.9.9.92
34028: 858 uPSI_udf_glob,                              //mX4
34029: 859 uPSI_TabGrid,                               //VCL
34030: 860 uPSI_JsDBTreeView,                          //mX4
34031: 861 uPSI_JsSendMail,                            //mX4
34032: 862 uPSI_dbTvRecordList,                        //mX4
34033: 863 uPSI_TreeVwEx,                              //mX4
34034: 864 uPSI_ECDataLink,                            //mX4
34035: 865 uPSI_dbTree,                                //mX4
34036: 866 uPSI_dbTreeCBox,                            //mX4
34037: 867 unit uPSI_Debug; //TfrmDebug                //mX4
34038: 868 uPSI_TimeFncs;                              //mX4
34039: 869 uPSI_FileIntf,                              //VCL
34040: 870 uPSI_SockTransport,                         //RTL
34041: 871 unit uPSI_WinInet;                          //RTL
34042: 872 unit uPSI_Wwstr;                            //mX4
34043: 873 uPSI_DBLookup,                              //VCL
34044: 874 uPSI_Hotspot,                               //mX4
34045: 875 uPSI_HList; //History List                  //mX4
34046: 876 unit uPSI_DrTable;                          //VCL
34047: 877 uPSI_TConnect,                              //VCL
34048: 878 uPSI_DataBkr,                               //VCL
34049: 879 uPSI_HTTPIntr;                              //VCL
34050: 880 unit uPSI_Mathbox;                          //mX4
34051: 881 uPSI_cyIndy,                                //cY
34052: 882 uPSI_cySysUtils,                            //cY
34053: 883 uPSI_cyWinUtils,                            //cY
34054: 884 uPSI_cyStrUtils,                            //cY
34055: 885 uPSI_cyObjUtils,                            //cY
34056: 886 uPSI_cyDateUtils,                           //cY
34057: 887 uPSI_cyBDE,                                 //cY
34058: 888 uPSI_cyClasses,                             //cY
34059: 889 uPSI_cyGraphics,  //3.9.9.94_2              //cY
34060: 890 unit uPSI_cyTypes;                          //cY
34061: 891 uPSI_JvDateTimePicker,                      //JCL
34062: 892 uPSI_JvCreateProcess,                       //JCL
34063: 893 uPSI_JvEasterEgg,                           //JCL
34064: 894 uPSI_WinSvc,  //3.9.9.94_3                  //VCL
34065: 895 uPSI_SvcMgr                                 //VCL
34066: 896 unit uPSI_JvPickDate;                       //JCL
34067: 897 unit uPSI_JvNotify;                         //JCL
34068: 898 uPSI_JvStrHlder                             //JCL
34069: 899 unit uPSI_JclNTFS2;                         //JCL
34070: 900 uPSI_Jcl8087 //math coprocessor             //JCL
34071: 901 uPSI_JvAddPrinter                           //JCL
34072: 902 uPSI_JvCabFile                              //JCL
34073: 903 uPSI_JvDataEmbedded;                        //JCL
34074: 904 unit uPSI_U_HexView;                        //mX4
34075: 905 uPSI_UWavein4,                              //mX4
34076: 906 uPSI_AMixer,                                //mX4
34077: 907 uPSI_JvaScrollText,                         //mX4
34078: 908 uPSI_JvArrow,                               //mX4
34079: 909 unit uPSI_UrlMon;                           //mX4
34080: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'    //mX4
34081: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;   //DFF
34082: 912 unit uPSI_DFFUtils;                         //DFF
34083: 913 unit uPSI_MathsLib;                         //DFF
34084: 914 uPSI_UIntList;                              //DFF
34085: 915 uPSI_UGetParens;                            //DFF
34086: 916 unit uPSI_UGeometry;                        //DFF
34087: 917 unit uPSI_UAstronomy;                       //DFF
34088: 918 unit uPSI_UCardComponentV2;                 //DFF
34089: 919 unit uPSI_UTGraphSearch;                    //DFF
34090: 920 unit uPSI_UParser10;                        //DFF
34091: 921 unit uPSI_cyIEUtils;                        //cY
34092: 922 unit uPSI_UcomboV2;                         //DFF
34093: 923 uPSI_cyBaseComm,                            //cY
34094: 924 uPSI_cyAppInstances,                        //cY
34095: 925 uPSI_cyAttract,                             //cY
34096: 926 uPSI_cyDERUtils                             //cY
34097: 927 unit uPSI_cyDocER;                          //cY
34098: 928 unit uPSI_ODBC;                             //mX
34099: 929 unit uPSI_AssocExec;                        //mX
34100: 930 uPSI_cyBaseCommRoomConnector,               //cY
34101: 931 uPSI_cyCommRoomConnector,                   //cY
34102: 932 uPSI_cyCommunicate,                         //cY
34103: 933 uPSI_cyImage;                               //cY
34104: 934 uPSI_cyBaseContainer                        //cY
34105: 935 uPSI_cyModalContainer,                      //cY
34106: 936 uPSI_cyFlyingContainer;                     //cY
34107: 937 uPSI_RegStr,                                //VCL
34108: 938 uPSI_HtmlHelpViewer;                        //VCL
34109: 939 unit uPSI_cyIniForm                         //cY
34110: 940 unit uPSI_cyVirtualGrid;                    //cY
34111: 941 uPSI_Profiler,                              //DA
34112: 942 uPSI_BackgroundWorker,                      //DA
34113: 943 uPSI_WavePlay,                              //DA
34114: 944 uPSI_WaveTimer,                             //DA
34115: 945 uPSI_WaveUtils;                             //DA
```

```
34116: 946 uPSI_NamedPipes,                          //TB
34117: 947 uPSI_NamedPipeServer,                     //TB
34118: 948 unit uPSI_process,                        //TB
34119: 949 unit uPSI_DPUtils;                        //TB
34120: 950 unit uPSI_CommonTools;                    //TB
34121: 951 uPSI_DataSendToWeb,                       //TB
34122: 952 uPSI_StarCalc,                            //TB
34123: 953 uPSI_D2_XPVistaHelperU                    //TB
34124: 954 unit uPSI_NetTools                        //TB
34125: 955 unit uPSI_Pipes                           //TB
34126: 956 uPSI_ProcessUnit,                         //mX
34127: 957 uPSI_adGSM,                               //mX
34128: 958 unit uPSI_BetterADODataSet;               //mX
34129: 959 unit uPSI_AdSelCom; //FTT                 //mX
34130: 960 unit unit uPSI_dwsXPlatform;              //DWS
34131: 961 uPSI_AdSocket;                            //mX Turbo Power
34132: 962 uPSI_AdPacket;                            //mX
34133: 963 uPSI_AdPort;                              //mX
34134: 964 uPSI_PathFunc;                            //Inno
34135: 965 uPSI_CmnFunc;                             //Inno
34136: 966 uPSI_CmnFunc2; //Inno Setup              //Inno
34137: 967 unit uPSI_BitmapImage;                    //mX4
34138: 968 unit uPSI_ImageGrabber;                   //mX4
34139: 969 uPSI_SecurityFunc,                        //Inno
34140: 970 uPSI_RedirFunc,                           //Inno
34141: 971 uPSI_FIFO, (MemoryStream)                 //mX4
34142: 972 uPSI_Int64Em,                             //Inno
34143: 973 unit uPSI_InstFunc;                       //Inno
34144: 974 unit uPSI_LibFusion;                      //Inno
34145: 975 uPSI_SimpleExpression;                    //Inno
34146: 976 uPSI_unitResourceDetails,                 //XN
34147: 977 uPSI_unitResFile,                         //XN
34148: 978 unit uPSI_simpleComport;                  //mX4
34149: 979 unit uPSI_AfViewershelpers;               //Async
34150: 980 unit uPSI_Console;                        //mX4
34151: 981 unit uPSI_AnalogMeter;                    //TB
34152: 982 unit uPSI_XPrinter,                       //TB
34153: 983 unit uPSI_IniFiles;                       //VCL
34154: 984 unit uPSI_lazIniFiles;                    //FP
34155: 985 uPSI_testutils;                           //FP
34156: 986 uPSI_ToolsUnit; (DBTests)                //FP
34157: 987 uPSI_fpcunit                              //FP
34158: 988 uPSI_testdecorator;                       //FP
34159: 989 unit uPSI_fpcunittests;                   //FP
34160: 990 unit uPSI_cTCPBuffer;                     //Fundamentals 4
34161: 991 unit uPSI_Glut,                           //Open GL
34162: 992 uPSI_LEDBitmaps,                          //mX4
34163: 993 uPSI_FileClass,                           //Inno
34164: 994 uPSI_FileUtilsClass,                      //mX4
34165: 995 uPSI_ComPortInterface; //Kit              //mX4
34166: 996 unit uPSI_SwitchLed;                      //mX4
34167: 997 unit uPSI_cyDmmCanvas;                    //cY
34168: 998 uPSI_uColorFunctions;                     //DFF
34169: 999 uPSI_uSettings;                           //DFF
34170: 1000 uPSI_cyDebug.pas                         //cY
34171: 1001 uPSI_cyColorMatrix;                      //cY
34172: 1002 unit uPSI_cyCopyFiles;                   //cY
34173: 1003 unit uPSI_cySearchFiles;                 //cY
34174: 1004 unit uPSI_cyBaseMeasure;                 //cY
34175: 1005 unit uPSI_PJIStreams;                    //DD
34176: 1006 unit uPSI_cyRunTimeResize;               //cY
34177: 1007 unit uPSI_jcontrolutils;                 //cY
34178: 1008 unit uPSI_kcMapViewer; (+GEONames)       //kc
34179: 1009 unit uPSI_kcMapViewerDESynapse;          //kc
34180: 1010 unit uPSI_cparserutils; (+GIS_SysUtils)  //kc
34181: 1011 unit uPSI_LedNumber;                     //TurboPower
34182: 1012 unit uPSI_StStrL;                        //SysTools
34183: 1013 unit uPSI_indGnouMeter;                  //LAZ
34184: 1014 unit uPSI_Sensors;                       //LAZ
34185: 1015 unit uPSI_pwmain;  //cgi of powtils      //Pow
34186: 1016 unit uPSI_HTMLUtil;                      //Pow
34187: 1017 unit uPSI_synwrap1; //httpsend           //Pow
34188: 1018 unit StreamWrap1                         //Pow
34189: 1019 unit uPSI_pwmain;                        //Pow
34190: 1020 unit pwtypes                             //Pow
34191: 1021 uPSI_W32VersionInfo                      //LAZ
34192: 1022 unit uPSI_IpAnim;                        //LAZ
34193: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
34194: 1024 unit uPSI_LrtPoTools;                    //LAZ
34195: 1025 unit uPSI_Laz_DOM;                       //LAZ
34196: 1026 unit uPSI_hhAvComp;                      //LAZ
34197: 1027 unit uPSI_GPS2;                          //mX4
34198: 1028 unit uPSI_GPS;                           //mX4
34199: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo//mX4
34200: 1030 unit uPSI_NMEA;     // GPS               //mX4
34201: 1031 unit uPSI_ScreenThreeDLab;               //mX4
34202: 1032 unit uPSI_Spin;                          //VCL
34203: 1033 unit uPSI_DynaZip;                       //mX4
34204: 1034 unit uPSI_clockExpert;                   //TB
```

```
34205: 1035 unit debugLn                                //mX4
34206: 1036 uPSI_SortUtils;                             //Jcl
34207: 1037 uPSI_BitmapConversion;                      //Jcl
34208: 1038 unit uPSI_JclTD32;                          //Jcl
34209: 1039 unit uPSI_ZDbcUtils;                        //Zeos
34210: 1040 unit uPSI_ZScriptParser;                    //Zeos
34211: 1041 uPSI_JvIni,                                 //JCL
34212: 1042 uPSI_JvFtpGrabber;                          //JCL
34213: 1043 unit uPSI_NeuralNetwork;                    //OCL
34214: 1044 unit uPSI_StExpr;                           //SysTools
34215: 1045 unit uPSI_GR32_Geometry;                    //GR32
34216: 1046 unit uPSI_GR32_Containers;                  //GR32
34217: 1047 unit uPSI_GR32_Backends_VCL,                //GR32
34218: 1048 unit uPSI_StSaturn;//Venus+Mercury+Mars++ //SysTools
34219: 1049 unit uPSI_JclParseUses;                     //JCL
34220: 1050 unit uPSI_JvFinalize;                       //JCL
34221: 1051 unit uPSI_panUnit1;                         //GLScene
34222: 1052 unit uPSI_DD83u1;  //Arduino Tester        //mX4
34223: 1053 unit uPSI_BigIni                            //Hinzen
34224: 1054 unit uPSI_ShellCtrls;                       //VCL
34225: 1055 unit uPSI_fmath;                            //FMath
34226: 1056 unit uPSI_fComp;                            //FMath
34227: 1057 unit uPSI_HighResTimer;                     //Lauer
34228: 1058 unit uconvMain; (Unit Converter)            //PS
34229: 1059 unit uPSI_uconvMain;                        //PS
34230: 1060 unit uPSI_ParserUtils;                      //PS
34231: 1061 unit uPSI_uPSUtils;                         //PS
34232: 1062 unit uPSI_ParserU;                          //PS
34233: 1063 unit uPSI_TypInfo;                          //VCL
34234: 1064 unit uPSI_ServiceMgr;                       //mX
34235: 1065 unit uPSI_UDict;                            //DFF
34236: 1066 unit uPSI_ubigFloatV3;                      //DFF
34237: 1067 unit uPSI_UBigIntsV4;                       //DFF
34238: 1068 unit uPSI_ServiceMgr2;                      //mX
34239: 1069 unit uPSI_UP10Build;                        //PS
34240: 1070 unit uPSI_UParser10;                        //PS
34241: 1071 unit uPSI_IdModBusServer;                   //MB
34242: 1072 unit uPSI_IdModBusClient;                   //MB
34243: 1073 unit uPSI_ColorGrd;                         //VCL
34244: 1074 unit uPSI_DirOutln;                         //VCL
34245: 1075 unit uPSI_Gauges;                           //VCL
34246: 1076 unit uPSI_CustomizeDlg;                     //VCL
34247: 1077 unit uPSI_ActnMan;                          //VCL
34248: 1078 unit uPSI_CollPanl;                         //VCL
34249: 1079 unit uPSI_Calendar2;                        //VCL
34250: 1080 unit uPSI_IBCtrls;                          //VCL
34251: 1081 unit uPSI_IdStackWindows;                   //Indy
34252: 1082 unit uPSI_CTSVendorUtils;                   //DBX
34253: 1083 unit uPSI_VendorTestFramework;              //DBX
34254: 1084 unit uPSI_TInterval;                        //mX4
34255: 1085 unit uPSI_JvAnimate                         //JCL
34256: 1086 unit uPSI_DBXCharDecoder;                   //DBX
34257: 1087 unit uPSI_JvDBLists;                        //JCL
34258: 1088 unit uPSI_JvFileInfo;                       //JCL
34259: 1089 unit uPSI_SOAPConn;                         //VCL
34260: 1090 unit uPSI_SOAPLinked;                       //VCL
34261: 1091 unit uPSI_XSBuiltIns;                       //VCL
34262: 1092 unit uPSI_JvgDigits;                        //JCL
34263: 1093 unit uPSI_JvDesignUtils;
34264: 1094 unit uPSI_JvgCrossTable;
34265: 1095 unit uPSI_JvgReport;
34266: 1096 unit uPSI_JvDBRichEdit;
34267: 1097 unit uPSI_JvWinHelp;
34268: 1098 unit uPSI_WaveConverter;
34269: 1099 unit uPSI_ACMConvertor;
34270: 1100 unit XSBuiltIns_Routines
34271: 1101 unit uPSI_ComObjOleDB_utils.pas
34272: 1102 unit uPSI_SMScript;
34273: 1103 unit uPSI_CompFileIo;
34274: 1104 unit uPSI_SynHighlighterGeneral;
34275: 1105 unit uPSI_geometry2;
34276: 1106 unit uPSI_MConnect;
34277: 1107 unit uPSI_ObjBrkr;
34278: 1108 unit uPSI_uMultiStr;
34279: 1109 unit uPSI_WinAPI.pas;
34280: 1110 unit uPSI_JvAVICapture;
34281: 1111 unit uPSI_JvExceptionForm;
34282: 1112 unit uPSI_JvConnectNetwork;
34283: 1113 unit uPSI_MTMainForm;
34284: 1114 unit uPSI_DdeMan;
34285: 1115 unit uPSI_DIUtils;
34286: 1116 unit uPSI_gnugettext;
34287: 1117 unit uPSI_Xmlxform;
34288: 1118 unit uPSI_SvrHTTPIndy;
34289: 1119 unit uPSI_CPortTrmSet;
34290: 1120 unit SvrLog; ----------------------------------------------------------------
34291: 1121 unit uPSI_IndySockTransport.pas (+IdHTTPHeaderInfo) //mX4
34292: 1122 unit uPSI_HTTPProd.pas
34293: 1123 unit uPSI_CppParser.pas
```

```
34294: 1124 unit uPSI_SynHighlighterCpp.pas
34295: 1125 unit uPSI_CodeCompletion.pas
34296: 1126 unit uPSI_U_IntList2.pas
34297: 1127 unit uPSI_SockHTTP.pas
34298: 1128 uPSI_SockAppNotify.pas
34299: 1129 uPSI_NSToIS.pas
34300: 1130 unit uPSI_DBOleCtl.pas
34301: 1131 unit uPSI_xercesxmldom;
34302: 1132 unit uPSI_xmldom;
34303: 1133 unit uPSI_Midas;
34304: 1134 unit uPSI_JclExprEval;
34305: 1135 uPSI_Gameboard;
34306: 1136 unit uPSI_ExtUtil;
34307: 1137 unit uPSI_FCGIApp;
34308: 1138 unit uPSI_ExtPascal;
34309: 1139 unit uPSI_PersistSettings;
34310: 1140 IdHTTPHeaderInfo.pas
34311: 1141 uPSI_SynEditAutoComplete;
34312: 1142 uPSI_SynEditTextBuffer.pas
34313: 1143 unit uPSI_JclPCRE;
34314: 1144 unit uPSI_ZConnection;
34315: 1145 unit uPSI_ZSequence;
34316: 1146 unit uPSI_ChessPrg;
34317: 1147 unit uPSI_ChessBrd;
34318: 1148 unit uPSI_Graph3D;
34319: 1149 uPSI_SysInfoCtrls2.pas
34320: 1150 unit uPSI_RegUtils;
34321: 1151 unit uPSI_VariantRtn;
34322: 1152 uPSI_StdFuncs,
34323: 1153 unit uPSI_SqlTxtRtns;
34324: 1154 unit uPSI_BSpectrum;
34325: 1155 unit IPAddressControl;
34326: 1156 unit uPSI_Paradox;
34327: 1157 unit uPSI_Environ;
34328: 1158 uPSI_GraphicsPrimitivesLibrary;
34329: 1159 uPSI_DrawFigures,
34330: 1160 unit uPSI_synadbg;
34331: 1161 unit uPSI_BitStream;
34332: 1162 unit uPSI_xrtl_util_FileVersion;
34333: 1163 uPSI_XmlRpcCommon,
34334: 1164 unit uPSI_XmlRpcClient;
34335: 1165 unit uPSI_XmlRpcTypes;
34336: 1166 unit uPSI_XmlRpcServer;
34337: 1167 unit uPSI_SynAutoIndent;
34338: 1168 unit uPSI_synafpc;
34339: 1169 unit uPSI_RxNotify;
34340: 1170 unit uPSI_SynAutoCorrect;
34341: 1171 unit uPSI_rxOle2Auto;
34342: 1172 unit uPSI_Spring_Utilsmx;
34343: 1173 unit uPSI_ulogifit;
34344: 1174 unit uPSI_HarmFade;
34345: 1175 unit uPSI_SynCompletionProposal;
34346: 1176 unit uPSI_rxAniFile;
34347: 1177 uPSI_ulinfit,
34348: 1178 uPSI_usvdfit;
34349: 1179 unit uPSI_JclStringLists;
34350: 1180 unit uPSI_ZLib;
34351: 1181 unit uPSI_MaxTokenizers;  //WANT
34352: 1182 unit uPSI_MaxStrUtils;
34353: 1183 unit uPSI_MaxXMLUtils;
34354: 1184 unit uPSI_MaxUtils;
34355: 1185 unit uPSI_VListBox;
34356: 1186 unit uPSI_MaxDOM;
34357: 1187 unit uPSI_MaxDOMDictionary;
34358: 1188 unit uPSI_MaxDOMDictionary_Routines;
34359: 1189 unit uPSI_cASN1;
34360: 1190 unit uPSI_cX509Certificate;
34361: 1191 unit uPSI_uCiaXml;
34362: 1192 unit uPSI_StringsW;
34363: 1193 unit uPSI_FileStreamW;  //WideString D7X
34364: 1194 unit Drawingutils;
34365: 1195 unit uPSI_InetUtilsUnified;
34366: 1196 unit uPSI_FileMask;
34367: 1197 unit uPSI_StrConv;
34368: 1198 unit uPSI_Simpat;
34369: 1199 unit uPSI_Tooltips.pas
34370: 1200 unit uPSI_StringGridLibrary.pas
34371: 1201 unit uPSI_ChronCheck.pas
34372: 1202 unit uPSI_REXX.pas
34373: 1203 uPSI_SysImg.pas
34374: 1204 unit uPSI_Tokens;
34375: 1205 unit uPSI_KFunctions;
34376: 1206 unit uPSI_KMessageBox;
34377: 1207 unit uPSI_CPUSpeed.pas
34378: 1208 uPSI_RoboTracker.pas
34379: 1209 unit uPSI_NamedPipesImpl.pas
34380: 1210 unit uPSI_KLog.pas
34381: 1211 unit uPSI_NamedPipeThreads.pas
34382: 1212 unit uPSI_MapFiles.pas //map stream of memory-mapped files
```

```
34383: 1213 unit uPSI_BKPwdGen,     //Password Generator
34384: 1214 unit uPSI_Kronos,  // big chrono date time library
34385: 1215 unit uPSI_TokenLibrary2;
34386: 1216 uPSI_KDialogs,
34387: 1217 uPSI_Numedit,
34388: 1218 unit uPSI_StSystem2;
34389: 1219 unit uPSI_KGraphics;
34390: 1220 uPSI_KGraphics_functions;
34391: 1221 uPSI_umaxPipes.pas
34392: 1222 unit uPSI_KControls;
34393: 1223 unit SysUtils_max2;
34394: 1224 uPSI_IdAntiFreeze.pas
34395: 1225 uPSI_IdLogStream.pas
34396: 1226 unit uPSI_IdThreadSafe;
34397: 1227 unit uPSI_IdThreadMgr;
34398: 1228 unit uPSI_IdAuthentication;
34399: 1229 unit uPSI_IdAuthenticationManager;
34400: 1230 uPSI_OverbyteIcsConApp
34401: 1231 unit uPSI_KMemo;
34402: 1232 unit uPSI_OverbyteIcsTicks64;
34403: 1233 unit uPSI_OverbyteIcsSha1.pas
34404: 1234 unit uPSI_KEditCommon.pas
34405: 1235 unit uPSI_UtilsMax4.pas
34406: 1236 unit uPSI_IdNNTPServer;
34407: 1237 unit uPSI_UWANTUtils;
34408: 1238 unit uPSI_UtilsMax5.pas;
34409: 1239 unit uPSI_OverbyteIcsAsn1Utils;
34410: 1240 unit uPSI_IdHTTPHeaderInfo; //mX4 response headers
34411: 1241 uPSI_wmiserv.pas {uPSI_SimpleSFTP.pas}
34412: 1242 uPSI_WbemScripting_TLB.pas
34413: 1243 unit uPSI_uJSON;
34414: 1244 uPSI_RegSvrUtils.pas
34415: 1245 unit uPSI_osFileUtil;
34416: 1246 unit uPSI_SHDocVw; //TWebbrowser
34417: 1247 unit uPSI_SHDocVw_TLB
34418: 1248 uPSC_classes.pas V2
34419: 1249 uPSR_classes.pas V2
34420: 1250 uPSI_U_Oscilloscope4_2
34421: 1251 uPSI_xutils.pas
34422: 1252 uPSI_ietf.pas
34423: 1253 uPSI_iso3166.pas
34424: 1254 uPSI_dateutil_real.pas  //Optima ISO 8601
34425: 1255 uPSI_dateext4.pas
34426: 1256 uPSI_locale.pas         //ISO Formater & Tests
34427: 1257 file charset.inc  //IANA Registered character sets
34428: 1258 unit uPSI_Strings;
34429: 1259 unit uPSI_crc_checks;  //ISO 3309 and ITU-T-V42
34430: 1260 unit uPSI_extDOS;
34431: 1261 uPSI_uBild;  //Steganography
34432: 1262 unit uPSI_SimpleTCP;
34433: 1263 unit uPSI_IdFTPList;
34434: 1264 uPSI_uTPLb_CryptographicLibrary.pas
34435: 1265 uPSI_uTPLb_RSA_Engine.pas
34436: 1266 uPSI_cHugeInt.pas
34437: 1267 unit uPSI_SimpleTCPServer;
34438: 1268 unit uPSI_xBase.pas
34439: 1269 unit uPSI_ImageHistogram.pas
34440: 1270 unit uP_PersistSettings2;
34441: 1271 uPSI_WDosDrivers.pas
34442: 1272 unit uPSI_cCipherRSA;
34443: 1273 uPSI_CromisStreams, (TStreamStorage)
34444: 1274 unit uPSI_Streams,
34445: 1275 uPSI_BitStream,
34446: 1276 uPSI_UJSONFunctions.pas
34447: 1277 uPSI_uTPLb_BinaryUtils.pas
34448: 1278 unit uPSI_USha256.pas  //PascalCoin Blockchain
34449: 1279 uPSI_uTPLb_HashDsc.pas
34450: 1280 uPSI_uTPLb_Hash.pas
34451: 1281 SIRegister_Series(X); (uPSI_Series) //4.2.6.10
34452: 1282 unit uPSI_UTime; (UTime); uPSI_mimeinln2;
34453: 1283 uPSI_uTPLb_StreamCipher.pas
34454: 1284 uPSI_uTPLb_BlockCipher.pas
34455: 1285 uPSI_uTPLb_Asymetric.pas
34456: 1286 uPSI_uTPLb_CodecIntf.pas
34457: 1287 uPSI_uTPLb_Codec.pas
34458: 1288 uPSI_ADOInt.pas
34459: 1289 uPSI_MidasCon.pas
34460: 1290 uPSI_XMLDoc.pas
34461: 1291 uPSI_XMLIntf.pas
34462: 1292 uPSI_ProxyUtils.pas
34463: 1293 unit uPSI_maxXMLUtils2;
34464: 1294 unit_StDict_Routines(S: TPSExec);
34465: 1295 unit uPSI_Hashes2
34466: 1296 unit uPSI_IdCoderHeader;
34467: 1297 unit uPSI_BackgroundWorker2
34468: 1298 uPSI_uMRU.pas
34469: 1299 unit FANN.pas
34470: 1300 unit uPSI_FannNetwork.pas
34471: 1301 unit uPSI_RTLDateTimeplus.pas
```

```
34472: 1302 uPSI_ULog.pas
34473: 1303 uPSI_UThread.pas
34474: 1304 uPSI_UTCPIP.pas
34475: 1305 Synapse_OpenSSLv11
34476: 1306 PascalCoin configuration config.inc
34477: 1307 unit uPSI_statmach,   {StateMachine}        //46310
34478: 1308 uPSI_uTPLb_RSA_Primitives,
34479: 1309 unit uPSI_UMatrix,
34480: 1310 uPSI_DXUtil,
34481: 1311 uPSI_crlfParser,
34482: 1312 unit uPSI_DCPbase64;
34483: 1313 unit uPSI_FlyFilesUtils;
34484: 1314 uPSI_PJConsoleApp.pas
34485: 1315 uPSI_PJStreamWrapper.pas
34486: 1316 uPSI_LatLonDist,    //DFF
34487: 1317 uPSI_cHash2.pas     //Fundamentals
34488: 1318 uPSI_ZLib2.pas
34489: 1319 unit uPSI_commDriver
34490: 1320 unit uPSI_PXLNetComs.pas   //PXL
34491: 1321 unit uPSI_PXLTiming.pas    //PXL
34492: 1322 uPSI_Odometer.pas
34493: 1323 unit uPSI_UIntList2;
34494: 1324 uPSI_UIntegerpartition.pas
34495: 1325 unit uPSI_idPHPRunner.pas  //prepare for PHP4D
34496: 1326 unit uPSI_idCGIRunner.pas
34497: 1327 uPSI_DrBobCGI,             //4.7.1.20
34498: 1328 uPSI_OverbyteIcsLogger, / / n  e w   c o u n t   1 2 2 8 - - > 1 3 2 8
34499: 1329 uPSI_OverbyteIcsNntpCli,  testset
34500: 1330 uPSI_OverbyteIcsCharsetUtils,
34501: 1331 uPSI_OverbyteIcsMimeUtils,
34502: 1332 uPSI_OverbyteIcsUrl(CL: TPSPascalCompiler);
34503: 1333 uPSI_uWebSocket.pas
34504: 1334 uPSI_KhFunction.pas
34505: 1335 uPSI_ALOpenOffice.pas
34506: 1336 unit uPSI_ALLibPhoneNumber
34507: 1337 unit uPSI_ALPhpRunner2;
34508: 1338 unit uPSI_ALWebSpider2;
34509: 1339 unit uPSI_ALFcnHTML2; // RunJavaScript2
34510: 1340 unit uPSI_ALExecute2.pas
34511: 1341 uPSI_ALIsapiHTTP.pas
34512: 1342 uPSI_ALOpenOffice_Routines
34513: 1343 unit uPSI_uUsb;
34514: 1344 uPSI_uWebcam.pas
34515: 1345 uPSI_PersistSettings.pas //fixing & refactoring
34516: 1346 uPSI_uTPLb_MemoryStreamPool.pas
34517: 1347 uPSI_uTPLb_Signatory.pas
34518: 1348 unit uPSI_uTPLb_Constants.pas  //TurboPower
34519: 1349 uPSI_uTPLb_Random.pas
34520: 1350 unit uPSI_uTPLb_PointerArithmetic;
34521: 1351 unit uPSI_EwbCoreTools.pas
34522: 1352 unit uPSI_EwbUrl.pas
34523: 1353 unit uPSI_SendMail_For_Ewb.pas
34524: 1354 unit uPSI_MaskEdit.pas    FCL
34525: 1355 unit uPSI_SimpleRSSTypes; BlueHippo
34526: 1356 unit uPSI_SimpleRSS;      BlueHippo
34527: 1357 unit uPSI_psULib.pas      Prometheus
34528: 1358 unit uPSI_psUFinancial;   Prometheus
34529: 1359 uPSI_PsAPI_2.pas          mX4
34530: 1360 uPSI_PersistSettings_2    mX4
34531: 1361 uPSI_rfc1213util2.pas     IP
34532: 1362 uPSI_JTools.pas           JCL
34533: 1363 unit uPSI_neuralbit.pas   CAI
34534: 1364 unit uPSI_neuralab.pas    CAI
34535: 1365 unit uPSI_winsvc2.pas     TEK
34536: 1366 unit uPSI_wmiserv2.pas    TEK
34537: 1367 uPSI_neuralcache.pas      CAI
34538: 1368 uPSI_neuralbyteprediction CAI
34539: 1369 unit uPSI_USolarSystem;   glscene.org
34540: 1370 uPSI_USearchAnagrams.pas  DFF
34541: 1371 uPSI_JsonsUtilsEx.pas     Randolph
34542: 1372 unit uPSI_Jsons.pas       Randolph
34543: 1373 unit uPSI_HashUnit;       DFF
34544: 1374 uPSI_U_Invertedtext.pas   DFF
34545: 1375 unit uPSI_Bricks;         Dendron
34546: 1376 unit uPSI_lifeblocks.pas  Dendron
34547: 1377 unit uPSI_SystemsDiagram.pas  Dendron
34548: 1378 unit uPSI_qsFoundation.pas    Dendron
34549: 1379 uPSI_JclStringLists2          JCL
34550: 1380 uPSI_cInternetUtils2          FLC
34551: 1381 uPSI_cWindows.pas            FLC
34552: 1382 uPSI_flcSysUtils.pas  +TBytes utils
34553: 1383 unit uPSI_RotImg.pas          DA
34554: 1384 uPSI_SimpleImageLoader.pas    LAZ
34555: 1385 uPSI_HSLUtils.pas             LAZ
34556: 1386 uPSI_GraphicsMathLibrary.pas  EF
34557: 1387 unit uPSI_umodels.pas         DMath
34558: 1388 uPSI_flcStatistics.pas        FLC5
34559: 1389 uPSI_flcMaths.pas             FLC5
34560: 1390 uPSI_flcCharSet.pas           FLC5
```

```
34561: 1391 uPSI_flcBits32.pas
34562: 1392 uPSI_flcTimers.pas
34563: 1393 uPSI_cBlaiseParserLexer.pas
34564: 1394 uPSI_flcRational.pas
34565: 1395 uPSI_flcComplex.pas
34566: 1396 unit uPSI_flcMatrix (uPSI_flcVectors.pas)
34567: 1397 unit uPSI_flcStringBuilder.pas
34568: 1398 unit PJResFile_Routines;
34569: 1399 uPSI_flcASCII.pas
34570: 1400 uPSI_flcStringPatternMatcher;
34571: 1401 unit uPSI_flcUnicodeChar.pas
34572: 1402 unit uXmlDates.pas;
34573: 1403 unit uPSI_SemaphorGrids;
34574: 1404 unit uXmlDates2;
34575: 1405 unit uPSI_JclTimeZones;
34576: 1406 unit uPSI_XmlDocRssParser.pas
34577: 1407 unit uPSI_RssParser.pas
34578: 1408 uPSI_SimpleParserRSS.pas
34579: 1409 unit uPSI_SimpleRSSUtils;
34580: 1410 unit uPSI_RssModel;      _BlueHippo
34581: 1411 unit uPSI_StrUtil;       _FIBPlus
34582: 1412 unit uPSI_TAChartUtils; _TEE
34583: 1413 unit uPSI_PythonEngine.pas _P4D_Beta
34584: 1414 unit uPSI_VclPythonGUIInputOutput;
34585: 1415 unit uPSI_VarPyth;
34586: 1416 unit JclUsesUtils;
34587: 1417 unit uPSI_cParameters;
34588: 1418 unit uPSI_WDCCMisc; (uPSI_cFileTemplates);
34589: 1419 uPSI_WDCCOleVariantEnum.pas
34590: 1420 unit uPSI_WDCCWinInet.pas  _WDCC
34591: 1421 uPSI_PythonVersions.pas    _P4D_
34592: 1422 unit uPSI_PythonAction.pas _P4D_
34593: 1423 uPSI_SingleList.pas
34594: 1424 unit uPSI_AdMeter.pas; Async Professional
34595: 1425 unit uPSI_neuralplanbuilder; CAI
34596: 1426 unit uPSI_neuralvolume.pas; CAI
34597: 1427 unit uPSI_neuralvolumev.pas; CAI
34598: 1428 unit uPSI_DoubleList4;  CapJack
34599: 1429 unit uPSI_ByteListClass;  CapJack
34600: 1430 unit uPSI_flcVectors4; FLC5
34601: 1431 unit uPSI_flcMatrix4;  FLC5
34602: 1432 uPSI_CurlHttpCodes.pas
34603: 1433 unit uPSI_NeuralNetwork.pas; CAI
34604: 1434 unit unit uPSI_neuralfit; CAI
34605: 1435 unit uPSI_neuraldatasets; CAI
34606: 1436 unit uPSI_neuraldatasetsv.pas CAI
34607: 1437 uPSI_flcFloats.pas FLC5
34608: 1438 unit UBigIntsForFloatV4.pas DFF
34609: 1439 unit uPSI_CustApp.pas Pas2js
34610: 1440 unit uPSI_NeuralNetworkCAI.pas CAI
34611: 1441 unit uPSI_neuralgeneric.pas; CAI
34612: 1442 unit uPSI_neuralthread.pas; CAI
34613: 1443 unit uPSI_uSysTools; TuO
34614: 1444 unit upsi_neuralsets; mX4
34615: 1445 unit uPSI_uWinNT.pas mX4
34616: 1446 unit uPSI_URungeKutta4.pas ICS
34617: 1447 unit uPSI_UrlConIcs.pas ICS
34618: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
34619: 1449 unit uPSI_Numedit2 mX4
34620: 1450 unit uPSI_PsAPI_3.pas mX4
34621: 1451 unit uPSI_SeSHA256.pas
34622: 1452 unit IdHashMessageDigest_max2;
34623: 1453 unit uPSI_BlocksUnit.pas
34624: 1454 unit uPSI_DelticsCommandLine.pas
34625: 1455 unit uPSI_DelticsStrUtils;
34626: 1456 unit uPSI_DelticsBitField;
34627: 1457 unit uPSI_DelticsSysUtils;
34628: 1458 unit uPSI_ALIniFiles2.pas
34629: 1459 unit uPSI_StarCalc2.pas
34630: 1460 unit uPSI_IdHashMessageDigest2.pas
34631: 1461 unit uPSI_U_Splines;
34632: 1462 unit uPSI_U_CoasterB.pas;
34633: 1463 U_SpringMass2.pas
34634: 1464 uPSI_MARSCoreUtils;
34635: 1465 unit uPSI_clJsonParser.pas
34636: 1466 unit uPSI_SynHighlighterPython.pas
34637: 1467 unit uPSI_DudsCommonDelphi;
34638: 1468 unit uPSI_AINNNeuron;
34639: 1469 unit uPSI_PJConsoleApp2;  Console Application Runner
34640: 1470 unit uPSI_PJPipeFilters2; CAR
34641: 1471 unit uPSI_uHTMLBuilder;
34642: 1472 unit uPSI_PJPipe2;
34643: 1473 uPSI_WinApiDownload.pas
34644: 1474 uPSI_pxQRcode,    //
34645: 1475 unit uPSI_neuralplanbuilder2
34646: 1476 unit uPSI_DelphiZXingQRCode;
34647: 1477 unit uPSI_RestJsonUtils;
34648: 1478 unit UtilsTimeCode;
34649: 1479 unit uPSC_classes2.pas; //TList
```

```
34650: 1480 unit uPSC_std2.pas
34651: 1481 unit uPSI_TBytes.pas
34652: 1482 unit uPSI_DelticsSysUtils2.pas
34653: 1483 unit uPSI_maxIniFiles.pas
34654: 1484 unit uROPSImports.pas
34655: 1485 unit uROPSServerLink.pas
34656: 1486 unit KLibVC_Redist.pas;
34657: 1487 unit HTTPApp2.pas;
34658: 1488 unit uPSI_XCollection2;
34659: 1489 unit uPSI_KLibWindows;
34660: 1490 unit KlibConstants;
34661: 1491 unit uPSI_AzuliaUtils.pas
34662: 1492 unit uPSI_ALHttpClient2;
34663: 1493 unit uPSI_ALWininetHttpClient2;
34664: 1494 unit uPSI_UtilsMax41.pas
34665: 1495 unit uPSI_JclSysUtils1;
34666: 1496 unit uPSI_RestUtils;
34667: 1497 unit uPSI_TeEngine2.pas
34668: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
34669: 1499 unit uPSI_PSResources.pas
34670: 1500 unit uPSI_TeCanvas2_1.pas
34671: 1501 unit uPSI_DataSetConverter4DUtil;
34672: 1502 unit uPSI_neuralfit2.pas;
34673:
34674: Total of Function Calls: 35958
34675: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
34676: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
34677: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
34678: Entry Point 24783132 - Contained Sections 10
34679:
34680: ******************************************************************
34681: Release Notes maXbox 4.7.6.20 Oct 2022 mX476
34682: ******************************************************************
34683: Add 60 Units + 18 Tutorials
34684:
34685: 1441 unit uPSI_neuralgeneric.pas; CAI
34686: 1442 unit uPSI_neuralthread.pas; CAI
34687: 1443 unit uPSI_uSysTools; TuO
34688: 1444 unit upsi_neuralsets; mX4
34689: 1445 unit uPSI_uWinNT.pas mX4
34690: 1446 unit uPSI_URungeKutta4.pas ICS
34691: 1447 unit uPSI_UrlConIcs.pas ICS
34692: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
34693: 1449 unit uPSI_Numedit2 mX4
34694: 1450 unit uPSI_PsAPI_3.pas mX4
34695: 1451 unit uPSI_SeSHA256.pas
34696: 1452 unit IdHashMessageDigest_max2;
34697: 1453 unit uPSI_BlocksUnit.pas
34698: 1454 unit uPSI_DelticsCommandLine.pas
34699: 1455 unit uPSI_DelticsStrUtils;
34700: 1456 unit uPSI_DelticsBitField;
34701: 1457 unit uPSI_DelticsSysUtils;
34702: 1458 unit uPSI_ALIniFiles2.pas
34703: 1459 unit uPSI_StarCalc2.pas
34704: 1460 unit uPSI_IdHashMessageDigest2.pas
34705: 1461 unit uPSI_U_Splines;
34706: 1462 unit uPSI_U_CoasterB.pas;
34707: 1463 U_SpringMass2.pas
34708: 1464 uPSI_MARSCoreUtils;
34709: 1465 unit uPSI_clJsonParser.pas
34710: 1466 unit uPSI_SynHighlighterPython.pas
34711: 1467 unit uPSI_DudsCommonDelphi;
34712: 1468 unit uPSI_AINNNeuron;
34713: 1469 unit uPSI_PJConsoleApp2;
34714: 1470 unit uPSI_PJPipeFilters2;
34715: 1471 unit uPSI_uHTMLBuilder;
34716: 1472 unit uPSI_PJPipe2;
34717: 1473 uPSI_WinApiDownload,
34718: 1474 uPSI_pxQRcode,    //beta
34719: 1475 unit uPSI_neuralplanbuilder2
34720: 1476 unit uPSI_DelphiZXingQRCode;
34721: 1477 unit uPSI_RestJsonUtils;
34722: 1478 unit UtilsTimeCode;
34723: 1479 unit uPSC_classes2.pas; //TList
34724: 1480 unit uPSC_std2.pas
34725: 1481 unit uPSI_maxIniFiles.pas
34726: 1482 unit uROPSImports.pas
34727: 1483 unit uROPSServerLink.pas
34728: 1484 unit uPSI_KLibUtils;
34729: 1485 unit uPSI_PathFunc2;  //inno setup
34730: 1486 unit KLibVC_Redist.pas;
34731: 1487 unit HTTPApp2.pas;
34732: 1488 unit uPSI_XCollection2;
34733: 1489 unit uPSI_KLibWindows;
34734: 1490 unit KlibConstants;
34735: 1491 unit uPSI_AzuliaUtils.pas
34736: 1492 unit uPSI_ALHttpClient2;
34737: 1493 unit uPSI_ALWininetHttpClient2;
34738: 1494 unit uPSI_UtilsMax41.pas
```

```
34739: 1495 unit uPSI_JclSysUtils1;
34740: 1496 unit uPSI_RestUtils;
34741: 1497 unit uPSI_TeEngine2.pas
34742: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
34743: 1499 unit uPSI_PSResources.pas
34744: 1500 unit uPSI_TeCanvas2_1.pas
34745: 1501 unit uPSI_DataSetConverter4DUtil
34746:
34747: Total of Function Calls: 35952
34748: SHA1: 4.7.6.20 C0459A5DD0CCC12CE9F3CDEC82E3E0C920D38D9C
34749: CRC32: 498B62AF 32.1 MB (33,690,392 bytes)
34750: Compilation Timestamp 2022-10-09 17:16:23 UTC
34751: Entry Point 24766748 - Contained Sections 10
34752:
34753: ****************************************************************
34754: Release Notes maXbox 4.7.6.10 November 2021 mX476
34755: ****************************************************************
34756: Add 5 Units + 2 Tutorials
34757:
34758: 1441 unit uPSI_neuralgeneric.pas; CAI
34759: 1442 unit uPSI_neuralthread.pas; CAI
34760: 1443 unit uPSI_uSysTools; TuO
34761: 1444 unit upsi_neuralsets; mX4
34762: 1445 unit uPSI_uWinNT.pas mX4
34763: Total of Function Calls: 34880
34764: SHA1: of 4.7.6.10  CF939E3A8D4723DB1DEF383C5FC961E06728C58F
34765: CRC32: 38F88218 30.5 MB (32,022,344 bytes)
34766: V 4.7.5.90 VI
34767: Amount of Functions: 20549
34768: Amount of Procedures: 12307
34769: Amount of Constructors: 1937
34770: Amount of Destructors: 14
34771: Totals of Calls: 34807
34772: SHA1: of 4.7.5.90 96DCDE2028125E00B67E42A801721AC513A5EAFC
34773: CRC32: BBC3A7E5:  31974216 bytes
34774: ▯▯▯ mX4 executed: 03/11/2021 15:25:47  Runtime: 0:12:33.807  Memload: 39% use
34775: Amount of Functions: 20994
34776: Amount of Procedures: 12492
34777: Amount of Constructors: 1984
34778: Amount of Destructors: 14
34779: Totals of Calls: 35484
34780: SHA1: of 4.7.6.10 EC20D0A371A57AA078762D6D45F7229A370549A1
34781: CRC32: 82E481A4:  33273160 bytes
34782: ▯▯▯ mX4 executed: 14/03/2022 15:21:38  Runtime: 0:13:41.253  Memload: 42% use
34783:
34784: // Project:     State Machine
34785: // Module:      statmach.pas
34786: // Description: Visual Finite State Machine.
34787: // Version:     2.2a  - Release:  6
34788:
34789: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
34790: http://www.slideshare.net/maxkleiner1/codereview-topics
34791: ////////////////////////////////////////////////////////////////////////////
34792: //Form Template Library FTL
34793: ////////////////////////////////////////////////////////////////////////////
34794:
34795: FTL For Form Building Lib out of the Script, eg. 399_form_templates.txt
34796: 045 unit uPSI_VListView                        TFormListView;
34797: 263 unit uPSI_JvProfiler32;                    TProfReport
34798: 270 unit uPSI_ImgList;                         TCustomImageList
34799: 278 unit uPSI_JvImageWindow;                  TJvImageWindow
34800: 317 unit uPSI_JvParserForm;                   TJvHTMLParserForm
34801: 497 unit uPSI_DebugBox;                       TDebugBox
34802: 513 unit uPSI_ImageWin;                       TImageForm, TImageForm2
34803: 514 unit uPSI_CustomDrawTreeView;             TCustomDrawForm
34804: 515 unit uPSI_GraphWin;                       TGraphWinForm
34805: 516 unit uPSI_actionMain;                     TActionForm
34806: 518 unit uPSI_CtlPanel;                       TAppletApplication
34807: 529 unit uPSI_MDIEdit;                        TEditForm  //RichEditApp
34808: 535 unit uPSI_CoolMain; {browser}             TWebMainForm
34809: 538 unit uPSI_frmExportMain;                  TSynexportForm
34810: 585 unit uPSI_usniffer; {//PortScanForm}      TSniffForm
34811: 600 unit uPSI_ThreadForm;                     TThreadSortForm;
34812: 618 unit uPSI_delphi_arduino_Unit1;           TLEDForm
34813: 620 unit uPSI_fplotMain;                      TfplotForm1
34814: 660 unit uPSI_JvDBQueryParamsForm;            TJvQueryParamsDialog
34815: 677 unit uPSI_OptionsFrm;                     TfrmOptions;
34816: 718 unit uPSI_MonForm;                        TMonitorForm
34817: 742 unit uPSI_SimplePortMain;                 TPortForm1
34818: 770 unit uPSI_ovccalc;                        TOvcCalculator  //widget
34819: 810 unit uPSI_DbExcept;                       TDbEngineErrorDlg
34820: 812 unit uPSI_GL_actorUnit1;                  TglActorForm1  //OpenGL Robot
34821: 846 unit uPSI_synhttp_daemon;                 TTCPHttpDaemon, TTCPHttpThrd, TPingThread
34822: 867 unit uPSI_Debug;                          TfrmDebug
34823: 901 unit uPSI_JvAddPrinter                    TJvAddPrinter     //JCL
34824: 904 unit uPSI_U_HexView;                      THexForm2
34825: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)   TOscfrmMain
34826: 959 unit uPSI_AdSelCom;                       TComSelectForm
34827: 1029 unit uPSI_GPSUDemo;                      TFDemo
```

```
34828: 1031 unit uPSI_ScreenThreeDLab;                TFormLab3D
34829: 1051 unit uPSI_panUnit1;                       TPanForm1 //GLScene
34830: 1052 unit uPSI_DD83u1;  {Arduino Tester Frm}   TDD83f1
34831: 1059 unit uPSI_uconvMain;                      TfconvMain //PS
34832: 1076 unit uPSI_CustomizeDlg;                   TCustomizeDlg / TJvAddPrinterDialog;
34833: 1111 unit uPSI_JvExceptionForm;                TJvErrorDialog  //ShowException
34834: 1113 unit uPSI_MTMainForm;                     TvtMainForm
34835: 1119 unit uPSI_CPortTrmSet;                    TComTrmSetForm
34836: 1146 unit uPSI_ChessPrg;                       TChessForm1
34837: 1216 uPSI_KDialogs,                            TKBrowseFolderDialog
34838: 1217 eg.execStr('from delphivcl import *');    LoadPyClass
34839:
34840: FormTemplates with <Ctrl> J
34841:
34842: [myformtemplate | formtemplate statement | Borland.EditOptions.Pascal] //with
34843: [myFastForm | class declaration  | Borland.EditOptions.Pascal]  //Dialog Form
34844: [myForm | class declaration  | Borland.EditOptions.Pascal]   //with Events
34845: [aForm | class declaration  | Borland.EditOptions.Pascal]   //single Form
34846: [getForm(x,y)] of object TForm [loadForm(x,y)]
34847: [ticker | ticker statement | Borland.EditOptions.Pascal]
34848: [except | exception statement | Borland.EditOptions.Pascal]
34849:
34850: Dark Color Style Darkcolor:
34851: memo1.color:= RGB2TColor(32,32,32)  //20,18,18
34852: memo2.color:= RGB2TColor(20,18,18)
34853:
34854: Proc SIRegister_JvDesignUtils(CL: TPSPascalCompiler);
34855: begin
34856:  Func DesignClientToParent(const APt: TPoint; AControl,AParent: TControl) : TPoint;
34857:  Func DesignMin(AA, AB : Int):Int);
34858:  Func DesignMax(AA, AB: Int): Int;
34859:  Func DesignRectWidth( const ARect : TRect) : Int;
34860:  Func DesignRectHeight( const ARect : TRect) : Int;
34861:  Func DesignValidateRect( const ARect : TRect) : TRect;
34862:  Func DesignNameIsUnique( AOwner : TComponent; const AName :Str) :Bool;
34863:  Func DesignUniqueName( AOwner : TComponent; const AClassName :Str) :Str;
34864:  Proc DesignPaintRubberbandRect(AContainer:TWinControl;ARect:TRect;APenStyle:TPenStyle);
34865:  Proc DesignPaintGrid(ACanvs:TCanvas;const ARect:TRect;ABackClr:TColr;AGridClr:TColor;ADivPixels:Int)
34866:  Proc DesignPaintRules(ACanvas:TCanvas;const ARect:TRect;ADivPixels:Int;ASubDivs:Bool;
34867:  Proc DesignSaveComponentToStream( AComp : TComponent; AStream : TStream);
34868:  Func DesignLoadComponentFromStream(AComp:TComponent;AStram:TStream;AOnError:TReaderError):TComponent;
34869:  Proc DesignSaveComponentToFile( AComp : TComponent; const AFileName :Str);
34870:  Proc DesignLoadComponentFromFile(AComp:TComp;const AFileName:str;AOnError:TReaderError);
34871: end;
34872:
34873: ex.:with TEditForm.create(self) do begin
34874:       caption:= 'Template Form Tester';
34875:       FormStyle:= fsStayOnTop;
34876:       with editor do begin
34877:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
34878:         SelStart:= 0; Modified:= False;
34879:       end;
34880:     end;
34881:   with TWebMainForm.create(self) do begin
34882:     URLs.Text:= 'http://www.kleiner.ch';
34883:     URLsClick(self); Show;
34884:   end;
34885:   with TSynexportForm.create(self) do begin
34886:     Caption:= 'Synexport HTML RTF tester';
34887:     Show;
34888:   end;
34889:   with TThreadSortForm.create(self) do begin
34890:      showmodal; free;
34891:   end;
34892:   with TCustomDrawForm.create(self) do begin
34893:        width:=820; height:=820;
34894:        image1.height:= 600; //add properties
34895:        image1.picture.bitmap:= image2.picture.bitmap;
34896:        //SelectionBackgroundlClick(self) CustomDrawlClick(self);
34897:        Background1.click;
34898:        bitmap1.click; Tile1.click;
34899:        Showmodal;
34900:        Free;
34901:      end;
34902:    with TfplotForm1.Create(self) do begin
34903:      BtnPlotClick(self);
34904:      Showmodal; Free;
34905:    end;
34906:   with TOvcCalculator.create(self) do begin
34907:     parent:= aForm;  //free;
34908:     setbounds(550,510,200,150);
34909:     displaystr:= 'maXcalc';
34910:   end;
34911:   with THexForm2.Create(self) do begin
34912:      ShowModal;
34913:      Free;
34914:   end;
34915:
34916: Func CheckBox:Str;
```

```
var idHTTP: TIDHTTP;
 begin
   result:= 'version not found';
   if IsInternet then begin
     idHTTP:= TIdHTTP.Create(NIL);
     try
       result:= idHTTP.Get(MXVERSIONFILE2);
       result:= result[1]+result[2]+result[3]+result[4]+result[5];
       if result = MBVER2 then begin
         //Speak(' A new Version '+vstr+' of max box is available! ;
         result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
       end;  //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
     finally
       idHTTP.Free
   end;
 end;
end;

charinset make your own charinset to prevent typematch:

Func CharInSet4(C: AnsiChar; const CharSet: TSysCharSet):Bool;
begin
  Result:= C in CharSet;
end;

Func CharInSet5(C: AnsiChar; const CharSet: CharSet):Bool;
begin
  Result:= C in CharSet;
end;

//Runtimer Spec Functions Edition 190 and more code blocks
Func ApWinExecAndWait32(FileName:PChar;CommandLine:PChar; Visibility:Int):Int;
Func KillTask(ExeFileName:Str): Int;
Proc KillProcess(hWindowHandle: HWND);
Func FindWindowByTitle(WindowTitle:Str): Hwnd;
Func OpenIE(aURL:Str):Bool;
Func XRTLIsInMainThread:Bool;
Func IsInMainThread:Bool;
Func IntToFloat(i: Int): double;
Func AddThousandSeparator(S:Str; myChr: Char):Str;
Func mciSendString(cmd: PChar; ret: PChar; len: Int; callback: Int):Card;
Proc FormAnimation(Sender: TObject; adelay: Int);
Proc LoadResourceFile2(aFile:str; ms:TMemoryStream);
Func putBinResTo(binresname: pchar; newpath:Str):Bool;
Proc ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:str);
Func IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:str;MouseX,MouseY:Int;var HyperLink:str):Bool
Func GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD) : DWORD;;
Func GetWindowTask( hWnd : HWND) : THandle;
Func LoadBitmap( hInstance : HINST; lpBitmapName : PChar) : HBITMAP;
Func GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL;
Func WinExecAndWait32(FileName:Str; Visibility: Int): Longword;
Func MakeHash( const s : TbtString) : Longint;
Func GetUsedUnitList( list : Tstringlist) :Str;
Func ConsoleCapture(const _dirName, _exeName, _cmdLine:Str; amemo: TStringlist): Bool;
Func ConsoleCaptureDOS(const _dirName, _exeName, _cmdLine:Str; amemo: TStrings): Bool;
         srlist:= TStringlist.create;
         ConsoleCapture('C:\', 'cmd.exe', '/c dir *.*',srlist);
         writeln(srlist.text); srlist.Free;
Func RunCaptured(const _dirName, _exeName,_cmdLine:str; amemo:TStringlist):Bool;;
Func SamePropTypeName( const Name1, Name2 : ShortString) :Bool;
Func FloatToStrEx( Value : Extended) :Str;
Func StrToFloatEx( const S :Str) : Extended;
Proc PerformanceDelayMS(ams: Int);  //microsecond resolution delay!
//http://www.swissdelphicenter.ch/en/showcode.php?id=2179
Func ExecuteProcess(FileName:Str;Visibility:Int; BitMask:Int; Synch:Boolean):Longword;
Func ExecuteMultiProcessor(FileName:str;Visibility:Int;BitMask:Int;Synch:Bool):Longword;
      if ExecuteMultiProcessor('notepad.exe', SW_SHOW, 2, true) = 0 then
                                writeln('Multiprocessing Runs on CPU 2;
Proc StartServiceAfterInstall(aserv: TService);
Func GetDllVersion2(DllName:Str; var DLLVersionInfo: TDLLVersionInfo):Bool;
Proc SendCopyMessage(amess,astation:str); //comunicate process-spanned with WM_COPYDATA
Func BrowseComputer2(DialogTitle:Str; var CompName:Str; bNewStyle: Bool): Bool;
Func ChangeAlphaTo(input:Str; aoffset: byte):Str;;
Func CheckIBAN(iban:Str):Bool;;
Func CreateIDStack; //instance to IdStack of CreateStack
Func GetRecordCount(DataSet: TBDEDataSet): Longint;;
Func CountPos(const subtxt:Str; Text:Str): Int;
Proc HTMLtoRTF(html:Str; var rtf: TRichedit);
Proc ReversePlay(const szFileName:Str);
Func ADOConnectionString(ParentHandle:THandle;InitialStr:WideString;out NewString:str):Bool;
Proc ShowEOleException(AExc: EOleException; Query:str);
Func UpdateBlob(Connection:TADOConnection;Spalte:str;Tabelle:str;Where:str;var
ms:TMemoryStream):Bool;..save HTML pages as MHTML(HTML Archiv Format)
   http://www.swissdelphicenter.ch/en/showcode.php?id=2300
Func SaveToMHT(const AUrl,AFileName:Str; AShowErrorMessage:Boolean = False):Bool;
Func FileType2MimeType(const AFileName:Str):Str;
Func DownloadURL_NOCache(const aUrl:Str; var s:Str):Bool;
Func IsCOMObjectActive(ClassName:Str):Bool;
Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str = ';
```

```
35005: Func CheckCreditCard(c:Str): Int;
35006:    0: Card is invalid or unknown 1: is a AmEx 2: is a Visa 3: is a valid MasterCard
35007: Func GetNewGUID:Str;
35008: Func FormatGUID(const GUID:Str):Str;
35009: Func GetNewFormatedGUID:Str;
35010: Func getFormRes(classname:Str):Str;  //shows DFM Res of Exe!
35011: Proc OutputDebugString(PChar(Format('[%s][%s]%s',[aCaption,GetFormatDT(StartDT),aText])));
35012: Proc ScanNetworkResources(ResourceType, DisplayType: DWord; List: TStrings);
35013: Func PrepareConstraint(Src:Tstrings):str;
35014: Proc DeleteEmptyStr(Src:Tstrings);
35015: Func NormalizeSQLText(const SQL:Str;MacroChar:Char):Str;
35016: Func CountSelect(const SrcSQL:str):str;
35017: Func GetModifyTable(const SQLText:str;AlreadyNormal:boolean):str;
35018: Func GetCharFromVKey(vkey: Word):Str;
35019: Func Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile:Str):Bool;
35020: Func IsObjectActive(ClassName:Str):Bool;
35021:  Func GetActiveObject(ClassID:TGUID; anil:TObject; aUnknown:IUnknown):HRESULT;;
35022: Func RegisterOCX(FileName:Str):Bool;
35023: Func UnRegisterOCX(FileName:Str):Bool;
35024: Func RegisterServer2(const aDllFileName:Str; aRegister:Bool):Bool;
35025: Proc mIRCDDE(Service, Topic, Cmd:Str); //mIRCDDE('mIRC', 'COMMAND', '/say Hallo von SwissDelphiCenter.ch;
35026: Func OpenIE(aURL:Str):Bool;
35027: Func XRTLIsInMainThread:Bool;
35028: Func IsInMainThread:Bool;
35029: TryConvertStrToDateTime(const s, format:str; out value: TDateTime):Boolean;;
35030:   ConvertStrToDateTime(const s, format:Str): TDateTime;;
35031:  Func CreateNotifyThread2(const FolderName:Str; WatchSubtree:Bool; Filter : TFileChangeFilters2):
       TNotifyThread;
35032:  Proc DetectImage(const InputFileName:Str; BM: TBitmap);
35033:  Func BitmapToString(Bitmap: TBitmap):Str;
35034:  Func StringToBitmap(S:Str): TBitmap;
35035:  Func RemoveChar(CONST s:Str; CONST c: CHAR):Str;
35036:  Proc SecureClearStr(var S: Ansistr);;
35037:  Proc movestring(const Source:str; var Destination:str; CopyCount : Int );
35038:  Proc movestringJV(const Source:str; var Destination:str; CopyCount : Int );
35039:  Proc moveint(const Source:Int; var destination: Int; CopyCount : Int );
35040:  Proc movefloat(const Source:double; var destination: double; CopyCount : Int );
35041:  Proc moveextended(const Source:extended;var Destination:extended;CopyCount:Int);
35042:  Proc ShowFilePropertiesSH(Files: TStrings; aWnd: HWND);
35043: //ScanNetworkResources(RESOURCETYPE_DISK, RESOURCEDISPLAYTYPE_SERVER, ListBox1.Items);
35044: Func GrabLine(const s:Str; ALine: Int):Str;
35045: Func GrabLineFast(const s:Str; ALine: Int):Str;
35046: Func IsTextFile(const sFile: TFileName):Bool;
35047: Func getODBC: Tstringlist;
35048:  Func getODBCString:Str;
35049:  Proc GetJPGSize(const sFile:Str; var wWidth, wHeight: Word);
35050:  Proc GetPNGSize(const sFile:Str; var wWidth, wHeight: Word);
35051:  function GetPNGSize2(const FileName: string): TSize;
35052:  Proc GetGIFSize(const sGIFFile:Str; var wWidth, wHeight: Word);
35053: // note: using TForm's BorderIcons, etc. is slow (form blinks) and not reliable (for some
35054: //   reson it causes TListView.Items to lose all associated objects and other things happen).
35055: Proc ChangeWindowStyle(const Form: HWND; Style: DWord; AddIt:Bool);
35056: Func VarByteArrayOf(const s:Str): OleVariant;
35057: Func bigdiv2(aval1:Str; aval2: integer):Str;
35058: Func modbig(aval:Str; amod: integer): integer;
35059: Func bigmod(aval:Str; amod: integer): integer;
35060: Func modPowBig3(aval, apow, amod:Str):Str;;
35061: Func BigPowMod(aval, apow, amod:Str):Str;;
35062: Func RSAEncrypt(aval, apow, amod:Str):Str;;
35063: Func RSADecrypt(aval, apow, amod:Str):Str;;
35064: Func SwitchToThread : BOOL;
35065: Func SetThreadDesktop( hDesktop : HDESK) : BOOL;
35066: Func CloseDesktop( hDesktop : HDESK) : BOOL;
35067: Func GetThreadDesktop( dwThreadId : DWORD) : HDESK;
35068: Func SetSyscallHook():Bool;;
35069: Func SetSwapcontextHook():Bool;;
35070: Func UnhookAll():Bool;;
35071: Func getWorld:Str;;
35072: Func getIPConfigAll:Str;;
35073: Func getIPConfig:Str;;
35074: Func WinsockEnabled:Bool;
35075: Func HTTPEncode2(const AStr:Str):Str;
35076: Func ComputePEChecksum(FileName:Str): DWORD;
35077:
35078: if not DynamicDllCallName(user32, 'LockWorkStation',true,returned,parameters) then begin
35079: Func DynamicDllCallNames(Dll:Str; const Name:Str; HasResult:Bool; var Returned:Card; const Parameters:
       array of string):Bool;;
35080: Proc GetMIDASAppServerList(List: TStringList; const RegCheck :Str);
35081: Proc SQLDropField(dbName, tblName,  fldName:Str);  {Field Name to Drop}
35082: type TCastType = (ctSmallInt, ctInt, ctDecimal, ctNumeric, ctFloat,
35083:                ctChar, ctVarChar, ctDate, ctBoolean, ctBLOB, ctTime,
35084:                ctTimeStamp, ctMoney, ctAutoInc, ctBytes);
35085: {Blob definition type 1 = Memo, 2 = Binary, 3 = Formatted Memo, 4 = OLE Object, 5 = Graphic}
35086:
35087: Proc SQLAddField(dbName,tblName,fldName:str;fldType:TCastType;fldLength,precisOrBlobLen,
       scaleOrBlobType:Int);
35088: const
35089:  UrlGeoLookupInfo  ='http://ipinfodb.com/ip_query.php?timezone=true&ip=%s';
35090:  UrlGeoLookupInfo2 ='http://backup.ipinfodb.com/ip_query.php?timezone=true&ip=%s'; //backup
```

```
35091:  http://ip-api.com/csv/
35092:
35093: Proc GetGeoInfo(const IpAddress:str;var GeoInfo:TGeoInfo;const UrlGeoLookupInfo:str);
35094:  TGeoInfo','record status:Str; countrycode : '
35095:    +string; countryname:str; regioncode:Str; city:Str; zippostalcode:Str;
       latitude:Str;longitude:str;timezonename:str;gmtoffset:str;isdst:str; end;
35096:
35097: Proc SIRegister_ubigFloatV3(CL: TPSPascalCompiler);
35098: begin
35099:   TMaxSig', 'Int;
35100:   TView', '( normal, Scientific );
35101:   SIRegister_TFloatInt(CL); SIRegister_TBigFloat(CL);
35102: end;
35103:
35104: Proc SIRegister_UBigIntsV4(CL: TPSPascalCompiler);
35105: begin TDigits', 'array of int64; SIRegister_TInt(CL);
35106:  Proc SetBaseVal( const newbase : Int);
35107:  Func GetBasePower : Int;
35108:  Func GetBase : Int;
35109:  Proc SetThreadSafe( newval :Bool);
35110:  Func bigdiv2(aval1:Str; aval2: integer):Str;
35111:  Func modbig(aval:Str; amod: integer): integer; ;
35112:  Func bigmod(aval:Str; amod: integer): integer; ;
35113:  Func modPowBig3(aval, apow, amod:Str):Str;;
35114:  Func BigPowMod(aval, apow, amod:Str):Str;;
35115:  Func RSAEncrypt(aval, apow, amod:Str):Str;;
35116:  Func RSADecrypt(aval, apow, amod:Str):Str;;
35117:  Func SwitchToThread : BOOL;
35118: end;
35119:
35120: Func BigDiv(aone, atwo:Str):Str;
35121: var tbig1, tbig2: TInt;
35122: begin
35123:   tbig1:= TInt.create(10);
35124:   tbig2:= TInt.create(10);
35125:   try
35126:     tbig1.assign2(atwo)
35127:     tbig2.assign2(aone)
35128:     tbig2.Divide(tbig1)
35129:   finally
35130:     result:= tbig2.ConvertToDecimalString(false)
35131:     tbig1.Free;
35132:     tbig2.free;
35133:   end;
35134: end;
35135:
35136: Proc SIRegister_UDict(CL: TPSPascalCompiler);
35137: begin 'dichighletter','String').SetString( 'z;
35138:   SIRegister_TDicForm(CL);
35139:   SIRegister_TDic(CL);
35140: end;
35141:
35142: Proc SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: Int);
35143: var i: Int;
35144: begin setlength(arr, asize1);
35145:    for i:= 0 to asize1-1 do SetLength(arr[i], asize2);
35146: end;
35147:
35148: Example of dynamic array of panels:
35149: Proc SetArrayLength2Panels(var arr: array of array of TPanel;
35150:                                          asize1, asize2: Integer);
35151: var i: Integer;
35152: begin setlength(arr, asize1);
35153:    for i:= 0 to asize1-1 do SetLength(arr[i], asize2);
35154: end;
35155:
35156: Proc TMyFormInitialisePanels(aform: Tform; RowCount,ColCount: Integer);
35157: var aLeft,aTop,aWidth,aHeight, row,col: Integer;
35158:   Panel: TPanel;
35159:   FPanels: array of array of TPanel;
35160: begin
35161:   //SetLength(FPanels, RowCount, ColCount);
35162:   SetArrayLength2Panels(Fpanels, RowCount, ColCount)
35163:   aTop:= 0;
35164:   for Row:= 0 to RowCount-1 do begin
35165:     aLeft:= 0;
35166:     aHeight:= (aform.ClientHeight-aTop) div (RowCount-Row);
35167:     for Col:= 0 to ColCount-1 do begin
35168:       Panel:= TPanel.Create(Self);
35169:       FPanels[Row][Col]:= Panel;
35170:       Panel.Parent:= aform; //Self;
35171:       //panel.parentcolor:= false;
35172:       panel.ParentBackground:= false;
35173:       panel.color:= random(clred)
35174:       aWidth:= (aform.ClientWidth-aLeft) div (ColCount-Col);
35175:       Panel.SetBounds(aLeft, aTop, aWidth, aHeight);
35176:       inc2(aLeft, aWidth);
35177:     end;
35178:     inc2(aTop, aHeight);
```

```
35179:   end;
35180: end;
35181:   aform:= TForm.create(self)
35182:   aform.setbounds(10,10,600,500);
35183:   TMyFormInitialisePanels(aform, 26, 19);
35184:   aform.show;
35185:
35186: Proc SIRegister_UP10Build(CL: TPSPascalCompiler);
35187: begin Proc ParseFunction(FunctionString:str; Variables:TStringlist; FunctionOne,
       FunctionTwo:TStringList;UsePascalNumbers:bool;var FirstOP:{TObject}POperation;var Error:bool);
35188: end;
35189:
35190: Proc SIRegister_ComObj2(CL: TPSPascalCompiler);
35191: begin
35192:  ClassN(CL.FindClass('TOBJECT'),'TComObjectFactory;
35193:  SIRegister_TComServerObject(CL);
35194:  SIRegister_ADOConst(CL);
35195:   FieldTypeNames: array[0..41] of string = (
35196:     Unknown', 'String', 'SmallInt', 'Int', 'Word', 'Boolean', 'Float',
35197:     'Currency', 'BCD', 'Date', 'Time', 'DateTime', 'Bytes', 'VarBytes',
35198:     'AutoInc', 'Blob', 'Memo', 'Graphic', 'FmtMemo', 'ParadoxOle',
35199:     'dBaseOle', 'TypedBinary', 'Cursor', 'FixedChar', 'WideString',
35200:     'LargeInt', 'ADT', 'Array', 'Reference', 'DataSet', 'HugeBlob', 'HugeClob',
35201:     'Variant', 'Interface', 'Dispatch', 'Guid', 'SQLTimeStamp', 'FMTBcdField',
35202:   FixedWideChar', 'WideMemo', 'SQLTimeStamp', 'String;
35203:   TFactoryProc', 'Proc (Factory : TComObjectFactory);
35204:   TCallingConvention','(ccRegister, ccCdecl, ccPascal, ccStdCall, ccSafeCall);
35205:    SIRegister_TComClassManager(CL);
35206:    SIRegister_IServerExceptionHandler(CL); SIRegister_TComObject(CL);
35207:   //TComClass', 'class of TComObject;
35208:   TClassInstancing', '(ciInternal,ciSingleInstance,ciMultiInstance );
35209:   TThreadingModel', '(tmSingle, tmApartment, tmFree, tmBoth, tmNeutral );
35210:   SIRegister_TComObjectFactory(CL); SIRegister_TTypedComObject(CL);
35211:   //TTypedComClass', 'class of TTypedComObject;
35212:   SIRegister_TTypedComObjectFactory(CL);
35213:   TConnectEvent2','Procedure(const Sink : IUnknown; Connecting :Bool);
35214:   ClassN(CL.FindClass('TOBJECT'),'TAutoObjectFactory;
35215:   SIRegister_TAutoObject(CL); //TAutoObject2  ?? in OleAuto and ComObj
35216:   //TAutoClass', 'class of TAutoObject;
35217:   SIRegister_TAutoObjectFactory(CL); SIRegister_TAutoIntfObject(CL);
35218:   //ClassN(CL.FindClass('TOBJECT'),'EOleError;
35219:   ClassN(CL.FindClass('Exception'),'EOleError;
35220:   SIRegister_EOleSysError(CL); SIRegister_EOleException(CL);
35221:   ClassN(CL.FindClass('TOBJECT'),'EOleRegistrationError;
35222:  //Proc DispatchInvoke(const
       Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Result: PVariant);
35223:  //Proc DispatchInvokeError( Status : Int; const ExcepInfo : TExcepInfo);
35224:  //Func HandleSafeCallException(ExceptObject:TObject;ExceptAddr:Pointer;const ErrorIID  TGUID;const ProgID,
       HelpFileName:WideString):HResult;
35225:  Func CreateComObject( const ClassID : TGUID) : IUnknown;
35226:  Func CreateRemoteComObject(const MachineName: WideString;const ClassID:TGUID):IUnknown;
35227:  //Func CreateOleObject(const ClassName :Str): IDispatch;
35228:  //Func GetActiveOleObject(const ClassName:str): IDispatch;
35229:  Proc OleError2( ErrorCode : HResult);
35230:  Proc OleCheck2( Result : HResult);
35231:  Func StringToGUID2( const S :Str) : TGUID;
35232:  Func GUIDToString2( const ClassID : TGUID) :Str;
35233:  Func ProgIDToClassID2( const ProgID :Str) : TGUID;
35234:  Func ClassIDToProgID2( const ClassID : TGUID) :Str;
35235:  Proc CreateRegKey(const Key,ValueName,Value:str;RootKey:DWord);
35236:  Proc DeleteRegKey(const Key :Str; RootKey : DWord);
35237:  Func GetRegStringValue( const Key, ValueName :Str; RootKey : DWord) :Str;
35238:  //Func StringToLPOLESTR( const Source :Str) : POleStr;
35239:  Proc RegisterComServer( const DLLName :Str);
35240:  Proc RegisterAsService( const ClassID, ServiceName :Str);
35241:  Func CreateClassID2:Str;
35242: Proc InterfaceConnect(const Srce:IUnknown;const IID:TIID;const Sink:IUnknown;var Connectin:Longint);
35243:  Proc InterfaceDisconnect(const Source:IUnknown;const IID:TIID;var Connection:Longint);
35244:  Func GetDispatchPropValue(Disp: IDispatch; DispID : Int) : OleVariant;;
35245:  Func GetDispatchPropValue1(Disp: IDispatch; Name : WideString) : OleVariant;;
35246:  Proc SetDispatchPropValue2(Disp:IDispatch;DispID:Int;const Value:OleVariant);
35247:  Proc SetDispatchPropValue3(Disp: IDispatch; Name:WideString; const Value:OleVariant);;
35248:  Func ComClassManager : TComClassManager;
35249: //  from ADODB OLE Utils
35250:   TOleEnum', 'LongWord; //DataTypeEnum = TOleEnum;
35251:   DataTypeEnum', 'TOleEnum;
35252:  Func CreateADOObject( const ClassID : TGUID) : IUnknown;
35253:  Func ADOTypeToFieldType(const ADOType:DataTypeEnum; EnableBCD:Boolean):TFieldType;
35254:  Func FieldTypeToADOType(const FieldType: TFieldType):DataTypeEnum;
35255:  Func StringToVarArray( const Value :Str) : OleVariant;
35256:  Func VarDataSize( const Value : OleVariant) : Int;
35257:  Func OleEnumToOrd( OleEnumArray : array of TOleEnum; Value : TOleEnum) : Int;
35258:  Func GetStates( State : Int) : TObjectStates;
35259:  Func ExecuteOptionsToOrd(ExecuteOptions:TExecuteOptions): Int;
35260:  Func OrdToExecuteOptions(Options: Int) : TExecuteOptions;
35261:  Func ExtractFieldName( const Fields : WideString; var Pos : Int) : WideString;
35262:  Func GetFilterStr( Field : TField; Value : Variant; Partial :Bool) : WideString;
35263:  Func FieldListCheckSum( DataSet : TDataset) : Int;
35264:  Func GlobalAllocString(s: Ansistr): HGlobal;
```

```
35265:  Func ScanTime(const S:Str; var Pos: Int; var Time: TDateTime):Bool;
35266:  Func ScanChar(const S:Str; var Pos: Int; Ch: Char):Bool;
35267:  Func ScanNumber(const S:Str; var Pos: Int; var Number: Word):Bool;
35268:  Func ScanString(const S:Str; var Pos: Int; const Symbol:Str):Bool;
35269:  Proc LV_InsertFiles(strPath:Str; ListView: TListView; ImageList: TImageList);
35270:  Func GetPasteLinkInfo(var Service:Str; var Topic:str; var Item:Str):Boolean;
35271:  Func IPToHostName(const IP:Str):Str;
35272:  Proc GetZoneIcon(IconPath:Str; var Icon: TIcon);
35273:  Func GetZoneAttributes(const URL:Str): TZoneAttributes;
35274:  //unit uPSI_PsAPI;
35275:  Proc CGITester;
35276:  //CGI will take name and email address from command line and place it into HTML
35277:  Proc CreateBrowserOnForm(aform: TForm; aurl:Str);
35278:  Proc WebOnForm(aform: TForm; aurl:Str);;
35279:  Proc WebToForm(aform: TForm; aurl:Str);;
35280:  Proc SearchAndHighlightWebText(aform: TForm; aurl:Str; aText:Str);
35281:  Proc SaveImagesOnWeb(aurl, apath:Str);
35282:  Func GetProcessNameFromWnd(Wnd: HWND):Str; //get EXE path from window handle
35283:  Func getallEvents(aform: TForm): TStringlist;
35284:  Proc GetKLList(List: TStrings);;
35285:  Proc GetKeyboardList(List: TStrings);;
35286:  Func SetSuspendState(Hibernate:Boolean;ForceCritical:Boolean;DisableWakeEvent:Bool):bool;
35287:          call: SetSuspendState(True, False, False);
35288:  Func ServiceRunning(sMachine, sService: PChar):Bool;
35289:  Func isServiceRunning(sMachine, sService: PChar):Bool;
35290:  Proc CloseOpenSockets( Sockets : array of TIdStackSocketHandle);
35291:  //SIRegister_SvrHTTPIndy - Linux
35292:  Proc TransformError( const Msg :Str);
35293:  Proc StringToFile2( const S, FileName :Str);
35294:  Func GetXMLData( DataSet : TClientDataSet) :Str;
35295:  Proc EditComTerminal( ComTerminal : TCustomComTerminal);  //TComTrmSetForm
35296:  Func search_adapter_key_networkcard:Str;  //at registry
35297:  Func getNetworkCard:Str;;
35298:  Func GetMacAddresses2(const Machine:Str; const Addresses: TStrings): Int;
35299: Func ConnectDrive(_drvLetter:str;_netPath:str;_showError:Bool;_reconnect:Bool):DWORD;
35300: Func ConnectPrinterDevice(_lptPort:str;_netPath:str;_showError:Bool;_reconnect:Bool):DWORD;
35301: Func DisconnectNetDrive(_locDrive:str;_showError:Bool;_force:Bool; _save:Bool): DWORD;
35302:     //ConnectDrive('k:', '\\Servername\C', True, True);
35303:     //DisconnectNetDrive('k:', True, True, True);
35304:  Func GetConnectionKind(var strKind:Str):Bool; MODEM =1; LAN=2; PROXY=4; BUSY= 8;
35305:  Func DownloadJPGToBitmap(const URL :Str; ABitmap: TBitmap):Bool;
35306:  Proc GetImageLinks(AURL:Str; AList: TStrings);
35307:  Func GetCharEncoding( alias :Str; var _name :Str) : Int;
35308:  Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
35309:  Func MicrosoftLangageCodeToISOCode( langcode : Int) :Str;
35310:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str = '';
35311:  Func RFC1123ToDateTime(Date:Str): TDateTime;
35312:  Func DateTimeToRFC1123(aDate: TDateTime):Str;
35313:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str);;
35314:  Proc DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);;
35315:  Proc DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);;
35316:  Proc DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);;
35317:
35318:  IPToHostName
35319:   //from ADOInt.pas //  TOleEnum = type LongWord;
35320:   TypeS('CursorOptionEnum', 'TOleEnum;
35321:    Interface(CL.FindInterface('IUNKNOWN'),_Recordset, '_Recordset;
35322:   //TypeS('_RecordsetDisp', 'dispinterface;
35323:   Interface(CL.FindInterface('IUNKNOWN'),_Command, '_Command;
35324:   Interface(CL.FindInterface('IUNKNOWN'),_Connection, '_Connection;
35325:   // SIRegister__Recordset(CL); // SIRegister__Command(CL);
35326:   Const IID_Recordset20','TGUID').SetString('{0000054F-0000-0010-8000-00AA006D2EA4}''];
35327:   'IID__Recordset','string').SetString( '00000555-0000-0010-8000-00AA006D2EA4;
35328:       //test with stringToGUID
35329:  //'CLASS_Command','TGUID').SetString( '{00000507-0000-0010-8000-00AA006D2EA4};
35330: // 'CLASS_Recordset','TGUID').SetString( '{00000535-0000-0010-8000-00AA006D2EA4};
35331:  { TypeS('Connection', '_Connection;
35332:   TypeS('Command', '_Command;  TypeS('Recordset', '_Recordset;
35333:   TypeS('Parameter', '_Parameter; TypeS('DataSpace', 'IDataspace;
35334:   TypeS('SearchDirection', 'SearchDirectionEnum; }
35335:   //TypeS('Command', '_Command; //TypeS('Recordset', '_Recordset;
35336: end;
35337:
35338: Proc SIRegister_DIUtils(CL: TPSPascalCompiler);
35339: begin
35340:  //'CRLF','String').SetString( #$0D#$0A);
35341:  'REPLACEMENT_CHARACTER','LongWord').SetUInt( $FFFD);
35342:  'HANGUL_SBase','LongWord').SetUInt( $AC00);
35343:  'HANGUL_LBase','LongWord').SetUInt( $1100);
35344:  'HANGUL_VBase','LongWord').SetUInt( $1161);
35345:  'HANGUL_TBase','LongWord').SetUInt( $11A7);
35346:  'HANGUL_LCount','LongInt').SetInt( 19);
35347:  'HANGUL_VCount','LongInt').SetInt( 21);
35348:  'HANGUL_TCount','LongInt').SetInt( 28);
35349:  'KEY_WOW64_32KEY','LongWord').SetUInt( $0200);
35350:  'KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
35351:  'KEY_WOW64_RES','LongWord').SetUInt( $0300);
35352:   TypeS('TAnsiCharSet', 'set of AnsiChar;
35353:   TypeS('TIsoDate', 'Cardinal;
```

```
35354:   TypeS('TJulianDate', 'Int;
35355:   //TypeS('PJulianDate', '^TJulianDate // will not work;
35356:   TypeS('TValidateCharFunc', 'function(const c: Char):Bool;;
35357:   // TValidateCharFunc = function(  const c: Char):Bool;
35358:  TypeS('TProcedureEvent', 'Procedure;
35359:  'MT19937_N','LongInt').SetInt( 624);
35360:  'MT19937_M','LongInt').SetInt( 397);
35361:   SIRegister_TMT19937(CL);
35362:   //SIRegister_TWideStrBuf(CL);
35363:  TypeS('TDITextLineBreakStyle', '( tlbsLF, tlbsCRLF, tlbsCR );
35364:  Func AdjustLineBreaksW(const s:UnicodeStr;const Style:TDITextLineBreakStyle):UnicodeStr;
35365:  Func BrightenColor( const Color : Int; const amount : Byte) : Int;
35366:  Func BSwap4( const Value :Card) :Card;;
35367:  Func BSwap5( const Value : Int) : Int;;
35368:  //Func BufCompNumIW(p1:PWideChar; l1:Int; p2 : PWideChar; l2 : Int) : Int;
35369:  Func BufSameA( p1, p2 : PChar; l :Card) :Bool;
35370: // Func BufSameW( p1, p2 : PWideChar; l :Card) :Bool;
35371:  Func BufSameIA( p1, p2 : PChar; l :Card) :Bool;
35372:  //Func BufSameIW( p1, p2 : PWideChar; l :Card) :Bool;
35373:  Func BufPosCharA(const Buf:PChar;l:Card;const c:AnsiChar;const Start:Card):Int;
35374:  Func BufPosCharsA(const Buf:PChar;l:Card;const Search:TAnsiCharSet;const Start:Card):Int;
35375:  Func BufStrSame(const Buf:PChar;const BufCharCount:Card;const s:Str):Bool;
35376:  Func BufStrSameA(const Buf:PChar; const BufCharCount:Card;const s:RawByteString):Bool;
35377:  Func BufStrSameI(const Buf:PChar; const BufCharCount:Card; const s :Str):Bool;
35378:  Func BufStrSameIA(const Buf:PChar;const BufCharCount:Card;const s:RawByteString): Bool;
35379:  Func diChangeFileExt(const FileName,Extension:Str):Str;
35380:  Func ChangeFileExtA( const FileName, Extension : Ansistr) : Ansistr;
35381:  Func ChangeFileExtW( const FileName, Extension : UnicodeString) : UnicodeString;
35382:  //Func CharDecomposeCanonicalW( const c : WideChar) : PCharDecompositionW;
35383:  Func CharDecomposeCanonicalStrW( const c : WideChar) : UnicodeString;
35384:  //Func CharDecomposeCompatibleW( const c : WideChar) : PCharDecompositionW;
35385:  Func CharDecomposeCompatibleStrW( const c : WideChar) : UnicodeString;
35386:  Func CharIn8( const c, t1, t2 : WideChar) :Bool;;
35387:  Func CharIn9( const c, t1, t2, t3 : WideChar) :Bool;;
35388:  Proc ConCatBuf(const Buffer: PChar;const CharCount:Card;var d:str;var InUse:Card);
35389: Proc ConCatBufA(const Buff:PChar;const AnsiCharCount:Card;var d:RawByteStr;var InUse:Card);
35390:  Proc ConCatChar( const c : Char; var d :Str; var InUse :Card);
35391:  Proc ConCatCharA( const c : AnsiChar; var d : RawByteString; var InUse :Card);
35392: // Proc ConCatCharW( const c : WideChar; var d : UnicodeString; var InUse :Card);
35393:  Proc ConCatStr( const s :Str; var d :Str; var InUse :Card);
35394:  Proc ConCatStrA( const s: RawByteString; var d : RawByteString; var InUse :Card);
35395:  //Proc ConCatStrW(const w:UnicodeString; var d : UnicodeString; var InUse :Card);
35396:  Func diCountBitsSet( const Value : Int) : Byte;
35397:  //Func Crc32OfStrA( const s : RawByteString) : TCrc32;
35398:  //Func Crc32OfStrW( const s : UnicodeString) : TCrc32;
35399:  Func CurrentDay : Word;
35400:  Func CurrentJulianDate : TJulianDate;
35401:  Func CurrentMonth : Word;
35402:  Func CurrentQuarter : Word;
35403:  Func diCurrentYear : Int;
35404:  Func DarkenColor( const Color : Int; const amount : Byte) : Int;
35405:  Func diDeleteFile( const FileName :Str) :Bool;
35406:  Func DeleteFileA( const FileName : Ansistr) :Bool;
35407:  //Func DeleteFileW( const FileName : UnicodeString) :Bool;
35408:  Func diDirectoryExists( const Dir :Str) :Bool;
35409:  Func DirectoryExistsA( const Dir : Ansistr) :Bool;
35410:  //Func DirectoryExistsW( const Dir : UnicodeString) :Bool;
35411:  Func diDiskFree( const Dir :Str) : Int64;
35412:  Func DiskFreeA( const Dir : Ansistr) : Int64;
35413:  //Func DiskFreeW( const Dir : UnicodeString) : Int64;
35414:  Func diExpandFileName( const FileName :Str) :Str;
35415:  Func ExpandFileNameA(const FileName: Ansistr): Ansistr;
35416:  //Func ExpandFileNameW( const FileName : UnicodeString) : UnicodeString;
35417:  Proc diExcludeTrailingPathDelimiter( var s :Str);
35418:  Proc ExcludeTrailingPathDelimiterA( var s : RawByteString);
35419:  //Proc ExcludeTrailingPathDelimiterW( var s : UnicodeString);
35420:  Func diExtractFileDrive( const FileName :Str) :Str;
35421:  Func ExtractFileDriveA( const FileName : RawByteString) : RawByteString;
35422:  //Func ExtractFileDriveW( const FileName : UnicodeString) : UnicodeString;
35423:  Func diExtractFileExt( const FileName :Str) :Str;
35424:  Func ExtractFileExtA( const FileName : RawByteString) : RawByteString;
35425:  //Func ExtractFileExtW( const FileName : UnicodeString) : UnicodeString;
35426:  Func diExtractFileName( const FileName :Str) :Str;
35427:  Func ExtractFileNameA(const FileName: Ansistr): Ansistr;
35428:  //Func ExtractFileNameW( const FileName : UnicodeString) : UnicodeString;
35429:  Func diExtractFilePath( const FileName :Str) :Str;
35430:  Func ExtractFilePathA( const FileName : RawByteString) : RawByteString;
35431:  //Func ExtractFilePathW( const FileName : UnicodeString) : UnicodeString;
35432:  Func ExtractNextWord10(const s:str;const ADelimiter:Char;var AStartIndex:Int):str;);
35433: Func ExtractNextWordA11(const s:RawByteStr;const ADelimiter:AnsiChar;var AStartIndex:Int): RawByteStr;
35434: Func ExtractNextWordW12(const s:UnicodeStr;const ADelimiter:WideChar;var AStartIndex:Int): UnicodeStr;
35435:  Func ExtractNextWord13(const s:str;const ADelims:TAnsiCharSet;var AStartIndex:Int):str;
35436:  Func ExtractNextWordA14(const s:RawByteString;const ADelimiters:TAnsiCharSet;var
       AStartIndex:Int):RawByteString;
35437:  Func diExtractWord(const Number:Card;const s:RawByteString;const Delimiters:TAnsiCharSet): RawByteString;
35438: Func ExtractWordA const Numb:Card;const s RawByteStr;const Delimiters:TAnsiCharSet): RawByteStr;
35439:  Func ExtractWordStartsA(const s:RawByteString;const MaxCharCount:Card;const WordSeparators:TAnsiCharSet):
       RawByteString;
35440:  Func diFileExists( const FileName :Str) :Bool;
```

```
35441:  Func FileExistsA( const FileName : Ansistr) :Bool;
35442:  //Func FileExistsW( const FileName : UnicodeString) :Bool;
35443:  Func diGCD( x, y :Card) :Card;
35444:  Func diGetTempFolder :Str;
35445:  Func GetTempFolderA : Ansistr;
35446:  //Func GetTempFolderW : UnicodeString;
35447:  Func diGetUserName( out UserName :Str) :Bool;
35448:  Func GetUserNameA( out UserName : Ansistr) :Bool;
35449:  //Func GetUserNameW( out UserName : UnicodeString) :Bool;
35450:  Func HexCodePointToInt( const c :Card) : Int;
35451:  Func diHexToInt( const s :Str) : Int;
35452:  Func HexToIntA( const s : RawByteString) : Int;
35453:  //Func HexToIntW( const s : UnicodeString) : Int;
35454:  Func BufHexToInt( p : PChar; l :Card) : Int;
35455:  Func BufHexToIntA( p : PChar; l :Card) : Int;
35456:  //Func BufHexToIntW( p : PWideChar; l :Card) : Int;
35457:  Proc IncludeTrailingPathDelimiterByRef( var s :Str);
35458:  Proc IncludeTrailingPathDelimiterByRefA(var s:RawByteString);
35459:  //Proc IncludeTrailingPathDelimiterByRefW(var w:UnicodeString);
35460:  Func IntToHex16(const Value: Int; const Digits : NativeInt) :Str;;
35461:  Func IntToHex17(const Value: Int64; const Digits : NativeInt) :Str;;
35462:  Func IntToHex18(const Value: UInt64; const Digits : NativeInt) :Str;;
35463:  Func IntToHexA( Value : UInt64; const Digits : NativeInt) : RawByteString;
35464:  //Func IntToHexW(Value : UInt64; const Digits : NativeInt) : UnicodeString;
35465:  Func IntToStrA19(const i: Int) : RawByteString;;
35466:  Func IntToStrW20(const i: Int) : UnicodeString;;
35467:  Func IntToStrA21(const i: Int64) : RawByteString;;
35468:  Func IntToStrW22(const i: Int64) : UnicodeString;;
35469:  Func CharDecomposeHangulW(const c: WideChar) : UnicodeString;
35470:  Func diIsPathDelimiter( const s :Str; const Index :Card) :Bool;
35471:  Func IsPathDelimiterA( const s : RawByteString; const Index :Card) :Bool;
35472:  //Func IsPathDelimiterW( const s : UnicodeString; const Index :Card) :Bool;
35473:  Func IsPointInRect( const Point : TPoint; const Rect : TRect) :Bool;
35474:  Func JulianDateToIsoDateStr(const Julian: TJulianDate):str;
35475:  Func JulianDateToIsoDateStrA( const Julian : TJulianDate) : RawByteString;
35476:  //Func JulianDateToIsoDateStrW( const Julian : TJulianDate) : UnicodeString;
35477:  Func LeftMostBit( Value :Card) : ShortInt;;
35478:  Func LeftMostBit2( Value : UInt64) : ShortInt;;
35479:  //Func MakeMethod( const AData, ACode : Pointer) : TMethod;
35480:  Func StrIsEmpty( const s :Str) :Bool;
35481:  Func StrIsEmptyA( const s : RawByteString) :Bool;
35482:  //Func StrIsEmptyW( const s : UnicodeString) :Bool;
35483:  Func PadLeftA(const Source:RawByteString;const Count:Card;const c:AnsiChar):RawByteStr;
35484:  //Func PadLeftW(const Source: UnicodeString; const Count :Card; const c : WideChar): UnicodeString;
35485:  Func PadRightA(const Source:RawByteString;const Count:Cardi;const c:AnsiChar):RawByteStr;
35486: //Func PadRightW(const Source:UnicodeString;const Count:Cardi;const c WideChar):UnicodeStr;
35487:  Func ProperCase( const s :Str) :Str;
35488:  Func ProperCaseA( const s : RawByteString) : RawByteString;
35489:  //Func ProperCaseW( const s : UnicodeString) : UnicodeString;
35490:  Proc ProperCaseByRefA( var s : RawByteString);
35491:  //Proc ProperCaseByRefW( var s : UnicodeString);
35492:  Func RegReadRegisteredOrganization( const Access : REGSAM) :Str;
35493:  Func RegReadRegisteredOrganizationA( const Access : REGSAM) : Ansistr;
35494:  //Func RegReadRegisteredOrganizationW( const Access : REGSAM) : UnicodeString;
35495:  Func RegReadRegisteredOwner( const Access : REGSAM) :Str;
35496:  Func RegReadRegisteredOwnerA( const Access : REGSAM) : Ansistr;
35497:  //Func RegReadRegisteredOwnerW( const Access : REGSAM) : UnicodeString;
35498:  Func RegReadStrDef( const Key : HKEY; const SubKey :Str; const ValueName:Str;const Default:str;const
       Access:REGSAM):Str;
35499:  Func RegReadStrDefA(const Key:HKEY;const SubKey:Ansistr;const ValueName:Ansistr;const Default:Ansistr;
       const Access:REGSAM):Ansistr;
35500:  Func StrDecodeUrlA( const Value : RawByteString): RawByteString;
35501:  Func StrEncodeUrlA( const Value : RawByteString): RawByteString;
35502:  Func diStrEnd( const s : PChar) : PChar;
35503:  Func StrEndA( const s : PChar) : PChar;
35504:  //Func StrEndW( const s : PWideChar) : PWideChar;
35505:  Proc StrIncludeTrailingChar( var s :Str; const c : Char);
35506:  Proc StrIncludeTrailingCharA( var s : RawByteString; const c : AnsiChar);
35507:  //Proc StrIncludeTrailingCharW( var s : UnicodeString; const c : WideChar);
35508:  Func diStrLen( const s : PChar) : NativeUInt;
35509:  Func StrLenA( const s : PChar) : NativeUInt;
35510:  //Func StrLenW( const s : PWideChar) : NativeUInt;
35511:  Func StrRandom(const ASeed:RawByteStr;const ACharacters:str;const ALength:Card):str);
35512:  Func StrRandomA(const ASeed:RawByteString;const ACharacters:RawByteString;const ALength:Card):RawByteStr;
35513:  Proc StrRemoveFromToIA(var Source:RawByteString;const FromString,ToString:RawByteString);
35514:  //Proc StrRemoveFromToIW(var Source:UnicodeString;const FromString, ToString : UnicodeString);
35515:  Proc StrRemoveSpacingA(var s:RawByteStr;const SpaceChars:TAnsiCharSet;const ReplaceChar:AnsiChar);
35516:  Proc diStrReplaceChar(var Source:Str;const SearchChar,ReplaceChar:Char);
35517:  Proc StrReplaceChar8(var s:Utf8String;const SearchChar,ReplaceChar:AnsiChar);
35518:  Proc StrReplaceCharA(var s:RawByteString;const SearchChar,ReplaceChar: AnsiChar);
35519:  //Proc StrReplaceCharW(var s:UnicodeString; const SearchChar, ReplaceChar : WideChar);
35520:  Func diStrReplace( const Source, Search, Replace :Str) :Str;
35521:  Func StrReplaceA( const Source, Search, Replace : RawByteString) : RawByteString;
35522:  //Func StrReplaceW( const Source, Search, Replace : UnicodeString) : UnicodeString;
35523:  Func StrReplaceI( const Source, Search, Replace :Str) :Str;
35524:  Func StrReplaceIA( const Source, Search, Replace : RawByteString) : RawByteString;
35525:  //Func StrReplaceIW( const Source, Search, Replace : UnicodeString) : UnicodeString;
35526:  Func StrReplaceLoopA( const Source, Search, Replace : RawByteString) : RawByteString;
35527:  //7Func StrReplaceLoopW( const Source, Search, Replace: UnicodeString) : UnicodeString;
```

```
35528:  Func StrReplaceLoopIA( const Source, Search, Replace : RawByteString) : RawByteString;
35529:  //Func StrReplaceLoopIW( const Source, Search, Replace: UnicodeString) : UnicodeString;
35530:  Func RightMostBit( const Value :Card) : ShortInt;;
35531:  Func RightMostBit2( const Value : UInt64) : ShortInt;;
35532:  Func LoadStrFromFile(const FileName:str; var s:RawByteString):Boolean;;
35533:  Func FileToStr( const FileName :Str; var s :Str) :Bool;;
35534:  Func LoadStrAFromFileA(const FileName:Ansistr;var s:RawByteString):Boolean;
35535:  //Func LoadStrAFromFileW(const FileName:UnicodeString; var s: RawByteString) :Bool;
35536:  Func LoadStrWFromFile28(const FileName:str;var s:UnicodeString):Boolean;;
35537:  Func LoadStrWFromFileA(const FileName: Ansistr;var s:UnicodeString):Bool;
35538:  //Func LoadStrWFromFileW(const FileName:UnicodeString;var s:UnicodeString):Bool;
35539:  Func QuotedStrW( const s : UnicodeString; const Quote : WideChar) : UnicodeString;
35540:  Func SaveStrToFile( const s :Str; const FileName :Str) :Bool;
35541:  Func StrToFile( const s :Str; const FileName :Str) :Bool;
35542:  Func SaveStrAToFile(const s:RawByteString;const FileName :Str) :Bool;
35543:  Func SaveStrAToFileA( const s : RawByteString; const FileName : Ansistr) :Bool;
35544:  //Func SaveStrAToFileW(const s:RawByteString; const FileName: UnicodeString) :Bool;
35545:  Func SaveStrWToFile(const s:UnicodeString;const FileName :Str):Boolean;
35546:  Func SaveStrWToFileA(const s: UnicodeString;const FileName:Ansistr): Bool;
35547:  //Func SaveStrWToFileW(const s:UnicodeString; const FileName : UnicodeString):Bool;
35548:  Func StrPosChar(const Source:str; const c:Char;const Start:Card):Card;
35549:  Func StrPosCharA(const Source:RawByteString;const c:AnsiChar;const Start:Car):Card;
35550:  //Func StrPosCharW(const Source:UnicodeStr;const c:WideChar;const Start:Cardi):Card;
35551:  Func StrPosCharBack(const Source:Str;const c:Char;const Start:Card):Card;
35552:  Func StrPosCharBackA(const Source:RawByteStr;const c:AnsiChar;const Start:Cardi):Card);
35553: Func StrPosCharsA(const Sourc:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
35554:  Func StrPosCharsBackA(const Src:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
35555:  Func StrPosNotCharsA(const Src:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
35556:  Func StrPosNotCharsBackA(const Source:RawByteString;const Search:TAnsiCharSet; const Start:Card):Card;
35557:  Func SetFileDate(const FileHandle:THandle;const Year:Int;const Month,Day:Word):Bool);
35558:  Func SetFileDate2( const FileName :Str; const JulianDate : TJulianDate) :Bool;
35559:  Func SetFileDateA( const FileName: Ansistr; const JulianDate:TJulianDate) :Bool;
35560:  //Func SetFileDateW(const FileName:UnicodeString;const JulianDate:TJulianDate):Boolean;
35561:  Func SetFileDateYmd(const FileName:str;const Year:Int;const Month,Day:Word):Bool;
35562:  Func SetFileDateYmdA(const FileName:Ansistr;const Year:Int;const Month,Day:Word):Bool;
35563:  Func SetFileDateYmdW(const FileName:UnicodeStr;const Year:Int;const Month,Day:Word):Bool;
35564:  Func StrContainsChar(const s:str; const c:Char; const Start :Card) :Bool;
35565:  Func StrContainsCharA(const s:RawByteString;const c:AnsiChar;const Start:Card):Bool;
35566:  //Func StrContainsCharW(const s:UnicodeString;const c:WideChar;const Start:Cardi):Bool;
35567:  Func StrContainsCharsA(const s:RawByteStr;const Chars:TAnsiCharSet;const Start:Card):Bool;
35568:  Func diStrSame( const s1, s2 :Str) :Bool;
35569:  Func StrSameA( const s1, s2 : RawByteString) :Bool;
35570:  //Func StrSameW( const s1, s2 : UnicodeString) :Bool;
35571:  Func StrSameI( const s1, s2 :Str) :Bool;
35572:  Func StrSameIA( const s1, s2 : RawByteString) :Bool;
35573:  //Func StrSameIW( const s1, s2 : UnicodeString) :Bool;
35574:  Func StrSameStart( const s1, s2 :Str) :Bool;
35575:  Func StrSameStartA( const s1, s2 : RawByteString) :Bool;
35576:  //Func StrSameStartW( const s1, s2 : UnicodeString) :Bool;
35577:  Func StrSameStartI( const s1, s2 :Str) :Bool;
35578:  Func StrSameStartIA( const s1, s2 : RawByteString) :Bool;
35579:  //Func StrSameStartIW( const s1, s2 : UnicodeString) :Bool;
35580:  Func diStrComp( const s1, s2 :Str) : Int;
35581:  Func StrCompA( const s1, s2 : RawByteString) : Int;
35582:  //Func StrCompW( const s1, s2 : UnicodeString) : Int;
35583:  Func StrCompI( const s1, s2 :Str) : Int;
35584:  Func StrCompIA( const s1, s2 : RawByteString) : Int;
35585:  //Func StrCompIW( const s1, s2 : UnicodeString) : Int;
35586:  Func StrCompNum( const s1, s2 :Str) : Int;
35587:  Func StrCompNumA( const s1, s2 : RawByteString) : Int;
35588: // Func StrCompNumW( const s1, s2 : UnicodeString) : Int;
35589:  Func StrCompNumI( const s1, s2 :Str) : Int;
35590:  Func StrCompNumIA( const s1, s2 : RawByteString) : Int;
35591: // Func StrCompNumIW( const s1, s2 : UnicodeString) : Int;
35592:  Func StrContains(const Search,Source:str; const Start:Card):Bool;
35593:  Func StrContainsA(const Search,Source:RawByteString;const Start:Card):Bool;
35594:  //Func StrContainsW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Bool;
35595:  Func StrContainsI(const Search, Source :Str; const Start :Card) :Bool;
35596:  Func StrContainsIA(const Search,Source:RawByteString; const Start:Card):Bool;
35597:  //Func StrContainsIW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Bool;
35598:  Func StrCountChar(const ASource:str;const c:Char;const AStartIdx:Card):Card;
35599:  Func StrCountCharA(const ASource:RawByteStr;const c:AnsiChar;const AStartIdx:Cardi):Card;
35600:  Func StrMatchesA(const Search,Source:RawByteString; const AStartIdx:Card):Bool;
35601:  Func StrMatchesIA(const Search,Source:RawByteString;const AStartIdx:Card):Bool;
35602:  Func StrMatchWild(const Source,Mask:str;const WildChar:Char;const MaskChar:Char):Bool;
35603:  Func StrMatchWildA(const Src,Mask:RawByteString;const WildChr:AnsiChar;const MaskChr:AnsiChar):Bool;
35604:  Func StrMatchWildI(const Source,Mask:str;const WildChar:Char;const MaskChar:Char):Bool;
35605:  Func StrMatchWildIA(const Source,Mask:RawByteString;const WildChar:AnsiChar;const MaskChar:AnsiChar):Bool;
35606:  Func diStrPos( const ASearch, ASource :Str; const AStartPos :Card) :Card;
35607:  Func StrPosA(const ASearch,ASource:RawByteString;const AStartPos:Card):Card;
35608:  //Func StrPosW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Card;
35609:  Func StrPosI( const ASearch, ASource :Str; const AStartPos :Card) :Card;
35610:  Func StrPosIA(const ASearch,ASource:RawByteString; const AStartPos:Card):Card;
35611:  //Func StrPosIW(const ASearch, ASource:UnicodeString;const AStartPos:Card):Card;
35612:  Func StrPosBackA( const ASearch, ASource: RawByteString; AStart :Card) :Card;
35613:  Func StrPosBackIA(const ASearch, ASource: RawByteString; AStart :Card) :Card;
35614:  //Func StrToIntDefW( const w : UnicodeString; const Default : Int) : Int;
35615:  Func StrToInt64DefW( const w : UnicodeString; const Default : Int64) : Int64;
35616:  Func StrToUpper( const s :Str) :Str;
```

```
35617:  Func StrToUpperA( const s : RawByteString) : RawByteString;
35618:  //Func StrToUpperW( const s : UnicodeString) : UnicodeString;
35619:  Proc StrToUpperInPlace( var s :Str);
35620:  Proc StrToUpperInPlaceA( var s : Ansistr);
35621:  //Proc StrToUpperInPlaceW31( var s : WideString);;
35622:  //Proc StrToUpperInPlaceW32( var s : UnicodeString);;
35623:  Func StrToLower( const s :Str) :Str;
35624:  Func StrToLowerA( const s : RawByteString) : RawByteString;
35625:  //Func StrToLowerW( const s : UnicodeString) : UnicodeString;
35626:  Proc StrToLowerInPlace( var s :Str);
35627:  Proc StrToLowerInPlaceA( var s : Ansistr);
35628:  //Proc StrToLowerInPlaceW33( var s : WideString);;
35629:  //Proc StrToLowerInPlaceW34( var s : UnicodeString);;
35630:  Proc StrTimUriFragmentA( var Value : RawByteString);
35631:  //Proc StrTrimUriFragmentW( var Value : UnicodeString);
35632:  //Func StrExtractUriFragmentW( var Value : UnicodeString): UnicodeString;
35633:  Func StrCountUtf8Chars(const AValue: Utf8String) :Card;
35634:  Func StrDecodeUtf8( const AValue: Utf8String): UnicodeString;
35635:  Func StrEncodeUtf8( const AValue: UnicodeString): Utf8String;
35636:  Func diSysErrorMessage( const MessageID :Card) :Str;
35637:  Func SysErrorMessageA(const MessageID:Card): Ansistr;
35638:  //Func SysErrorMessageW( const MessageID :Card) : UnicodeString;
35639:  Func TextExtentW( const DC : HDC; const Text : UnicodeString) : TSize;
35640:  Func TextHeightW( const DC : HDC; const Text : UnicodeString) : Int;
35641:  Func TextWidthW( const DC : HDC; const Text : UnicodeString) : Int;
35642:  Func diStrTrim( const Source :Str) :Str;
35643:  Func StrTrimA( const Source : RawByteString) : RawByteString;
35644:  //Func StrTrimW( const w : UnicodeString) : UnicodeString;
35645:  Func StrTrimCharA(const Source:RawByteString;const CharToTrim:AnsiChar): RawByteString;
35646:  Func StrTrimCharsA(const Source:RawByteStr;const CharsToTrim:TAnsiCharSet):RawByteString;
35647: //Func StrTrimCharsW(const s:UnicodeStr;const IsCharToTrim:TValidateCharFuncW):UnicodeStr;
35648:  Proc TrimLeftByRefA( var s : RawByteString; const Chars : TAnsiCharSet);
35649:  Func TrimRightA( const Source : RawByteString; const s : TAnsiCharSet) : RawByteString;
35650:  Proc TrimRightByRefA( var Source : RawByteString; const s : TAnsiCharSet);
35651:  Proc StrTrimCompressA(var s:RawByteString;const TrimCompressChars:TAnsiCharSet;const
        ReplaceChar:AnsiChar);
35652:  Func TryStrToIntW( const w : UnicodeString; out Value : Int) :Bool;
35653:  Func TryStrToInt64W( const w : UnicodeString; out Value : Int64) :Bool;
35654:  Func ValInt( const p : PChar; const l : Int; out Code : Int) : Int;;
35655:  Func ValIntA36( p : PChar; l : Int; out Code : Int) : Int;;
35656:  //Func ValIntW37( p : PWideChar; l : Int; out Code : Int) : Int;;
35657:  Func ValInt2(const s:Str; out Code : Int): Int;;
35658:  Func ValIntA39( const s : RawByteString; out Code : Int) : Int;;
35659:  Func ValIntW40( const s : UnicodeString; out Code : Int) : Int;;
35660:  Func ValInt64A41( p : PChar; l : Int; out Code : Int) : Int64;;
35661:  //Func ValInt64W42( p : PWideChar; l : Int; out Code : Int) : Int64;;
35662:  Func ValInt64A43( const s : RawByteString; out Code : Int) : Int64;;
35663:  Func ValInt64W44( const s : UnicodeString; out Code : Int) : Int64;;
35664:  Func YmdToIsoDateStr(constYear:Int;const Month:Word;const Day:Word):str;
35665:  Func YmdToIsoDateStrA(const Year:Int;const Month:Word; const Day:Word):RawByteString;
35666:  Func YmdToIsoDateStrW(const Year:Int;const Month:Word; const Day:Word):UnicodeString;
35667:  Func CharIsLetterW( const c : WideChar) :Bool;
35668:  Func CharIsLetterCommonW( const c : WideChar) :Bool;
35669:  Func CharIsLetterUpperCaseW(const c : WideChar) :Bool;
35670:  Func CharIsLetterLowerCaseW(const c : WideChar) :Bool;
35671:  Func CharIsLetterTitleCaseW(const c : WideChar) :Bool;
35672:  Func CharIsLetterModifierW(const c : WideChar) :Bool;
35673:  Func CharIsLetterOtherW(const c : WideChar) :Bool;
35674:  Func CharIsMarkW( const c : WideChar) :Bool;
35675:  Func CharIsMarkNon_SpacingW( const c : WideChar) :Bool;
35676:  Func CharIsMarkSpacing_CombinedW(const c: WideChar):Bool;
35677:  Func CharIsMarkEnclosingW( const c : WideChar) :Bool;
35678:  Func CharIsNumberW( const c : WideChar) :Bool;
35679:  Func CharIsNumber_DecimalW( const c : WideChar) :Bool;
35680:  Func CharIsNumber_LetterW( const c : WideChar) :Bool;
35681:  Func CharIsNumber_OtherW( const c : WideChar) :Bool;
35682:  Func CharIsPunctuationW( const c : WideChar) :Bool;
35683:  Func CharIsPunctuation_ConnectorW( const c : WideChar) :Bool;
35684:  Func CharIsPunctuation_DashW( const c : WideChar) :Bool;
35685:  Func CharIsPunctuation_OpenW( const c : WideChar) :Bool;
35686:  Func CharIsPunctuation_CloseW( const c : WideChar) :Bool;
35687:  Func CharIsPunctuation_InitialQuoteW( const c : WideChar) :Bool;
35688:  Func CharIsPunctuation_FinalQuoteW( const c : WideChar) :Bool;
35689:  Func CharIsPunctuation_OtherW( const c : WideChar) :Bool;
35690:  Func CharIsSymbolW( const c : WideChar) :Bool;
35691:  Func CharIsSymbolMathW( const c : WideChar) :Bool;
35692:  Func CharIsSymbolCurrencyW( const c : WideChar) :Bool;
35693:  Func CharIsSymbolModifierW( const c : WideChar) :Bool;
35694:  Func CharIsSymbolOtherW( const c : WideChar) :Bool;
35695:  Func CharIsSeparatorW( const c : WideChar) :Bool;
35696:  Func CharIsSeparatorSpaceW( const c : WideChar) :Bool;
35697:  Func CharIsSeparatorLineW( const c : WideChar) :Bool;
35698:  Func CharIsSeparatorParagraphW( const c : WideChar) :Bool;
35699:  Func CharIsOtherW( const c : WideChar) :Bool;
35700:  Func CharIsOtherControlW( const c : WideChar) :Bool;
35701:  Func CharIsOtherFormatW( const c : WideChar) :Bool;
35702:  Func CharIsOtherSurrogateW( const c : WideChar) :Bool;
35703:  Func CharIsOtherPrivateUseW( const c : WideChar) :Bool;
35704:  Func BitClear( const Bits, BitNo : Int) : Int;
```

```
35705:  Func BitSet( const Bits, BitIndex : Int) : Int;
35706:  Func BitSetTo( const Bits, BitIndex : Int; const Value :Bool) : Int;
35707:  Func BitTest( const Bits, BitIndex : Int) :Bool;
35708:  Func CharCanonicalCombiningClassW( const Char : WideChar) :Card;
35709:  Func CharIsAlphaW( const c : WideChar) :Bool;
35710:  Func CharIsAlphaNumW( const c : WideChar) :Bool;
35711:  Func CharIsCrLf( const c : Char) :Bool;
35712:  Func CharIsCrLfA( const c : AnsiChar) :Bool;
35713:  //Func CharIsCrLfW( const c : WideChar) :Bool;
35714:  Func diCharIsDigit( const c : Char) :Bool;
35715:  Func CharIsDigitA( const c : AnsiChar) :Bool;
35716:  //Func CharIsDigitW( const c : WideChar) :Bool;
35717:  Func CharIsHangulW( const Char : WideChar) :Bool;
35718:  Func CharIsHexDigitW( const c : WideChar) :Bool;
35719:  Func CharIsWhiteSpaceW( const c : WideChar) :Bool;
35720:  Func CharToCaseFoldW( const Char : WideChar) : WideChar;
35721:  Func CharToLowerW( const Char : WideChar) : WideChar;
35722:  Func CharToUpperW( const Char : WideChar) : WideChar;
35723:  Func CharToTitleW( const Char : WideChar) : WideChar;
35724:  Func DayOfJulianDate( const JulianDate : TJulianDate) : Word;
35725:  Func diDayOfWeek( const JulianDate : TJulianDate) : Word;
35726:  Func DayOfWeekYmd( const Year : Int; const Month, Day : Word) : Word;
35727:  Func diDaysInMonth( const JulianDate : TJulianDate) : Word;
35728:  Func DaysInMonthYm( const Year : Int; const Month : Word) : Word;
35729:  Proc DecDay( var Year : Int; var Month, Day : Word);
35730:  Proc DecDays( var Year : Int; var Month, Day : Word; const Days : Int);
35731:  Func diDeleteDirectory( const Dir :Str; const DeleteItself :Bool) :Bool;
35732:  Func DeleteDirectoryA( Dir : Ansistr; const DeleteItself :Bool) :Bool;
35733:  //Func DeleteDirectoryW( Dir : UnicodeString; const DeleteItself :Bool) :Bool;
35734:  Func diEasterSunday( const Year : Int) : TJulianDate;
35735:  Proc EasterSundayYmd( const Year : Int; out Month, Day : Word);
35736:  Func diFirstDayOfWeek( const JulianDate : TJulianDate) : TJulianDate;
35737:  Proc FirstDayOfWeekYmd( var Year : Int; var Month, Day : Word);
35738:  Func diFirstDayOfMonth( const Julian : TJulianDate) : TJulianDate;
35739:  Proc FirstDayOfMonthYmd( const Year : Int; const Month : Word; out Day : Word);
35740:  Func diForceDirectories( const Dir :Str) :Bool;
35741:  Func ForceDirectoriesA( Dir : Ansistr) :Bool;
35742:  //Func ForceDirectoriesW( Dir : UnicodeString) :Bool;
35743:  Proc FreeMemAndNil( var Ptr: TObject);
35744:  Func diGetCurrentFolder :Str;
35745:  Func GetCurrentFolderA : Ansistr;
35746:  //Func GetCurrentFolderW : UnicodeString;
35747:  Proc SetCurrentFolder( const NewFolder :Str);
35748:  Proc SetCurrentFolderA( const NewFolder : Ansistr);
35749:  //Proc SetCurrentFolderW( const NewFolder : UnicodeString);
35750:  Func diGetDesktopFolder :Str;
35751:  Func GetDesktopFolderA : Ansistr;
35752:  //Func GetDesktopFolderW : UnicodeString;
35753:  Func diGetFileSize( const AFileName :Str) : Int64;
35754:  Func GetFileSizeA( const AFileName : Ansistr) : Int64;
35755:  //Func GetFileSizeW( const AFileName : UnicodeString) : Int64;
35756:  Func diGetDesktopDirectoryFolder :Str;
35757:  Func GetDesktopDirectoryFolderA : Ansistr;
35758:  //Func GetDesktopDirectoryFolderW : UnicodeString;
35759:  Func GetFileLastWriteTime(const FileName:Str; out FileTime : TFileTime) :Bool;
35760:  Func GetFileLastWriteTimeA(const FileName:Ansistr; out FileTime:TFileTime):Boolean;
35761:  //Func GetFileLastWriteTimeW(const FileName:UnicodeString;out FileTime:TFileTime):Bool;
35762:  Func diGetPersonalFolder( const PersonalFolder : Int) :Str;
35763:  Func GetPersonalFolderA : Ansistr;
35764:  //Func GetPersonalFolderW : UnicodeString;
35765:  Func GetSpecialFolder( const SpecialFolder : Int) :Str;
35766:  Func GetSpecialFolderA( const SpecialFolder : Int) : Ansistr;
35767:  //Func GetSpecialFolderW( const SpecialFolder : Int) : UnicodeString;
35768:  Proc diIncMonth( var Year : Int; var Month, Day : Word);
35769:  Proc diIncMonths(var Year :Int; var Month, Day:Word; const NumberOfMonths:Int);
35770:  Proc diIncDay( var Year : Int; var Month, Day : Word);
35771:  Proc IncDays( var Year : Int; var Month, Day : Word; const Days : Int);
35772:  Func IsDateValid( const Year : Int; const Month, Day : Word) :Bool;
35773:  Func IsHolidayInGermany( const Julian : TJulianDate) :Bool;
35774:  Func IsHolidayInGermanyYmd( const Year : Int; const Month, Day : Word) :Bool;
35775:  Func diIsLeapYear( const Year : Int) :Bool;
35776:  Func ISODateToJulianDate( const ISODate : TIsoDate) : TJulianDate;
35777:  Proc ISODateToYmd(const ISODate: TIsoDate; out Year : Int; out Month, Day : Word);
35778:  Func IsCharLowLineW( const c : WideChar) :Bool;
35779:  Func IsCharQuoteW( const c : WideChar) :Bool;
35780:  Func IsShiftKeyDown :Bool;
35781:  Func IsCharWhiteSpaceOrAmpersandW( const c : WideChar) :Bool;
35782:  Func IsCharWhiteSpaceOrNoBreakSpaceW( const c : WideChar) :Bool;
35783:  Func IsCharWhiteSpaceOrColonW( const c : WideChar) :Bool;
35784:  Func CharIsWhiteSpaceGtW( const c : WideChar) :Bool;
35785:  Func CharIsWhiteSpaceLtW( const c : WideChar) :Bool;
35786:  Func CharIsWhiteSpaceHyphenW( const c : WideChar) :Bool;
35787:  Func CharIsWhiteSpaceHyphenGtW( const c : WideChar) :Bool;
35788:  Func IsCharWordSeparatorW( const c : WideChar) :Bool;
35789:  Func diISOWeekNumber( const JulianDate : TJulianDate) : Word;
35790:  Func ISOWeekNumberYmd( const Year : Int; const Month, Day : Word) : Word;
35791: Func ISOWeekToJulianDate(const Year:Int;const WeekOfYear:Word;const DayOfWeek:Word):TJulianDate);
35792:  Func JulianDateIsWeekDay( const JulianDate : TJulianDate) :Bool;
35793:  Func JulianDateToIsoDate( const Julian : TJulianDate) : TIsoDate;
```

```
35794:  Proc JulianDateToYmd(const JulianDate:TJulianDate;out Year: Int;out Month,Day:Word);
35795:  Func LastDayOfMonth( const JulianDate : TJulianDate) : TJulianDate;
35796:  Proc LastDayOfMonthYmd( const Year : Int; const Month : Word; out Day : Word);
35797:  Func LastDayOfWeek( const JulianDate : TJulianDate) : TJulianDate;
35798:  Proc LastDayOfWeekYmd( var Year : Int; var Month, Day : Word);
35799:  Func LastSysErrorMessage :Str;
35800:  Func LastSysErrorMessageA : Ansistr;
35801:  Func LastSysErrorMessageW : UnicodeString;
35802:  Func diMax( const a : Int; const b : Int) : Int;;
35803:  Func diMax3( const a, b, c : Int) : Int;
35804:  Func MaxCard( const a :Card; const b :Card) :Card;;
35805:  Func MaxCard3( const a :Card; const b :Card; const c :Card):Card;;
35806:  Func diMaxint64( const a : Int64; const b : Int64) : Int64;;
35807:  Func dimaxint643( const a : Int64; const b : Int64; const c : Int64) : Int64;;
35808:  Func diMin( const a, b : Int) : Int;;
35809:  Func diMin3( const a, b, c : Int) : Int;
35810:  Func MinCard( const a, b :Card) :Card;;
35811:  Func Mincard3( const a, b, c :Card) :Card;;
35812:  Func diMinint64( const a, b : Int64) : Int64;;
35813:  Func diMinint643( const a, b, c : Int64) : Int64;;
35814:  Func diMinint64U( const a, b : UInt64) : UInt64;;
35815:  Func diMinint643U( const a, b, c : UInt64) : UInt64;;
35816:  Func MonthOfJulianDate( const JulianDate : TJulianDate) : Word;
35817:  Func YearOfJuilanDate( const JulianDate : TJulianDate) : Int;
35818:  Func YmdToIsoDate( const Year : Int; const Month, Day : Word) : TIsoDate;
35819:  Func YmdToJulianDate( const Year : Int; const Month, Day : Word) : TJulianDate;
35820: end;
35821:
35822:  Func FiboMaxE2(n: Int): Extended;
35823:   begin
35824:     result:= (pow((1+SQRT5)/2,n)-pow((1-SQRT5)/2,n))/SQRT5
35825:   end;
35826:  TDLLVersionInfo=Record
35827:     cbSize,   // Size of the structure, in bytes.
35828:     dwMajorVersion, // Major version of  DLL
35829:     dwMinorVersion, // Minor version of  DLL
35830:     dwBuildNumber, // Build number of  DLL
35831:     dwPlatformID: DWord; //Identifies the platform for which the DLL was built
35832:  end;
35833:
35834:  {$IFDEF MSWINDOWS}
35835: Proc TIdAntiFreeze_Process;
35836: var Msg: TMsg;
35837:   ApplicationHasPriority:Bool;
35838: begin
35839:   if ApplicationHasPriority then begin
35840:     Application.ProcessMessages;
35841:   end else begin
35842:     // This guarantees it will not ever call Application.Idle
35843:     if PeekMessage(Msg, 0, 0, 0, PM_NOREMOVE) then begin
35844:       Application.HandleMessage;
35845:     end;
35846:   end;
35847: end;
35848: {$ENDIF}
35849:
35850: with lockbox you do sha1 with strings and hex:
35851: writeln(SHA1ofStr(loadstringj(exepath+'maxbox4.exe')))
35852: // ¢'ÞRL'mÍÕïÝ$1ÎÓôî
35853: tempstr:= SHA1ofStr(loadstringj(exepath+'maxbox4.exe'))
35854: writeln(' ')
35855: for i:= 1 to 20 do
35856:   Write(charToHexStr(tempstr[i]));
35857:   A291DE524C1B916DCDD502EFDDA76C0BCED3F0EE
35858:
35859: Example with WMI:
35860: Func TestWMIOS2:Str;
35861: var wmiService, computerName, objectsList: OLEvariant;
35862:   sysID:Str;
35863:   isloc: ISWBemLocator; isser: ISWBemServices;
35864:   iset: ISWbemObjectSet; ENum: IEnumVariant;
35865:   tempObj: OleVariant;
35866: begin
35867:   computerName := '.';
35868:   isloc:= WMIStart;
35869:   isser:= WMIConnect2(isloc, '\root\cimv2','maxbox10','max',''); //: ISWBemServices;
35870:   //Exception: incorrect credentials. WMI connection failed.
35871:   //Set wmiService = GetObject("WinMgmts:{impersonationLevel=impersonate, "&_
35872:   // "(SystemTime, Shutdown)}!\\" & computerName & "\root\cimv2")
35873:   //Set objectsList = wmiService.InstancesOf("Win32_OperatingSystem")
35874:   //For Each ObjService in objectsList
35875:
35876:   iset:= WMIExecQuery(isser,'Select * from Win32_OperatingSystem')
35877:   result:=(botoStr(WMIRowFindFirst(iset, ENum, tempObj)));
35878:   repeat
35879:         sysid:= ('OSid: '+vartoStr(tempobj.csname)+
35880:         ' - '+vartoStr(tempobj.name)+
35881:         ' - '+vartoStr(tempobj.caption))
35882:   until not WMIRowFindNext(ENum, tempobj); //}
```

```
35883:   tempObj:= unassigned;
35884:   result:= result+' '+sysID;
35885: //Call Log.Message("SysInfo", sysID)
35886: End; // Sub
35887:
35888: >>> Sysid2: TRUE OSid: MAXBOX10 - Microsoft Windows 10 Home|C:\WINDOWS|\Device\Harddisk0\Partition4 -
       Microsoft Windows 10 Home
35889:   command1:= 'play "'+songpath+'maxbox.wav"'; command2:= 'play "'+songpath+'moon.wav"';
35890:   SendMCICommand('open waveaudio shareable;   //parallels
35891:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav";
35892:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav";
35893:   SendMCICommand('close waveaudio;
35894: //MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
35895:   if Not fileexists(exepath+'\mytest1.wav') then begin
35896:     Writeln('Generate 2 sounds with 440 and 443 Hz');
35897:     MakeSound(440,5000, 80, Exepath+'\mytest1.wav');
35898:     MakeSound(443,5000, 80, Exepath+'\mytest2.wav');
35899:   end else begin
35900:     Writeln('Play Beat of 2 sounds together with 3 Hz Amp Mod:');
35901:     mciSendString('PLAY "'+exepath+'\mytest1.wav'+'"', 'Nil', 0, 0);
35902:     mciSendString('PLAY "'+exepath+'\mytest2.wav'+'"', 'Nil', 0, 0);
35903:   end;
35904:
35905: Strings (and other non ordinals) in Case Statements
35906: This has to be most requested feature by far in history of D imho, sure to make many long time D fans
       happy.}
35907:   writeln(floattostr(pow(256,200000)))
35908:     //writeln(itoa(length(bigpow(256,200000))))   //481648
35909:   sr:= sworld;
35910:   case sr of
35911:     'hello' : xi := 1;
35912:     'goodbye' : xi := 2;
35913:     sWorld : xi := 3; //sWorld is a string constant
35914:   end;
35915:   writeln('res of case '+itoa(xi));
35916:
35917: initialization  XRaise
35918:
35919: type
35920:  EPointsArrayIsNotAssigned = Exception;
35921:  writeln(itoa(length(aptsarr)));
35922:  setlength(aptsarr, 2)
35923:  writeln(itoa(length(aptsarr)));
35924:  if length(aptsarr) = 0 then begin
35925:    Xraise(EPointsArrayIsNotAssigned.Create('Points array is not assigned...'));
35926:    //Exit;
35927:  end;
35928:
35929: maXbox SSL fox471 for
35930: The first step the following components, maybe we will add further in the future:
35931: 1. Orpheus (Win32 and Win64)
35932: 2. Abbrevia (Win32, Win64, MacOSX, iOS and Android)
35933: 3. Virtual Tree (Win32 and Win64)
35934: 4. SynEdit (Win32 and Win64)
35935: 5. LockBox (Win32, Win64, MacOSX, iOS and Android)
35936: 6. Async Professional (Win32)
35937: 7. PowerPDF (Win32 and Win64)
35938:
35939: ////////////////////////////////////////////////////////////////////////////
35940: All maXbox Tutorials Table of Content 2014/2015/2016/2017/2018/2019/2020/2021/2022
35941: ////////////////////////////////////////////////////////////////////////////
35942:   Tutorial 00 Function-Coding  (Blix the Programmer)
35943:   Tutorial 01 Procedural-Coding
35944:   Tutorial 02 OO-Programming
35945:   Tutorial 03 Modular Coding
35946:   Tutorial 04 UML Use Case Coding
35947:   Tutorial 05 Internet Coding
35948:   Tutorial 06 Network Coding
35949:   Tutorial 07 Game Graphics Coding
35950:   Tutorial 08 Operating System Coding
35951:   Tutorial 09 Database Coding
35952:   Tutorial 10 Statistic Coding
35953:   Tutorial 10_1 Probability Coding
35954:   Tutorial 11 Forms Coding
35955:   Tutorial 12 SQL DB Coding
35956:   Tutorial 13 Crypto Coding
35957:   Tutorial 14 Parallel Coding
35958:   Tutorial 15 Serial RS232 Coding
35959:   Tutorial 16 Event Driven Coding
35960:   Tutorial 17 Web Server Coding
35961:   Tutorial 18 Arduino System Coding
35962:   Tutorial 18_3 RGB LED System Coding
35963:   Tutorial 19 WinCOM /Arduino Coding
35964:   Tutorial 20 Regular Expressions RegEx
35965:   Tutorial 21 Android Coding (coming 2013)
35966:   Tutorial 22 Services Programming
35967:   Tutorial 23 Real Time Systems
35968:   Tutorial 24 Clean Code
35969:   Tutorial 25 maXbox Configuration I+II
```

```
35970:   Tutorial 26 Socket Programming with TCP
35971:   Tutorial 27 XML & TreeView
35972:   Tutorial 28 DLL Coding (available)
35973:   Tutorial 29 UML Scripting (2014)
35974:   Tutorial 30 Web of Things (2014)
35975:   Tutorial 31 Closures (2014)
35976:   Tutorial 32 SQL Firebird (2014)
35977:   Tutorial 33 Oscilloscope (available)
35978:   Tutorial 34 GPS Navigation (2014)
35979:   Tutorial 35 Web Box (available)
35980:   Tutorial 36 Unit Testing (avail)
35981:   Tutorial 37 API Coding (avail)
35982:   Tutorial 38 3D Coding (coming 2015)
35983:   Tutorial 39 GEO Map Coding (available)
35984:   Tutorial 39_1 GEO Map Layers Coding (available)
35985:   Tutorial 40 REST Coding (coming 2015)
35986:   Tutorial 41 Big Numbers Coding (2015)
35987:   Tutorial 42 Parallel Processing (2015)
35988:   Tutorial 43 Code Metrics: June2016
35989:   Tutorial 44 IDE Extensions
35990:   Tutorial 45 Robotics: July2016
35991:   Tutorial 46 WineHQ: Dez2016
35992:   Tutor 47 RSA Crypto Jan2017
35993:   Tutor 48 Microservice Jan2017
35994:   http://www.softwareschule.ch/download/maxbox_starter48.pdf
35995:   Tutorial 49 Refactoring: March 2017
35996:   Tutorial 50 Big Numbers II: April 2017
35997:   Tutorial 51 5 Use Cases April 2017
35998:   Tutorial 52 Work with WMI Mai 2017
35999:   Tutorial 53 Real Time UML 2017
36000:   Tutor 54 Microservice II Nov 2017
36001:   Tutor 55 ASCII Talk Dez 2017
36002:   Tutor 56 Neural Network 2018
36003:   Tutor 57 Neural Network II
36004:   Tutor 58 Data Science
36005:   Tutor 59 Big Data Feb 2018
36006:   Tutor 60 Machine Learning March 2018
36007:   Tutor 60.1 Sentiment Analysis
36008:   Tutor 60.2 ML II
36009:   Tutor 63 Machine Games
36010:   Tutor 64 Install Routines
36011:   Tutor 65 Machine Learning III
36012:   Tutor 66 Machine Learning IV
36013:   Tutor 67 Machine Learning V
36014:   Tutor 68 Machine Learning VI
36015:   Tutor 69 Machine Learning VII
36016:   Tutor 70 No GUI Shell
36017:   Tutor 71 CGI and WebSocket
36018:   Tutor 72 Multilanguage
36019:   Tutor 73 EKON 24 Edition
36020:   Tutor 74 BASTA 2020 Vision
36021:   Tutor 75 Object Detection
36022:   Tutor 76 Image Classifier CAI
36023:   Tutor 77 Image Classifier Matrix
36024:   Tutor 78 Portable pixmap format(PPM)
36025:   Tutor 79 Unit Testing with Asserts
36026:   Tutor 80 My Tips & Tricks
36027:   Tutor 81 RSS Feeds of BBC News - Digital Forensic March 2021
36028:   Tutorial 82 JSON Code API
36029:   Tutorial 82_2 Sentiment NLTK API Code
36030:   #Tutorial 83 Unicode + Bayes Theorem & Confusion Matrix April 2021
36031:   #Tutorial 84 JSON Primer Mai 2021
36032:   Tutorial 83 Machine Learning XI Classification April 2021
36033:   Tutorial 84 Machine Learning XII Baseline Mai 2021
36034:   Tutorial 85 JSON4Delphi Automation Mai 2021
36035:   Tutorial 86 Python4Delphi P4D June 2021 (Part 1-4)
36036:   Tutorial 87 Image Detection with Lazarus
36037:   Tutorial 88 Performance Tuning
36038:   Tutorial 89 CAI mX4
36039:   Tutorial 89_1 CAI mX4 model define
36040:   Tutorial 89_2 CAI mX4 model fits
36041:   Tutorial 90 Python CheatSheet Nov. 2021
36042:   Tutorial 91 CAI CheatSheet Faker SynDat Dez. 2021
36043:   Tutorial 92 Python_CAI CheatSheet Jan. 2022
36044:   Tutorial 92_1 VCL4Python Feb. 2022
36045:   Tutorial 93 Geocoding March 2022
36046:   Tutorial 94 Post API Services April 2022
36047:   Tutorial 95 Language Services April 2022
36048:   Tutorial 96 CNN Pipeline June 2022
36049:   Tutor 97 Operating System Services OSS
36050:   Tutor 98 Python Integration
36051:
36052: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
36053: using Docu for this file is maxbox_functions_all.pdf
36054: PEP - Pascal Education Program  Low Lib Lab ShellHell in UDEMY
36055:
36056: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
36057: http://stackoverflow.com/tags/pascalscript/hot
36058: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
```

```
36059: http://sourceforge.net/projects/maXbox   #locs:62000
36060: http://sourceforge.net/apps/mediawiki/maXbox
36061: http://www.blaisepascal.eu/
36062: https://github.com/maxkleiner/maXbox3.git
36063: https://github.com/maxkleiner/maXbox4.git
36064: http://www.heise.de/download/maxbox-1176464.html
36065: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
36066: https://www.facebook.com/pages/Programming-maXbox/166844836691703
36067: http://www.softwareschule.ch/arduino_training.pdf
36068: http://www.delphiarea.com
36069: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
36070: http://entwickler-konferenz.de/2014/speakers/max-kleiner
36071: http://www.heise.de/download/maxbox-1176464.html
36072: https://www.udemy.com/learn-coding-from-the-scratch
36073: http://www.slideshare.net/maxkleiner1/codesign-2015
36074: https://www.dropbox.com/s/yolconwmg4oqta4/Blaise_2and3_SP_Total.pdf?dl=0
36075: http://www.softwareschule.ch/download/maxbox_promo.png
36076: http://max.kleiner.com/maxbox_functions_all.htm
36077: http://max.kleiner.com/boxart.htm
36078: http://www.jurgott.org/linkage/util.htm
36079: http://www.swissdelphicenter.ch/en/niklauswirth.php
36080: http://www.softwareschule.ch/images/maxbox_20years_delphi.jpg
36081: https://softwareschule.code.blog/
36082: https://repl.it/@MaxKleiner/machinelearning4#main.py
36083: https://github.com/project-jedi/jvcl/tree/master/jvcl/examples/JvChartDemo
36084: https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
36085:
36086: All maXbox 4 Examples List Exampleslist
36087: https://github.com/maxkleiner/maXbox3/releases
36088: https://github.com/maxkleiner/maXbox4/releases
36089: http://www.softwareschule.ch/download/exampleedition2016.zip
36090:
36091: ***********************************************************************************************
36092: 000_pas_baseconvert.txt                          282_fadengraphik.txt
36093: 000_pas_baseconvert.txt_encrypt                  283_SQL_API_messagetimeout.txt
36094: 000_pas_baseconvert.txt_decrypt                  284_SysTools4.txt
36095: 001_1_pas_functest - Kopie.txt                   285_MineForm_GR32.TXT
36096: 001_1_pas_functest.txt                           285_MineForm_GR32main.TXT
36097: 001_1_pas_functest2.txt                          285_MineForm_GR32mainsolution.TXT
36098: 001_1_pas_functest_clx2.txt                      285_MineForm_propas.TXT
36099: 001_1_pas_functest_clx2_2.txt                    285_MineForm_propas2.TXT
36100: 001_1_pas_functest_openarray.txt                 285_minesweeper2.TXT
36101: 001_pas_lottogen.txt                             285_Patterns_process.txt
36102: 001_pas_lottogen_template.txt                    286_colormixer_jpeg_charcounter.txt
36103: 001_pas_lottogen.txtcopy                         286_colormixer_jpeg_charcounter2.txt
36104: 002_pas_russianroulette.txt                      287_eventhandling.txt
36105: 002_pas_russianroulette.txtcopy                  287_eventhandling2.txt
36106: 002_pas_russianroulette.txtcopy_decrypt          287_eventhandling2_negpower.txt
36107: 002_pas_russianroulette.txtcopy_encrypt          288_bitblt.txt
36108: 003_pas_motion.txt                               288_bitblt_resize.txt
36109: 003_pas_motion.txtcopy                           289_regression.txt
36110: 004_pas_search.txt                               289_regression2.txt
36111: 004_pas_search_replace.txt                       290_bestofbox.txt
36112: 004_search_replace_allfunctionlist.txt           290_bestofbox2.txt
36113: 005_pas_oodesign.txt                             290_bestofbox3.txt
36114: 005_pas_shelllink.txt                            291_3sort_visual_thread.txt
36115: 006_pas_oobatch.txt                              292_refactoring2.txt
36116: 007_pas_streamcopy.txt                           293_bold_utils.txt
36117: 008_EINMALEINS_FUNC.TXT                          293_ib_utils.txt
36118: 008_explanation.txt                              293_ib_utils_timetest.txt
36119: 008_pas_verwechselt.txt                          294_maxcalc_demo.txt
36120: 008_pas_verwechselt_ibz_bern_func.txt            294_maxcalc_demo2.txt
36121: 008_stack_ibz.TXT                                295_easter_calendar.txt
36122: 009_pas_umlrunner.txt                            295_easter_calendar2.txt
36123: 009_pas_umlrunner_all.txt                        295_easter_combobox.txt
36124: 009_pas_umlrunner_componenttest.txt              297_atomimage.txt
36125: 009_pas_umlrunner_solution.txt                   297_atomimage2.txt
36126: 009_pas_umlrunner_solution_2step.txt             297_atomimage3.txt
36127: 010_pas_oodesign_solution.txt                    297_atomimage4.txt
36128: 011_pas_puzzlepas_defect.txt                     297_maxonmotor.TXT
36129: 012_pas_umlrunner_solution.txt                   297_maxon_atomimage9.txt
36130: 012_pas_umlrunner_solution2.txt                  298_bitblt_animation.txt
36131: 013_pas_linenumber.txt                           298_bitblt_animation2.txt
36132: 014_pas_primetest.txt                            298_bitblt_animation3.txt
36133: 014_pas_primetest_first.txt                      298_bitblt_animation4.txt
36134: 014_pas_primetest_sync.txt                       298_bitblt_animation4_screensaver.txt
36135: 015_pas_designbycontract.txt                     298_bitblt_animation5_screensaver.txt
36136: 015_pas_designbycontract_solution.txt            299_animation.txt
36137: 016_pas_searchrec.txt                            299_animationmotor_arduino.txt
36138: 017_chartgen.txt                                 299_animation_formprototype.txt
36139: 018_data_simulator.txt                           299_realtimeclock_arduino.txt
36140: 019_dez_to_bin.txt                               299_realtimeclock_arduino2.txt
36141: 019_dez_to_bin_grenzwert_ibz.txt                 300_treeview.txt
36142: 020_proc_feedback.txt                            300_treeview_test.txt
36143: 021_pas_symkey.txt                               300_treeview_test2.txt
36144: 021_pas_symkey_solution.txt                      300_treeview_test3.txt
36145: 022_pas_filestreams.txt                          301_LED_Arduino3.txt
36146: 023_pas_find_searchrec.txt                       301_led_arduino3_simple.txt
36147: 023_pas_pathfind.txt                             301_led_arduino3_simplecode.txt
```

36148: 024_pas_TFileStream_records.txt
36149: 025_prime_direct.txt
36150: 026_pas_memorystream.txt
36151: 027_pas_shellexecute_beta.txt
36152: 027_pas_shellexecute_solution.txt
36153: 028_pas_dataset.txt
36154: 029_pas_assignfile.txt
36155: 029_pas_assignfile_dragndropexe.txt
36156: 030_palindrome_2.txt
36157: 030_palindrome_tester.txt
36158: 030_pas_recursion.txt
36159: 030_pas_recursion2.txt
36160: 031_pas_hashcode.txt
36161: 032_pas_crc_const.txt
36162: 033_pas_cipher.txt
36163: 033_pas_cipher_def.txt
36164: 033_pas_cipher_file_2_solution.txt
36165: 034_pas_soundbox.txt
36166: 035_pas_crcscript.txt
36167: 035_pas_CRCscript_modbus.txt
36168: 036_pas_includetest.txt
36169: 036_pas_includetest_basta.txt
36170: 037_pas_define_demo32.txt
36171: 038_pas_box_demonstrator.txt
36172: 039_pas_dllcall.txt
36173: 040_paspointer.txt
36174: 040_paspointer_old.txt
36175: 041_pasplotter.txt
36176: 041_pasplotter_plus.txt
36177: 042_pas_kgv_ggt.txt
36178: 043_pas_proceduretype.txt
36179: 044_pas_14queens_solwith14.txt
36180: 044_pas_8queens.txt
36181: 044_pas_8queens_sol2.txt
36182: 044_pas_8queens_solutions.txt
36183: 044_queens_performer.txt
36184: 044_queens_performer2.txt
36185: 044_queens_performer2tester.txt
36186: 045_pas_listhandling.txt
36187: 046_pas_records.txt
36188: 047_pas_modula10.txt
36189: 048_pas_romans.txt
36190: 049_pas_ifdemo.txt
36191: 049_pas_ifdemo_BROKER.txt
36192: 050_pas_primetest2.txt
36193: 050_pas_primetester_thieves.txt
36194: 050_program_starter.txt
36195: 050_program_starter_performance.txt
36196: 051_pas_findtext_solution.txt
36197: 052_pas_text_as_stream.txt
36198: 052_pas_text_as_stream_include.txt
36199: 053_pas_singleton.txt
36200: 054_pas_speakpassword.txt
36201: 054_pas_speakpassword2.txt
36202: 054_pas_speakpassword_searchtest.txt
36203: 055_pas_factorylist.txt
36204: 056_pas_demeter.txt
36205: 057_pas_dirfinder.txt
36206: 058_pas_filefinder.txt
36207: 058_pas_filefinder_pdf.txt
36208: 058_pas_filefinder_screview.txt
36209: 058_pas_filefinder_screview2.txt
36210: 058_pas_filefinder_screview3.txt
36211: 059_pas_timertest.txt
36212: 059_pas_timertest_2.txt
36213: 059_pas_timertest_time_solution.txt
36214: 059_timerobject_starter2.txt
36215: 059_timerobject_starter2_ibz2_async.txt
36216: 059_timerobject_starter2_uml.txt
36217: 059_timerobject_starter2_uml_main.txt
36218: 059_timerobject_starter4_ibz.txt
36219: 060_pas_datefind.txt
36220: 060_pas_datefind_exceptions2.txt
36221: 060_pas_datefind_exceptions_CHECKTEST.txt
36222: 060_pas_datefind_fulltext.txt
36223: 060_pas_datefind_plus.txt
36224: 060_pas_datefind_plus_mydate.txt
36225: 061_pas_randomwalk.txt
36226: 061_pas_randomwalk_plus.txt
36227: 062_paskorrelation.txt
36228: 063_pas_calculateform.txt
36229: 063_pas_calculateform_2list.txt
36230: 064_pas_timetest.txt
36231: 065_pas_bitcounter.txt
36232: 066_pas_eliza.txt
36233: 066_pas_eliza_include_sol.txt
36234: 067_pas_morse.txt
36235: 068_pas_piezo_sound.txt
36236: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT

301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT
315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt

36237: 069_my_LEDBOX.TXT
36238: 069_pas_ledmatrix.txt
36239: 069_pas_LEDMATRIX_Alphabet.txt
36240: 069_pas_LEDMATRIX_Alphabet_run.txt
36241: 069_pas_LEDMATRIX_Alphabet_tester.txt
36242: 069_PAS_LEDMATRIX_COLOR.TXT
36243: 069_pas_ledmatrix_fixedit.txt
36244: 069_pas_LEDMATRIX_soundbox.txt
36245: 069_pas_LEDMATRIX_soundbox2.txt
36246: 069_Richter_MATRIX.TXT
36247: 070_pas_functionplot.txt
36248: 070_pas_functionplotter2.txt
36249: 070_pas_functionplotter2_mx4.txt
36250: 070_pas_functionplotter2_tester.txt
36251: 070_pas_functionplotter3.txt
36252: 070_pas_functionplotter4.txt
36253: 070_pas_functionplotter_digital.txt
36254: 070_pas_functionplotter_elliptic.txt
36255: 070_pas_function_helmholtz.txt
36256: 070_pas_textcheck_experimental.txt
36257: 071_pas_graphics.txt
36258: 071_pas_graphics_drawsym.txt
36259: 071_pas_graphics_drawsym_save.txt
36260: 071_pas_graphics_random.txt
36261: 072_pas_fractals.txt
36262: 072_pas_fractals_2.txt
36263: 072_pas_fractals_blackhole.txt
36264: 072_pas_fractals_perfomance.txt
36265: 072_pas_fractals_perfomance_new.txt
36266: 072_pas_fractals_perfomance_sharp.txt
36267: 072_pas_fractals_performance.txt
36268: 072_pas_fractals_performance_mX4.txt
36269: 073_pas_forms.txt
36270: 074_pas_chartgenerator.txt
36271: 074_pas_chartgenerator_solution.txt
36272: 074_pas_chartgenerator_solution_back.txt
36273: 074_pas_charts.txt
36274: 075_bitmap_Artwork2.txt
36275: 075_pas_bitmappuzzle.txt
36276: 075_pas_bitmappuzzle24.prod.txt
36277: 075_pas_bitmappuzzle2_prod.txt
36278: 075_pas_bitmappuzzle3.txt
36279: 075_pas_bitmapsolve.txt
36280: 075_pas_bitmap_Artwork.txt
36281: 075_pas_puzzlepas_solution.txt
36282: 076_pas_3dcube.txt
36283: 076_pas_circle.txt
36284: 077_pas_mmshow.txt
36285: 078_pas_pi.txt
36286: 079_pas_3dcube_animation.txt
36287: 079_pas_3dcube_animation4.txt
36288: 079_pas_3dcube_plus.txt
36289: 080_pas_hanoi.txt
36290: 080_pas_hanoi2.txt
36291: 080_pas_hanoi2_file.txt
36292: 080_pas_hanoi2_sol.txt
36293: 080_pas_hanoi2_tester.txt
36294: 080_pas_hanoi2_tester_fast.txt
36295: 080_pas_hanoi3.txt
36296: 081_pas_chartist2.txt
36297: 082_pas_biorythmus.txt
36298: 082_pas_biorythmus_solution.txt
36299: 082_pas_biorythmus_solution_3.txt
36300: 082_pas_biorythmus_test.txt
36301: 083_pas_GITARRE.txt
36302: 083_pas_soundbox_tones.txt
36303: 084_pas_waves.txt
36304: 085_mxsinus_logo.txt
36305: 085_sinus_plot_waves.txt
36306: 086_pas_graph_arrow_heart.txt
36307: 087_bitmap_loader.txt
36308: 087_pas_bitmap_solution.txt
36309: 087_pas_bitmap_solution2.txt
36310: 087_pas_bitmap_subimage.txt
36311: 087_pas_bitmap_test.txt
36312: 088_pas_soundbox2_mp3.txt
36313: 088_pas_soundbox_mp3.txt
36314: 088_pas_sphere_2.txt
36315: 089_pas_gradient.txt
36316: 089_pas_maxland2.txt
36317: 090_pas_sudoku4.txt
36318: 090_pas_sudoku4_2.txt
36319: 091_pas_cube4.txt
36320: 092_pas_statistics4.txt
36321: 093_variance.txt
36322: 093_variance_debug.txt
36323: 094_pas_daysold.txt
36324: 094_pas_stat_date.txt
36325: 095_pas_ki_simulation.txt

339_ntfs.txt
340_docutype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_bigInt.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT
362_maxonmotor2.TXT
363_compress_services.txt
363_compress_services2.txt
364_pdf_services.txt
365_memorystream.txt
365_memorystream2.txt
365_memorystream_test.txt
365_U_HexView.txt
366_mp3player.txt
366_mp3player2.txt
366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maXbook_v4tester.txt
372stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefunctesterfilemon.txt
380_coolfunc.txt
380_coolfunc2.txt
380_coolfunc_tester.txt
381_bitcoin_simulation.txt
382_GRMath.TXT
382_GRMath_PI_Proof.TXT
382_GRMath_Riemann.TXT
383_MDAC_DCOM.txt
384_TeamViewerID.TXT
386_InternetRadio.TXT
387_fulltextfinder.txt
387_fulltextfinder_cleancode.txt
387_fulltextfinder_fast.txt

```
36326: 096_pas_geisen_problem.txt
36327: 096_pas_montyhall_problem.txt
36328: 097_lotto_proofofconcept.txt
36329: 097_pas_lottocombinations_beat_plus.txt
36330: 097_pas_lottocombinations_beat_plus2.txt
36331: 097_pas_lottocombinations_universal.txt
36332: 097_pas_lottosimulation.txt
36333: 098_pas_chartgenerator_plus.txt
36334: 099_pas_3D_show.txt
36335: 200_big_numbers.txt
36336: 200_big_numbers2.txt
36337: 201_streamload_xml.txt
36338: 202_systemcheck.txt
36339: 203_webservice_simple_intftester.txt
36340: 204_webservice_simple.txt
36341: 205_future_value_service.txt
36342: 206_DTD_string_functions.txt
36343: 207_ibz2_async_process.txt
36344: 208_crc32_hash.txt
36345: 209_cryptohash.txt
36346: 210_public_private.txt
36347: 210_public_private_cryptosystem5_ibz_spektrum.txt
36348: 210_public_private_cryptosystem.txt
36349: 211_wipe_pattern.txt
36350: 211_wipe_pattern2.txt
36351: 211_wipe_pattern_solution.txt
36352: 212_pas_statisticmodule4.TXT
36353: 212_pas_statisticmoduletxt.TXT
36354: 212_statisticmodule4.txt
36355: 213_pas_BBP_Algo.txt
36356: 214_mxdocudemo.txt
36357: 214_mxdocudemo2.txt
36358: 214_mxdocudemo3.txt
36359: 215_hints_test.TXT
36360: 216_warnings_test.TXT
36361: 217_pas_heartbeat.txt
36362: 218_biorythmus_studio.txt
36363: 219_cipherbox.txt
36364: 219_crypt_source_comtest_solution.TXT
36365: 220_cipherbox_form.txt
36366: 220_cipherbox_form2.txt
36367: 221_bcd_explain.txt
36368: 222_memoform.txt
36369: 223_directorybox.txt
36370: 224_dialogs.txt
36371: 225_sprite_animation.txt
36372: 226_ASCII_Grid2.TXT
36373: 227_animation.txt
36374: 227_animation2.txt
36375: 228_android_calendar.txt
36376: 229_android_game.txt
36377: 229_android_game_tester.txt
36378: 230_DataProvider.txt
36379: 230_DataSetProvider.txt
36380: 230_DataSetXMLBackupScholz.txt
36381: 231_DBGrid_access.txt
36382: 231_DBGrid_XMLaccess.txt
36383: 231_DBGrid_XMLaccess2.txt
36384: 231_DBGrid_XMLaccess_locatetester.txt
36385: 231_DBGrid_XML_CDS_local.txt
36386: 232_outline.txt
36387: 232_outline_2.txt
36388: 233_modular_form.txt
36389: 234_debugoutform.txt
36390: 235_fastform.TXT
36391: 236_componentpower.txt
36392: 236_componentpower_back.txt
36393: 237_pas_4forms.txt
36394: 238_lottogen_form.txt
36395: 239_pas_sierpinski.txt
36396: 239_pas_sierpinski2.txt
36397: 240_unitGlobal_tester.txt
36398: 241_db3_sql_tutorial.txt
36399: 241_db3_sql_tutorial2.txt
36400: 241_db3_sql_tutorial2fix.txt
36401: 241_db3_sql_tutorial3.txt
36402: 241_db3_sql_tutorial3connect.txt
36403: 241_db3_sql_tutorial3_fptest.txt
36404: 241_RTL_SET2.txt
36405: 241_RTL_SET2_tester.txt
36406: 242_Component_Control.txt
36407: 243_tutorial_loader.txt
36408: 244_script_loader_loop.txt
36409: 245_formapp2.txt
36410: 245_formapp2_tester.txt
36411: 245_formapp2_testerX.txt
36412: 246_httpapp.txt
36413: 247_datecalendar.txt
36414: 248_ASCII_Grid2_sorted.TXT
387_fulltext_getscripttest.txt
388_TCPServerSock.TXT
388_TCPServerSock2.TXT
388_TCPServerSockClient.TXT
389_TAR_Archive.TXT
389_TAR_Archive_test.TXT
389_TAR_Archive_test2.TXT
390_Callback3.TXT
390_Callback3Rec.TXT
390_CallbackClean.TXT
390_StringlistHTML.TXT
391_ToDo_List.TXT
392_Barcode.TXT
392_Barcode2.TXT
392_Barcode23.TXT
392_Barcode2scholz.TXT
392_Barcode3scholz.TXT
393_QRCode.TXT
393_QRCode2.TXT
393_QRCode2Direct.TXT
393_QRCode2DirectIndy.TXT
393_QRCode2Direct_detlef.TXT
393_QRCode3.TXT
394_networkgraph.TXT
394_networkgraph_depwalkutilstest.TXT
394_networkgraph_depwalkutilstest2.TXT
395_USBController.TXT
396_Sort.TXT
397_Hotlog.TXT
397_Hotlog2.TXT
398_ustrings.txt
399_form_templates.txt
400_fplottchart.TXT
400_fplottchart2.TXT
400_fplottchart2teetest.TXT
400_QRCodeMarket.TXT
401_tfilerun.txt
402_richedit2.txt
403_outlookspy.txt
404_simplebrowser.txt
405_datefinder_today.txt
406_portscan.txt
407_indydemo.txt
408_testroboter.txt
409_excel_control.txt
410_keyboardevent.txt
411_json_test.txt
412_Zeosutils.txt
413_listview2.txt
414_avrdude_flash.txt
415_avrdude_writehex.txt
416_sonar_startscriptEKON.TXT
416_sonar_startscriptEKON_reporting.TXT
417_GRMath_PI_Proof2.TXT
418_functional_paradigm.txt
419_archimedes_spiral.txt
419_archimedes_spiral2.txt
420_archimedes_arduino.txt
420_Lissajous.txt
421_PI_Power.TXT
421_PI_Power2.TXT
422_world_bitboxx.txt
423_game_of_life.TXT
423_game_of_life2.TXT
423_game_of_life3.TXT
423_game_of_life3_test.TXT
423_game_of_life4.TXT
423_game_of_life4_kryptum.TXT
424_opengl_tester.txt
425_reversi_game.txt
426_IBUtils.TXT
427_IBDatabase.TXT
428_SortGrid.TXT
429_fileclass.txt
430_fileoperation.txt
430_fileoperation_tester.txt
431_performance_index.txt
432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
```

36415: 249_picture_grid.TXT
36416: 250_tipsandtricks2.txt
36417: 250_tipsandtricks3.txt
36418: 250_tipsandtricks3api.txt
36419: 250_tipsandtricks3_admin_elevation.txt
36420: 250_tipsandtricks3_tester.txt
36421: 250_tipsandtricks4_tester.txt
36422: 250_tipsandtricks4_tester2.txt
36423: 251_compare_noise_gauss.txt
36424: 251_whitenoise.txt
36425: 251_whitenoise2.txt
36426: 252_hilbert_turtle.txt
36427: 252_pas_hilbert.txt
36428: 253_opearatingsystem3.txt
36429: 254_dynarrays.txt
36430: 255_einstein.txt
36431: 256_findconsts_of_EXE.txt
36432: 256_findfunctions2_of_EXE.txt
36433: 256_findfunctions2_of_EXEaverp.txt
36434: 256_findfunctions2_of_EXEspec.txt
36435: 256_findfunctions3.txt
36436: 256_findfunctions_of_EXE.txt
36437: 257_AES_Cipher.txt
36438: 258_AES_cryptobox.txt
36439: 258_AES_cryptobox2.txt
36440: 258_AES_cryptobox2_passdlg.txt
36441: 259_AES_crypt_directory.txt
36442: 260_sendmessage_2.TXT
36443: 260_sendmessage_beta.TXT
36444: 261_probability.txt
36445: 262_mxoutputdemo4.txt
36446: 263_async_sound.txt
36447: 264_vclutils.txt
36448: 264_VCL_utils2.txt
36449: 265_timer_API.txt
36450: 266_serial_interface.txt
36451: 266_serial_interface2.txt
36452: 266_serial_interface3.txt
36453: 267_ackermann_rec.txt
36454: 267_ackermann_variants.txt
36455: 268_DBGrid_tree.txt
36456: 269_record_grid.TXT
36457: 270_Jedi_FunctionPower.txt
36458: 270_Jedi_FunctionPowertester.txt
36459: 271_closures_study.txt
36460: 271_closures_study_workingset2.txt
36461: 272_pas_function_show.txt
36462: 273_pas_function_show2.txt
36463: 274_library_functions.txt
36464: 275_turtle_language.txt
36465: 275_turtle_languagehouse2_time2.txt
36466: 275_turtle_language_save.txt
36467: 276_save_algo.txt
36468: 276_save_algo2.txt
36469: 277_functionsfor39.txt
36470: 278_DB_Dialogs.TXT
36471: 279_hexer2.TXT
36472: 279_hexer2macro.TXT
36473: 279_hexer2macroback.TXT
36474: 280_UML_process.txt
36475: 280_UML_process_knabe2.txt
36476: 280_UML_process_knabe3.txt
36477: 280_UML_process_TIM_Botzenhardt.txt
36478: 280_UML_TIM_Seitz.txt
36479: 281_picturepuzzle.txt
36480: 281_picturepuzzle2.txt
36481: 281_picturepuzzle3.txt
36482: 281_picturepuzzle4.txt
36483: 479_inputquery.txt
36484: 480_regex_pathfinder2.txt
36485: 482 processPipe.txt
36486: 483_PathFuncTest_mX.txt
36487: 485_InnoFunc.txt
36488: 487_asyncKeyState.txt
36489: 489_simpleComport.txt
36490: 491_analogmeter.txt
36491: 493_gadgets.txt
36492: 496_InstallX.txt
36493: 498_UnitTesting.txt
36494: 500_diceoflifes.txt
36495: 502_findalldocs.txt
36496: 504_fileclass.txt
36497: 506_colormatrix.txt
36498: 508_simplecomportmorse.txt
36499: 509_509_GEOMap2_SReverse.TXT
36500: 511_LEDLabel.txt
36501: 513_StreamIntegration.txt
36502: 515_ledclock3.txt
36503: 517_animation7.txt

437_WinAPItop_Firebirdtester.txt
438_OvcInternational.txt
439_AsyncFreeDemo.txt
439_AsyncFreeDemoForm.txt
440_DLL_Tutor.txt
440_DLL_Tutor2.txt
440_XML_Tutor.txt
440_XML_Tutor2.txt
441_make_app.txt
442_arduino_rgb_led.txt
443_webserver_arduino_rgb_light.txt
443_webserver_arduino_rgb_light4.txt
444_webserver_arduino3ibz_rgb_led_basta.txt
445_datagrid.txt
445_datagrid2.txt
445_datagrid_android_arduino.txt
446_arduino_timer.txt
447_patternFrm_mx3.txt
448_Synapse.txt
448_Synapse2.txt
449_dweb_start_tester.txt
450_Synapse_HTTPS.txt
450_Synapse_Mime.txt
450_Synapse_ScanPing.txt
451_ocx_player.txt
451_OCX_WinPlayer2.txt
452_dbtreeview.txt
452_dbtreeview2.txt
453_stdfuncs.txt
454_fileStream.txt
455_functionfun.txt
455_functionfun2.txt
455_functionfun2_test.txt
457_ressource_grid.txt
458_atomimageX.txt
459_cindyfunc.txt
459_cindyfunc2.txt
460_TopTenFunctions.txt
461_sqlform_calwin.txt
462_caesarcipher.txt
463_global_exception.txt
464_function_procedure.txt
464_function_procedure2.txt
464_function_procedure3.txt
465_U_HexView.txt
466_moon.txt
466_moon_inputquery.txt
4671_cardmagic.txt
467_helmholtz_graphic.txt
468_URLMon.txt

468_URLMon2.txt
469_formarrow.txt
469_formarrow_datepicker.txt
469_formarrow_datepicker_ibz_result.txt
469_ibzresult.txt
470_DFFUtils_compiled.txt
470_DFFUtils_ScrollingLED.txt
470_Oscilloscope.txt
470_Oscilloscope_code.txt
471_cardmagic.txt
471_cardmagic2.txt
472_allcards.TXT
473_comboset.txt
474_wakeonlan.txt
474_wakeonlan2.txt
476_getscripttest.txt
477_filenameonly.txt
480_regex_pathfinder.txt
481_processList.txt
482 processPipeGCC.txt
484_filefinder3.txt
486_VideoGrabber.txt
488_asyncTerminal.txt
490_webCamproc.txt
492_snowflake2.txt
495_fourierfreq.txt
497_LED.txt
499_mulu42.txt
501_firebird_datasnap_tests.txt
503_led_switch.txt
505_debug.txt
507_derutils.txt
509_GEOMap2.txt
510_510_bonn_gpsdata_mX4.pas
512_LED_moon.txt
514_LED_moon2.txt
516_mapview.txt
518_sensors_meter.txt

```
36504: 519_powtils.txt                                    520_run_bytecode.txt
36505: 521_iputils2.txt                                   522_getgeocode.txt
36506: 523_NMEA.txt                                       524_NAV_Utils.txt
36507: 525_GEO84s.txt                                     526_Compass_meter.txt
36508: 527_GPSDemo.txt                                    528_linescount.txt
36509: 529_profilertest.txt                               530_3DLab.txt
36510: 531_profilertest.txt                               532_mcicommand.txt
36511: 533_syncasync_demo.txt                             534_arduino_cockpit.TXT
36512: 535_Battleship3.pas                                536_ressource_grid2.txt
36513: 537_iniplus.TXT                                    538_shellbatch.txt
36514: 539_timeturtle123.txt                              540_NeuralNetwork.pas
36515: 541_webserver_arduino_motorturtle.txt              542_arduino_sound.txt
36516: 543_MATH_TurboP.PAS                                544_UTIL01.PAS
36517: 545_strips.TXT                                     546_fourier3.pas
36518: 547_regexmaster.TXT                                548_STExpressions.TXT -Services
36519: 549_3D_Panorama.txt                                550_Expressions.TXT - 550_ADO_OLEDB.txt
36520: 551_ArduinoTester.txt                              552_WaitExec32.txt
36521: 553_ArduinoCockBit3.txt                            554_Watdchdog.txt -   555_CODEsign2.txt
36522: 556_stringlistrandom.TXT                           557_4dice2015.txt
36523: 558_highrestimer.TXT  -559_higthrestimer2          560_PSUtils.TXT
36524: 561_newfunctions399160.txt                         562_shellctrldemo.txt
36525: 563_moonpaper.txt                                  564_queryperformance.txt
36526: 565_ConsoleCapture.txt                             566_queryperformance2.txt
36527: 567_SquareWordGrids2.txt                           568_U_BigFloatTestscript2.pas
36528: 569_keylog.txt - 569_ServiceMgr2.TXT               570_turingspeech.txt
36529: 571_myPing.txt                                     572_shellctrlplus.txt
36530: 573_modbusfrm_Main.pas                             574_arduino_cockpit5.TXT
36531: 575_TARTARUGA_Desktop.txt                          576_outlineEX11.PAS
36532: 577_listbox2list.txt                               578_access_db_logsimulation3.txt
36533: 579_numbersystems_sort.txt                         580_indystacksearch_geo.txt
36534: 581_stringstream.txt                               582_indystackwin.txt
36535: 583_VirtualConstructor_savereport.txt              584_ProcessList2fontwidth.txt last
36536: 585_fulltextfinder_cleancode_override.txt          586_STRandom.txt
36537: 587_one_function.txt                               588_XSBuiltins.txt
36538: 589_avi_animate.txt                                590_HTML_to_RTF.txt
36539: 591_emailattach.txt                                592_getTypeLibList.txt
36540: 593_round_time.txt                                 594_check_creditcard.txt
36541: 595_check_memory.txt                               596_time_delays.txt
36542: 597_ole_commands.txt                               598_software_list.txt
36543: 599_bug.txt                                        600_surprise_nice.txt
36544: 601_PECheckSum.txt                                 602_multilang_game.txt (moonbug)
36545: 603_cupids_arrow.TXT                               604_GEOCodeReverse4.TXT
36546: 605_maxonmotor3DTage2BASTA2015.TXT                 606_U_FibonacciSunflower.TXT
36547: 607_DataSetProvider_CDS_ADO.txt                    608_ColorMixer_Arduino2.txt
36548: 609_ScriptExecutor (beta)                          610_3D_DLL.txt
36549: 611_Arduino_COMOutputs.txt                         612_MTerminal.pas
36550: 611_Arduino_COMOutputs_TMP36_3.txt
36551: 613_uPSI_DIUtils_test.pas                          614_inbrowserock.txt
36552: 615_SONAR_51_Starter.txt                           616_ComTerminalDlg.pas
36553: (589_AVI_Throbber.res.txt)                         589_avi_animate.txt
36554: 616_ComTerminalDlg.pas                             //end for 3.9.9.195
36555:
36556: 617_API_coding_tut37.txt                           617_API_coding_tut39.txt
36557: 618_bayes_filter.txt                               618_bayes_filter_dic.txt
36558: 618_bayes_filter_dic2.txt                          619_crypto_package_demo.txt
36559: 619_crypto_package_demo_policy.txt                 619_crypto_packr.txt
36560: 620_serialtimer.txt                                621_bruteforceattack_dic.txt
36561: 622_netsh_master_restore4.psb                      622_netsh_master_sonar.pas
36562: 623_exspawnu.pas                                   623_exspawnu_buffoverflow.pas
36563: 624_Misctest_lexscanner.pas                        624_Misctest_lexscanner2.pas
36564: 625_binominal_bigint3_testcase.txt                 626_lorentztransformation.TXT
36565: 627_4gewinnt_main3.txt                             628_FormatFloat.pas
36566: 629_Word_FORMULA.PAS                               630_multikernel2.TXT
36567: 630_multikernel3.TXT                               631_dmath_testfunc_mx3.pas
36568: 632_safecall12.txt                                 632_safecheck2.txt
36569: 632_safecheck2_regex_basta.txt                     633_gamma_func.txt
36570: 634_briefcase_XMLDB.xml                            634_BriefcaseMain2.pas
36571: 634_maxbase3server.mdb                             635_bitcount_bytecodestudy.txt
36572: 636_rest_apiexamples.txt                           637_psnconvert3.TXT
36573: 637_psnconvert4.TXT                                637_startscript_psnconvert2.TXT
36574: 638_Mathslib_prime.pas.txt                         639_prime_factors.txt
36575: 640_rest_geocode.txt                               640_rest_weather_report.txt
36576: 640_weather_cockpit5.TXT                           640_weather_cockpit6.TXT
36577: 641_voicecommand.TXT                               642_BDE_export.PAS
36578: 643_EKON19_Tester3_OLE_Reg.TXT                     644_Synapse_Mimetest_regEx3.txt
36579: 645_age_guess_REX.pas                              646_pi_evil.TXT
36580: 646_pi_evil2.TXT                                   647_StringtoHTMLBasta.TXT
36581: 648_SYNEdit_ExportDemo.TXT                         649_dayio.pas
36582: 649_StringlistHTMLtestBasta.TXT                    650_sched.pas
36583: 650_http_server2tester.txt                         650_time_routines.txt
36584: 650_dbblober.txt                                   650_drawdice2016.txt
36585: 651_StrUtilmaxbase.pas                             652_graph3DMainUnit2.pas
36586: 653_PdoxTstbox.pas                                 654_spectrum_dice2016_3.txt
36587: 655_arduino_chess.txt                              655_arduino_chess2.txt
36588: 656_XML_RpcCommon.pas                              656_XML_RpcCommon2.pas
36589: 657_PEM_Cert.txt                                   658_ASN1_Cert.txt
36590: 659_VirtListBox.txt                                660_X509_Cert.pas
36591: 660_InetUtils_Rest.txt                             661_REXX.pas
36592: 662__VisualChronForm.pas                           663_Tokens.pas
```

```
36593: 664_DataAnnotationsValid.pas.pas
36594: 666_SysInfoCtrls.pas.pas
36595: 667_URobo2.pas.pas
36596: 669_uptime.txt.pas
36597: 670_interesting_birthday_paradox.txt
36598: 672_regex_ask_task.txt
36599: 674_pipe_graphics.txt
36600: 676_run_as_admin.txt
36601: 678_fmTraceRouteMainU.pas
36602: 680_gravity_waves.txt
36603: 682_ScreenMatrixToBitmap.pas
36604: 684_mathe4.pas
36605: 686_namedpipe_demo.txt
36606: 687_memorymapped_filestream.txt
36607: 689_proctype_alias.txt
36608: 691_httpservermain3.txt
36609: 692_imageserv_fClient33.pas
36610: 694_OverbyteIcsAsn1Utils.pas
36611: 696_kmemo_demo_Main.pas
36612: 697_OverbyteIcsTicks64.pas
36613: 698_new_classes_tester.pas
36614: 698_Bitcoin_blockchaintester.pas
36615: 700_mX422_functions_demo2.txt
36616: 700_new_function_snippets.pas
36617: 701_all_fibonacci.txt
36618: 702_X509_Cert_OpenSSL_1.txt
36619: 703_pc_sensors.txt
36620: 705_magsubs1_functions.pas
36621: 706_population_count.txt
36622: 708_XML_DOM_ReaderTutor.txt
36623: 710_enigma_riddler.txt
36624: 710_geo_distance_form_mapbox.txt
36625: 710_RegSvrUtils_mX4.pas
36626: 712_rootsfunction.txt
36627: 713_SnakeA2.pas
36628: 714_astroids_GAME3.PAS
36629: 715_shape_calculator.PAS
36630: 717_AdLog.PAS
36631: 719_rootsfunction.txt
36632: 720_HTML_TableExport_CSS.pas
36633: 721_DOSCapture_netstat2.TXT
36634: 723_remote_notepad_message.TXT
36635: 725_CRC32_live.txt
36636: 725_725_Cholesky.pas
36637: 725_blockchain_sensors3.txt
36638: 726_DBGrid_XML_CDataSet.txt
36639: 728_mci_macros.TXT
36640: 729_PrivateKey_mX.pas
36641: 730_modular_exponentiation_IBZ.txt
36642: 730_ternary_logice.txt
36643: 731_DWS_Client4_EKON20.pas
36644: 732_hailstonesequence_delphi.txt
36645: 733_jensensdevice.txt
36646: 735_Linear_congruential_generator.txt
36647: 735_Euler_p117_openSSL_scripts.txt
36648: 737_xmastree2.txt
36649: 738_mX4_exact_fishertest2.pas
36650: 739_GIFPAL.PAS
36651: 740_Globalreplacetext.txt
36652: 740_msgpump1.txt
36653: 740_uFTPServermX2_listtest.pas
36654: 740_RosettaIsaac_Cryptocode_Delphi
36655: 741_DrawYinAndYang.txt
36656: 742_VectorProducts.txt
36657: 744_Web_scraping_UTCTime.txt
36658: 745_crypto_memory.txt
36659: 746_uText.pas
36660: 747_uSteganos.pas
36661: 749_helloWebServer.txt
36662: 749_helloWebServer3.txt
36663: 750_EchoServer3.txt
36664: 210_RSA_crypto_complete8hybrid.txt
36665: 750_ibz_cryptomem_RSA_proof_64.txt
36666: 750_SimpleTCPMain.pas
36667: 751_Elevator_Simulator.pas
36668: 752_BitmapHistogram.txt
36669: 752_U_AstroDemo_mX4_2.pas
36670: 753_xbase_tester.pas
36671: 755_science_ibz_spektrum.txt
36672: 755_sedol2.txt
36673: 756_shellscripthell_javascript.txt
36674: 756_shellscripthell_javascript2.txt
36675: 667_URobo2EKON_FUN_Tracking2.pas
36676: 757_tower_of_hanoiX.txt
36677: 758_speechmachine.txt
36678: 760_Normal_distribution.txt
36679: 760_XML_DOM_OLE.TXT
36680: 760_systeminfo_tester.txt
36681: 766_wmi_GetServiceStatus.txt
665_keditcommon.pas.pas
667_URobo1.pas.pas
668_U_SoundGen3.pas.pas
700_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_fservermain3.pas
693_OverbyteIcsUtils_tester.pas
695_IdAntiFreeze_tester.pas
696_kmemo_demo.pas
697_OverbyteIcsTicks64Demo1.pas
693_OverbyteIcsDnsQuerytester.pas
699_OverbyteIcsSha1_tester.pas
700_mX422_functions_demo3.txt
700_function_snippets3_metrics.inc
702_pascal_fibonacci.txt
702_X509_Cert_OpenSSL_2.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
714_GAME_spaceinvaders.PAS
716_webplan.PAS
718_AdPerformanceCounter.PAS
720_euler_project.txt
720_selfdescribing_number.txt
722_refererHeader_monitor.TXT
724_spyware_monitor_test.TXT
725_Install_Routines.TXT
725_Voice3.TXT
725_X509_SHA1_TestCert.txt
727_Canvas_Artist.TXT
728_mci_macros_imagetest.TXT
729_U_BigIntTest_mX.pas
730_topological4.txt
730_ternary_logice2.txt
731_DWS_Server4_EKON20.pas
732_hailstonesequence.txt
734_boxthecompass.txt
735_mx_screenshot_duesseldorf.png
736_hiddenstreams_ads.txt
737_xmastree3.txt
738_mX4_exact_fishertest3.pas
739_mXbinom.pas
740_Globalreplacetext_test.txt
740_uFTPServermX2.pas
740_RosettaIsaac_cryptocode.txt
740_RosettaIsaac_cryptocode2.txt
741_DrawYinAndYang_console.txt
743_ProperDivisor.txt
745_Web_scraping_UTCTime_DNSQuery.txt
745_euler_method.txt
746_mx_screenshot_stream.png
748_ubits.pas
749_helloWebServer2.txt
749_helloWebServer3_tempsensor.txt
750_EchoServer4.txt
210_public_private_crypto_complete7hybrid.txt
750_RSA_Toolproof4.txt
750_SimpleTCPMainForm.pas
751_thelift_simulation_uMain.pas
752_U_ConvertTest2Astronomy.pas
752_U_AstroDemo_mX4_2Form.pas
754_U_Find3Primes.pas
755_science_ibz_spektrum2.txt
755_bigint_demo.txt
756_shellscripthell_saferemove.txt
strasbourgoutdoor2.bmp
757_PowerEfficiencyDiagnostics.txt
757_factorizeX.txt
759_euler_ppk_mutex_ibz.txt
xdata.xml - cd_catalog.xml
760_XML_DOM_OLE_bestofbox4.TXT
751_Elevator_Simulator2.pas
751_Elevator_Simulator4.pas
```

```
36682: 760_systeminfo_tester2.txt
36683: 761_BoulesBillard3.txt
36684: 760_systeminfo_tester2.txt
36685: 761_BoulesBillard3.txt
36686: 762_regex_analytica.txt
36687: 762_matlab.txt
36688: 764_reverse.txt
36689: 765_codesign.txt
36690: 766_wmi_management.txt
36691: 767_mint_geo_map.txt
36692: 769_tzutil.pas
36693: 770_pascal_coin_api_ascii_talk.txt
36694: 770_U_ParsingDemo3_ASCIITalk.pas
36695: 770_3hires_benchmarktimer3.txt
36696: 771_fibonacci_first.txt
36697: 773_streamfibonacci_first.txt
36698: 774_findwindow_calc.txt
36699: 776_fiboseq_state_android_pi.txt
36700: 777_netportscan.txt
36701: 777_tictactoe2.txt
36702: 777_tictac4picboard_16F887_arduino.txt
36703: 778_advapi32_dll_SHA256.txt
36704: 779_drivemanager.txt
36705: 780_mandelbrotset_max.txt
36706: 780_set_enumeration.txt
36707: 780_newfunctions_calctest_42610.txt
36708: 780_abc_problem.txt
36709: 780_mandelbrotset_max2.txt
36710: 780_thingspeak_mathworks.txt
36711: 780_JSONDemo.txt
36712: 780_ppk_xmltransform.txt
36713: 780_xrtl_net_URIUtils.pas
36714: 781_ProxyUtils.txt
36715: 781_ProxyUtils2performance.txt
36716: 782_XML_DOM_ADO_API_REST_demo.txt
36717: 783_turtle_sign.txt
36718: 785_Matrix_Transpose.txt
36719: 787_cologne_Bitmap21.txt
36720: 789_DOM_XML.txt
36721: 791_uFTPServer.pas
36722: 793_uOneInstance.pas
36723: 795_uOutput.pas
36724: 797_SynEditCodeFolding_routines.pas
36725: 799_list_duplicates.txt
36726: 800_isodatetime.txt
36727: 802_indystack_geo2_picture.txt
36728: 803_textanalysis2.txt
36729: 805_webspider_Mainconsole.pas
36730: 805_webspider_Mainconsole2_2.pas
36731: 807_FANN_XorSample.pas
36732: 807_FANN_XorSample2.pas
36733: 810_bullcow_mastermind.txt
36734: 810_neuralnetwork2.txt
36735: 810_neuralnetwork21.txt
36736: 811_mXpcatest_dmath_datascience2.pas
36737: 813_PCA_datascience_iris.txt
36738: 814_bigmulu.txt
36739: 813_PCA_datascience_iris3.txt
36740: 815_look_and_say.txt
36741: 816_SnowballProgram.pas
36742: 818_spider_traversialdemo.txt
36743: 819_ES.Utils.pas
36744: 820_U_ProbabilityDist.pas
36745: 820_U_ProbabilityDist2.psb
36746: 820_bostondataset_socks4test.pas
36747: 820_bostondataset_socks4test_irisset.pas
36748: 820_bostondataset_irisset_clean.pas
36749: 821_Narcissistic.txt
36750: 822_perceptron1_2.htm
36751: 822_perceptron1_2.txt
36752: 822_perceptron1_3.htm
36753: 807_FANN_PerceptronSample.pas
36754: 822_perceptron_for_pascal2.txt
36755: 822_perceptron_for_pascal2regression.txt
36756: 824_catamorphism.txt
36757: 825_uTPLb_Hash_tester.pas
36758: 826_sentiment_api2_1.txt
36759: 826_sentiment_api2_2.txt
36760: 820_bytecodetest.txt
36761: 826_reglin_regression_fit2.pas
36762: 827_save_UTF8.txt
36763: 828_install_routine2.txt
36764: 829_save_JSON.txt
36765: 830_RSA_cryptosystem_wiki.txt
36766: 831_regex_light.txt
36767: 833_findfactors.txt
36768: 835_fibonsteps_sequence.txt
36769: 837_penneysgame.txt
36770: 838_endofuniverse_explain.txt
760_systeminfo_tester3.txt
maxbox_billiard2.png
760_systeminfo_tester3.txt
761_maxbox_billiard2.png
762_staroffice.txt
763_google_rest_api.txt
764_reverse.htm
765_codesign2.txt
766_wmi_management2.txt
768_pascal_coin_utils.txt
770_pascal_coin_api.txt
770_U_ParsingDemo2.pas
770_3hires_benchmarktimer2.txt
770_U_ParsingDemo3_ASCIITalk_test.pas
772_mutex_instance.txt
773_key_encryptort.txt
775_geo_mapbox.txt
776_thermotacho.txt
777_keydownform_analyze.pas
777_tictactoe3.txt
777_tictactoe3tacho.txt
766_wmi_GetAntiVirusProduct2.txt
780_doubleSHA256_cipher.txt
780_maxboxmandel.png
780_set_enumeration.pdf
780_esp8266_wifiscan.txt
780_abc_problem_delphi.txt
780_DLL_functionlist.TXT
766_wmi_GetUSB_Devices.txt
735_openSSL_scripts4.txt
780_ppk_xmltransform_provider.txt
766_wmi_GetAntiVirusProduct2.txt
781_DataSetUtils.pas
780_ppk_xmltransform_provider2.txt
782_XML_DOM_ADO_API_REST_demo2.txt
784_DOM_travers.txt
786_FONTVIE1mX4.PAS
788_WMI_Transpose.txt
790_Start_Elasticsearch.pas
792_uMRU.pas
794_uFind.pas
796_uUtils.pas
798_turtle_python_time2.txt
800_wraplines.txt
801_algorithms_Devices.txt
803_textanalysis.txt
804_BackgroundWorker.pas
805_webspider_Mainconsole2.pas
806_primes_threadMain.pas
808_FANN_XorSample_traindata.pas
809_FANN_XorSample_traindata.pas
810_python_shell.txt
810_python_shell2.txt
811_mXpcatest_dmath_datascience.pas
812_smtp_email4.txt
813_PCA_datascience_iris2.txt
814_bigmulu.htm
814_machinelearning_overview.jpg
816_snowball_Readme.doc
817_logextract.txt
818_spider_traversialdemo2.txt
820_eigenvalue.txt
820_U_ProbabilityDist2.pas
820_182_Cologne42_EKON21sunset.jpeg
820_bostondataset.txt
820_iris.data
820_bostondataset3.txt
821_Narcissistic2.txt
822_perceptron.txt
822_perceptron1_3.txt
822_FANN_Perceptron_Sample_Plus.pas
822_perceptron1_31.txt
822_perceptron_for_pascal.htm
823_truth_table.txt
825_hash_class.txt
826_sentiment_api2.txt
826_sentiment_api2_1.psb
sentiment2.txt
826_reglin_regression_fit.pas
714_astroids_GAME5.PAS
828_install_routine.txt
asteroid.txt
830_save_Binaries.txt
830_RSA_cryptosystem_wiki_huge2.txt
832_euler92.txt
834_array_concat.txt
836_date_time_routines.txt
837_penneysgame2.txt
839_percolation_mean.txt
```

36771: 840_maxboxmandel2.png
36772: 840_URungeKutta4test.pas
36773: 842_U_Keno2.pas
36774: 842_U_JPGClock2.pas
36775: 844_OpenSSL_synapsetester.txt
36776: 845_Gray_Code.txt
36777: 845_TestPermutations2.txt
36778: 846_longmultiplication.pas
36779: 848_xml_intfn.pas
36780: 849_pi3.pas
36781: 810_neuralnetwork21pascal_tutor66.txt
36782: 810_neuralnetwork21pascal_tutor66predict.txt
36783: 854_U_Invertedtext.pas
36784: 855_dll_from_maXbox.pas
36785: 856_snake_maxbox4_4.png
36786: 856_snake_maxbox4_5.png
36787: 856_snake_maxbox4_6.png
36788: 856_SnakeAmX42_tab.pas
36789: 859_maputils_client.pas
36790: 860_virtualbox_overview.png
36791: 826_similarity_api2.txt
36792: 860_similarity_api3.txt
36793: 860_sexy_primes2performance.txt
36794: 860_sexy_primes2performance.psb
36795: 860_sexy_primes.txt
36796: 860_PascalMatrix_Out.txt
36797: 860_graph3.png
36798: 826_similarity_api2.txt
36799: 860_virtualbox_overview.png
36800: 859_maputils_client.pas
36801: 861_stringtokenizer.txt
36802: 861_stringtokenizer2.txt
36803: 862_benfordslaw_synedit.txt
36804: 863_stack_tcp_chatroom.TXT
36805: 865_caesar_cipher.txt
36806: 865_winservices2.txt
36807: 866_tutor70consolepowershell2.png
36808: 780_abc_problem_delphi_synapse2.txt
36809: 781_LatLonDistmX.pas
36810: 781_JLatLon_JFK-ZRH.JPG
36811: 782_U_MakeCityLocations2mX4.pas
36812: 784_checkers.txt
36813: 786_web_scrap.txt
36814: 788_PETRA.MID
36815: 789_totient_primes_tester.txt
36816: 790_sha512.txt
36817: 867_metadefener_api.txt
36818: 869_factsum.txt
36819: 871_checkemptyfolders.txt
36820: 872_RSA_codes.txt
36821: 873_Unittester_mX47.txt
36822: 873_Unittester_mX47_2.psb
36823: 874_LinkGrabUnit1.pas
36824: 875_filehistounit2app_forensic.txt
36825: 876_console_output.txt
36826: 877_flowpanel2.pas
36827: 879_psvCGI.pas
36828: 880_visualstatemachine_main.pas
36829: 880_cgiMain_server2.pas
36830: 881_odometer_uMain2.pas
36831: 883_create_dataset.txt
36832: 884_google_chart_api2.TXT
36833: 886_AmicablePairs.TXT
36834: 888_Sundial.TXT
36835: 890_Jensens_Device.TXT
36836: 892_microwaves.TXT
36837: 891_universal_turing_machine3.TXT
36838: 893_montecarlo_PI.TXT
36839: 894_csv_to_html.TXT
36840: 896_tickets_tree.png
36841: 896_all_tickets.csv
36842: 897_openoffice_ole_automation.TXT
36843: 899_remote_control_key.TXT
36844: 900_XML.TXT
36845: 901_turtle_purple_circle.txt
36846: 902_sha1_explicit_aes.TXT
36847: 904_Unit1NEMESIS.pas
36848: 905_rfc1213util.pas
36849: 905_PlotCompCode.pas
36850: 905_net_ugraph_formplot.pas
36851: 905_rfc1213sys.pas
36852: 907_OverbyteIcsSHA1Test1.pas
36853: 880_cgiMain_server3.pas
36854: 880_CGI1.exe
36855: 908_Netlister.pas
36856: 909_sinewave_circle.pas
36857: 880_cgiMain_server4.pas
36858: 910_kh_function_plot_demo2.pas
36859: 910_tictactoe_form.png

840_maxbox_art3.jpg
841_U_Alphametics.pas
842_U_JPGClock2.dfm
843_snake2.pas
844_OpenSSL_synapsetester2.txt
845_TestPermutations.txt
846_longmultiplication.psb
847_blockchain_hashexplain.pas
849_pi3.psb
850_pi3_filemodify.pas
810_neuralnetwork21python_tutor66.txt
853_U_SixOfAKind2_mX4.pas
855_dll_reflection.jpg
856_SnakeAmX4.pas
857_printer_function.txt
856_SnakeAmX42.pas
858_dragdrop_function.txt
859_maputils.pas
860_graph.png
860_brownian_tree.pas
860_graph3.png
860_benfordslaw.txt
860_dateformat.txt
860_sexy_primes.psb
860_neuralnetwork21_levenshtein_dist.txt
860_similarity_api3.txt
860_brownian_tree.pas
860_brownian_tree.pas
860_graph.png
827_save_UTF8_xml_oledb.txt
861_stringtokenizer.htm
862_socialmention_api3.txt
863_stack_tcp_chatroom_client.TXT
864_form_canvas_font_client.TXT
865_winservices.txt
866_native_console.txt
780_abc_problem_delphi_synapse.txt
780_abc_problem_delphi_synapse3.txt
781u_LatLonDistanceTestmX.pas
781u_LatLonDistanceTestmX2.pas
783_replaceall.txt
785_reviews.txt
787_Excel_ImportExport.txt
788_pcu_mX4.pas
789_totient_primes_tester.psb
790_googlemaps.txt
868_6MainFmnetworkmX47.pas
870_totient_primes_tester_471_functions.txt
496_installX.txt
872_RSA_codes2.txt
873_Unittester_mX47_2.txt
873_httpclient.png
875_filehistounit2app.txt
875_codetester_GUI_maXbox4exe.png
877_flowpanel.pas
878_code_reflection.txt
880_cgiMain_server.pas
880_ModemEmulator.pas
640_API_LayerDetection_EKON23.TXT
882_bitmap_brusher.txt
884_google_chart_api.TXT
885_IP_Geolocation.TXT
887_Anti_Primes.TXT
889_Jaro_distance.TXT
891_uturing_machine.TXT
891_uturing_machine2.TXT
892_microwaves2.TXT
893_montecarlo_PI.psb
895_screenpixel.TXT
896_textclassification.py
690_calc_pi_ex_stat4.pas
898_exit_procedure.TXT
569_ServiceMgr3.TXT
XML.TXT
902_sha1_explicit.TXT
903_ODBC_list.TXT
905_uiptools.pas
905_uipservices.pas
905_net_ugraph.pas
903_microtonal_csound.TXT
906_OverbyteIcsLogger.pas
170_4gewinnt_main3.txt
880_psvCGI.pas
880_CGI1.pas 880_DrBobCGI.pas
909_MandelBrot.pas
910_tictac.pas 910_tictac_form.pas
880_CGI1.exe
910_kh_plot.png
910_tictac_form2.pas

36860: 911_gapfulnumbers2ibz.pas
36861: 912_TTCPBlockSocket_Tester3.txt
36862: 729_U_BigIntTest_mX4.pas
36863: 913_al_web_spider_test.pas
36864: 915_OpenOffice_API.pas
36865: 916_fmTraceRouteMainU.pas
36866: 916_CGIMailerfMain.pas
36867: 918_TIdDateTimeStamp_main.pas
36868: 920_TURL_class_test.pas
36869: 920_TURL_class_test2.pas
36870: 919_uLockBox_Signatory_TestCasesmX471.pas
36871: 913_al_web_spider34form_trivial.pas
36872: 921_object_filehandling.pas
36873: 922_u_dir.pas
36874: 915_OpenOffice_API_2.pas
36875: 923_indybasicclientserverfMain.pas
36876: 925_U_DoubletsX3_2.pas
36877: 925_U_CountPhrases_form.pas
36878: 926_singlesamplepredict.py
36879: 927_Emirp.pas
36880: 927_Emirp2.pas
36881: 928_cd_catalog2.xml
36882: 927_Emirp2.pas
36883: 928_cd_catalog2.xml
36884: 930_service_tools.pas
36885: 930_service_tools2.pdf
36886: 930_service_tools2.htm
36887: 931_spiralmatrix2.txt
36888: 932_unixdict.txt
36889: 932_ordered_words42.txt
36890: 932_correlation.txt
36891: 932_correlation_regression.txt
36892: 548_STExpressions_Services2.TXT
36893: 749_helloWebServer3refresh2.txt
36894: 938_xmldemo_out.xml
36895: 937_xml_input.pas
36896: 935_Primality_by_trial_division.pas
36897: 933_draw_sphere.pas
36898: 940_primefactors.psb
36899: 941_picturebox.txt
36900: 942_gaussen.txt
36901: 942_gaussen_data_science.txt
36902: 942_statpackage2.png
36903: 942_statpackage2.png
36904: 942_gauss_jordan.pas
36905: 942_gauss_jordan2.pas
36906: 944_Perl_Regex.txt
36907: 945_Many_Regex.txt
36908: 946_sudokuform.png
36909: 945_registrysearch_processlist2.psb
36910: 947_TWinControl_Canvas.txt
36911: 949_setbit.txt
36912: 950_uniqueinstance_pythonversions.pas
36913: 950_saint_source.txt
36914: 185_memorymax3_internet.txt
36915: 942_gaussen_data_science3.txt
36916: 952_psUFinancial.pas
36917: 953_Res2BmpUtils.pas
36918: 953_JTools.pas
36919: 955_logregclassifier.py
36920: 957_blobstream.txt
36921: 958_webcam.txt
36922: 959_convert_temperature2.txt
36923: 960_newjapan.mp3
36924: 960_neuralbitmx2.pas
36925: 960_Legendre_Gaussintegration.txt
36926: 950_saint_source2.txt
36927: 963_movie_reviewd_words42.txt
36928: 963_moviereviewswordlist2listboxsort.txt
36929: 861_stringtokenizer2primes_moviesreview2.txt
36930: 363_compress_services3.txt
36931: 964_geo_map_zoom.txt
36932: 965_RSA_IBZ_PrivatePublicKey_Form.pas
36933: 965_RSA_IBZ_PrivatePublicKey_Form3p.pas
36934: 966_GUItester2.png
36935: 967_U_RSADemo2.pas
36936: 967_CryptoGUItester4.png
36937: 967_CryptoGUItester1.png
36938: 968_recordobject.txt
36939: 969_gapfulnumbers.txt
36940: 970_integrate.txt
36941: 970_integrate_vandercorput_sequence.txt
36942: 970_U_wordladder_mX4.pas
36943: 970_book_portal_usecase2.png
36944: 970_skype_firewall.png
36945: 970_U_Invertedtext.uc
36946: 970_U_Decrypt3.pas
36947: 970_U_Decrypt4Forms.pas
36948: 971_U_DrawScrambledPie.pas

912_Primestest_AKS_2.pas
912_ssl_blcksock_tester.pas
913_al_php_runner_test.pas
913_ALOpenOffice_tester.pas
897_openoffice_ole_automation2.TXT
914_run_javascript_471.pas
917_EwbCoreTools_experimantal.pas
919_uLockBox_HugeCardinalTestCases.pas
919_uLockBox_RSA_TestCases.pas
920_TimeServer_TickClock.pas
919_umfmMakeSampleKey.pas
913_al_web_spider34form.pas
922_u_stringlist.pas
922_u_c_url.pas
915_OpenOffice_API_21.pas
924_arrays.pas
925_U_CountPhrases.pas
925_phrasecount2.png
927_primetrial.pas
927_Emirp.psb
928_wave_async_parallel.pas
929_xmlstream.pas
928_wave_async_parallel.pas
929_xmlstream.pas
930_service_tools2.rtf
930_service_tools2.pas
931_spiralmatrix.txt
932_ordered_words4.txt
932_ordered_words4.psb
932_getgithub2.txt
932_correlationplot2.png
932_correlation_regression_light.txt
932_getgithub22.txt
938_xmldemo.xml
938_xml_output.pas
936_md5.pas
934_datamanipulation.pas
939_soundex.pas
940_primefactors.pas
941_picturebox2.txt
942_gaussen_data_science.psb
942_statpackage.png
942_gauss_jordan2.pas
942_gaussjordan.png
942_gaussjordan2.png
943_TEEChart.txt
945_Many_Regex.psb
946_sudokuUnit1Form.pas
945_registrysearch_processlist.txt
945_registrysearch_processlist2.txt
948_dynamic_page_control.txt
950_uniqueinstance.txt
950_ParserUtils.pas
950_saint_source.psb
185_memorymax4_internet.txt
951_dclublib.pas
952_psULib.pas
953_JConvertUtils.pas
954_jumpto_registrykey.txt
956_jscript.txt
958_foto2webcam.png
959_convert_temperature.txt
960_getsounddevices.txt
960_neuralbitmx.pas
960_neuralnetworkx.pas
961_URLDecoding.txt
962_cookiesread.txt
963_moviereviews2csv.txt
861_stringtokenizer2primes_moviesreview.txt
751_Elevator_Simulator45.pas
363_compress_services4simple.txt
965_RSA_IBZ_PrivatePublicKey.pas
965_RSA_IBZ_PrivatePublicKey_Form2.pas
966_U_DivideTest.pas
966_U_PointInSpace52_mX4Form.pas
967_CryptoGUItester3.png
967_CryptoGUItester2.png
967_U_RSADemo2forms.pas
968_recordobject2.txt
969_gapfulnumbers.psb
970_integrate.psb
970_U_Spellbound2mx.pas
970_U_ScrambledPiemX4.pas
970_U_CrossWordHelper2mX4.pas
970_crosswordhelper.png
970_U_Invertedtext.pas
970_U_Decrypt3Suggest.pas
971_CharFreqCounter.txt
971_wordscramble2.png

```
36949: 971_wordscramble_forms1.png                          970_U_ScrambledPiemX4Forms.pas
36950: 970_U_ScrambledPiemX4Forms2.pas                      970_U_Decrypt4Forms22.pas
36951: 972_words_speclinelist_hashdemocount.txt             972_U_HashedWordCount.pas
36952: 972_json_tester.txt                                  972_U_HashedWordCount2.pas
36953: 972_char_counts.txt                                  972_primenumber_routines.txt
36954: 972_json_tester2.txt                                 973_RCDATA_resource_export.txt
36955: 973_RCDATA_resource_export64.txt                     974_systemsdiagram_prepare.pas
36956: 470_U_ScrollingLEDs3mX4_21.pas                       974_64bit_Routines.txt
36957: 975_facofinteger.txt                                 975_facofinteger2.txt
36958: 975_U_PrimeFactors1.pas                              975_facofinteger2.psb
36959: 975_U_PrimeFactors2.pas                              975_U_PrimeFactors21.pas
36960: 975_euler_pot.txt                                    975_euler_pot2.txt
36961: 975_euler_pot2.psb                                   602_multilang_game2ball.txt
36962: 602_multilang_game2.txt                              602_multilang_gameball.txt
36963: 976_stripchars.txt                                   977_Rosetta_Dijkstra_Console.txt
36964: 977_Rosetta_Dijkstra_Console.htm                     978_ALHttpclient_cookie.txt
36965: 978_wininet_universaldownload.txt                    979_U_ResponseStats3.pas
36966: 979_U_DensityPlot.pas                                979_UDFFRegistry.pas
36967: 979_U_ReactionTimes4.pas                             979_testdetail.rsd
36968: 980_Chart_DensityPlot.pas                            980_testdetail.rsd
36969: 980_U_TelephoneWords.pas                             980_telephone-keypad-letters.jpg
36970: 980_telephonewords.png                               980_U_TelephoneWords2.rtf
36971: 980_U_TelephoneWords2.pas                            980_U_TelephoneWords2.htm
36972: 980_U_TelephoneWords2.psb                            980_U_TelephoneWords21.pas
36973: 980_telephonewords21.png                             980_U_TelephoneWords22.pas
36974: 980_Chart_DensityPlot2.pas                           981_tesseractutils.pas
36975: 982_win_graphviz.txt                                 983_Average_loop_length.txt
36976: 984_Drawaclock_computergraphic.txt                   984_Drawaclock_computergraphic.psb
36977: RXMisc.pas                                           985_nist_regression_eval.pas
36978: 985_nist.dat                                         985_sha1_impl.txt
36979: 985_open_subtitles.txt                               985_regex_subtitles.txt
36980: 985_sha1_impl2.txt                                   985_sha1_impl21.txt
36981: 985_sha1_impl21.psb                                  985_missing_permutation.txt
36982: 985_missing_permutation2.txt                         986_wiwin32.txt
36983: 986_findallfiles.txt                                 detector21_wdef.py
36984: detector_jupyter.ipynb                               objectdetector3.ipynb
36985: hoppenstedtstwinwiz_out2.jpg                         twinwiz_out2.jpg
36986: 987_shell32_operation.txt                            987_shell32_file_operations.txt
36987: 988_anti_aliasing.txt                                989_paintbox.txt
36988: 990_ADO_Connection.txt                               990_maxbase3.mdb
36989: 990_GUI_ADO_Output.png                               989_paintbox.png
36990: 990_ADO_Connection_Create_DB.txt                     990_SQL_Workbench_ADOCreate_Output.png
36991: 990_ADO_Connection_Create_DB2.txt                    990_maxbase4runtimedb6.mdb
36992: 990_metadefender_API.txt                             986_findallfiles_metadefender.txt
36993: 990_metadefender_API_2.txt                           990_ADO_Connection_Create_DB21.txt
36994: 990_httpsend.png                                     990_httpsend2.png
36995: 990_httpsend.pac                                     990_metadefender_API_21.txt
36996: 990_winhttprequest_API_51.txt                        990_httprequest_API.txt
36997: 991_oma_chartregression2.py                          991_python_oma_svr.txt
36998: 991_python_oma_svr21.txt                             991_python_oma_svr2.txt
36999: 991_oma_charts_market_btc_150y.png                   991_python_oma_svr21integrate.txt
37000: 840_RungeKuttaTest.exe                               840_URungeKutta4test2.pas
37001: 991_python_oma_svr21integrate_dotnetcore.txt         991_pyoma_svr22integrate_dotnetcore2.txt
37002: 992_detector21_wget_integrate.txt                    992_detector21_wget_integrate.htm
37003: 992_detector21_wget_integrate2.rtf                   992_detector21_wget_integrate2.txt
37004: 992_detector21_wget_integrate2.htm                   993_perceptron_graph2.png
37005: 993_perceptron_form.txt                              993_perceptron_graph.png
37006: 993_perceptron_form2.txt                             993_perceptron_form2.psb
37007: 994_Find_URI_in_text.txt                             995_RunLengthEncoding.txt
37008: 995_RunLengthEncoding.psb                            995_RunLengthEncoding_Pascal.txt
37009: 996_Precision_Functions.txt                          997_Base64_decode_data.txt
37010: 998_string_has_all_unique_characters.txt             998_string_has_all_unique_characters.psb
37011: 998_weatherapp474_forecast2.txt                      weatherapp474.txt
37012: 999_word_frequency.txt                               999_word_frequency2.txt
37013: 999_synedit_codecompletion.txt                       999_RotImg.pas
37014: 1000_earth_rotation.txt                              1000_GraphicsMathLibrary.PAS
37015: 1000_coffeemax.bmp                                   1000_earth_rotation3d.txt
37016: 1000_3DLabScreen.txt                                 1000_GraphicsMathLibrary_Pantograph.PAS
37017: 961_URLDecoding_indyresponse_WMI_Info.txt            1000_graph2.pas
37018: 1000_GenomeLibrary.pas                               1000_Coordinates222.txt
37019: 1000_chr01_GEN_sequence.txt                          1000_Coordinates_Gen.ods
37020: 1000_JVCL_Chart_Demo.PAS                             1000_JVCL_Chart_Demo2.PAS
37021: 1000_GraphicsMathLibrary_PantographF.PAS             1000_JVCL_Chart_Demo2Form.PAS
37022: 1001_entropy_test.txt                                1001_entropy_test2.txt
37023: 1001_entropy_test21.txt                              1002_protein_translation.txt
37024: 1002_protein_translation2.txt                        1000_sphere_rotation3d.txt
37025: 1000_sphere_rotation3direct.txt                      1000_Sphere_big.png
37026: 1000_sphere_sourceforge2.png                         1000_sphere_sourceforge.png
37027: 1000_under_overfit.png                               1000_Sphere_small.png
37028: 1000_Sphere_4poles.png                               1000_sphere_sourceforge.txt
37029: 1000_sphere_sourceforge2.txt                         1000_Nap.bmp
37030: 1003_molecules.txt                                   1003_molecules1_2.txt
37031: 1003_molecules1_3.txt                                1003_molecules1_4.txt
37032: 1003bellsound.wav                                    1003doorbell.wav
37033: 1003templebell.wav                                   1003_sound_molecules.png
37034: 1003_molecules1_5.txt                                1004_wordwheel.txt
37035: 1005_drawsphere_ascii.txt                            1006_write_PPM.txt
37036: 1006_write_PPM_2.txt                                 1006_write_PPM_2read.txt
37037: 1006_renderspherefull2.png                           1006_writeSphere_PPM_2readBMP.txt
```

37038: 1006_writeSphere_PPM_2readBMPfix.txt
37039: 1006spheremaxsmallout2.png
37040: 1006_bitbang_spheremaxsmallout2w.png
37041: 1006_bitbang_spheremaxsmallout2y.png
37042: 1006_manmachine.ppm
37043: 1009_with_multiple_ranges.txt
37044: 1010_ScreenPixel2.txt
37045: 1010_ScreenPixel2.txt
37046: 1010_U_GraphicsEffects2.pas
37047: 1010_geometer_intersection2.png
37048: 1010_circle_point.png
37049: 1010_U_Geometry4Forms.psb
37050: 1010_U_Geometry4Forms.htm
37051: 1010_U_ClockAngles2.pas
37052: 1011_cInternetUtils.pas
37053: 1012_flcTimers.pas
37054: 1012_flcBits32.pas
37055: 1013_saint_source23tests.txt
37056: 210_public_private_cryptosystem6_ibz_tuned.txt
37057: 1014_jcl_stringlist2.txt
37058: 1015_Scripting_Dictionary.txt
37059: 1016_newsfeed_sentiment.py
37060: 1016_sentiment_api2_bbc_newsfeed.txt
37061: 1017_XmlDocRssParser.pas
37062: 1017_bbcnews2.xml
37063: 1017_RSSexportUnit2Webform.pas
37064: 1018_Twelve_statements.txt
37065: 1018_Twelve_statements2.txt
37066: 1019_math_source_cap5.pas
37067: 1019_math_learn_tool.png
37068: 1020_FASTA_format.txt
37069: 1020_Power2FirstDigits.psb
37070: 1020_TopRankPerGroup.txt
37071: 1020_CustomSort_Stringlist_Department.txt
37072: 1021_uCEFMiscFunctions_mX4.pas
37073: 1022_U_SciGraph13_mX47.pas
37074: 1023_FANN_NeuralNetworkLogTrainer.txt
37075: 1024_floydalgo.txt
37076: 1025_Deranged_anagrams2.txt
37077: 1026_json_automation_coviddata.txt
37078: 1026_json_automation_coviddata2.txt
37079: 1026_worldwide5zoom.png
37080: 1026_worldwide4.png
37081: 1026_worldwide3.png
37082: 1026_worldwide.png
37083: 1026_swiss_weeklypattern.png
37084: 1026_swiss_firstwave.png
37085: 1026_jsondos_forecast.png
37086: 1026_json_automation_refactor.txt
37087: 1026_worldwide9_8confirmed.png
37088: 1017_RSSDatacampSSLStreamSentiment.pas
37089: 972_json_tester31.txt
37090: 1026_uJson_testeroauth_print.pas
37091: 1027_fpvectorial.jpg
37092: 1028_agg_math.pas
37093: 1028_JcfUnicodeFiles.pas
37094: 1028_filestream4test.pas
37095: 1028_filestream4test2.pas
37096: 1030_filepilot2.psb
37097: 1030_StrUtil.pas
37098: 1026_SVC_BitVision_MLPrimer.py
37099: 1031_Bitcoinaddress_validation.txt
37100: 1032_population_count.txt
37101: 1033_Corpus_from_Dir.py
37102: 1034_tamath.pas
37103: 1034_tamath_safe.pas
37104: 1035_covidapp.txt
37105: 1035_covidapp3.htm
37106: 652_graph3Dspiral2.pas
37107: 1037_bufferoverflow_tester.txt
37108: 1035_covidapp31.txt
37109: 1039_pcatest.pas
37110: 1039_pcatest3.pas
37111: 1039_roman.pas
37112: 1016_newsfeed_sentiment_integrate2.py
37113: 1016_newsfeed_sentiment_integrate2.htm
37114: 1041_signproof_tester.txt
37115: 1042_superpermutation_minim.txt
37116: 1034_weather_report_py.png
37117: weatherbox.txt
37118: 1045_XMLTreeview2.png
37119: xmldemo.txt
37120: 1045_semaphorgrids.pas
37121: weatherapp475.txt
37122: 1046_openweather_integrate.txt
37123: 1046_openweather_P4D_2.png
37124: 1046_JclTimeZones.pas
37125: 1047_microtonal_tomath_mX4.txt
37126: 1047_microtonal_tomath_mX4_2.txt

1006_writeSphere_PPM_2readBMPfix2.txt
1006_writeSphere_PPM_2readBMPfix2frm.txt
1006_bitbang_spheremaxsmallout2r.png
1007_disabled_bitmap.txt
1008_Knuth_shuffle.txt
1010_ScreenPixel.txt
1010_JPGClock.txt
1010_ProcsVecteurs.pas
1010_geometer_tangent.png
1010_circle_tangent.png
1010_geometer_polygon.png
1010_U_Geometry4Forms.pas
1010_U_ClockAngles2.htm
1010_U_DecToFract2.pas
1011_flcStatistics_mX4.pas
1012_flcCharSet.pas
1010_U_DecToFract21.pas
unittests4.txt
1014_jcl_stringlist.txt
1014_jcl_stringlist21.txt
1015_Scripting_Dictionary_RegEx.txt
1016_newsfeed_sentiment.htm
1016_sentiment_api2_bbc_newsfeed2.txt
1017_bbcnews.xml
1017_XmlDocRssParser_BBCNewsDemo.pas
1017_RSSWexportUnit2formSSLStream.pas
1018_ludic_numbers.txt
1019_math_source_cap5train.pas
1018_Twelve_statements21.txt
1020_perlinNoise.txt
1020_FASTA_format2.txt
1020_Power2FirstDigits.txt
1020_ReST_Countries_API.txt
1020_CustomSort_Stringlist.txt
1021_uCEFMiscFunctions_mX47.pas
1022_scigraphplot.png
1023_FANN_Alllogic2.png
1025_Deranged_anagrams.txt
1026_SVC_BitVision.py
1026_covidtimeserie.txt
1026_json_automation_coviddata21.txt
1026_worldwide5.png
1026_json_automation_coviddata22.txt
1026_worldwide2.png
1026_US_firstyear.png
1026_germany_weeklypattern.png
1026_jsondos_forecast2.png
1026_json_automation_coviddata22teechart.txt
972_json_tester3.txt
1026_json_automation_refactor2.txt
1017_RSSDatacampSSLStreamSentiment2.pas
972_json_tester32.txt
1027_fpvutils.pas
1028_STRandomSystem_tester32.txt
1028_JcfMiscFunctions.pas
1028_JcfMiscFunctions2.pas
1028_filestream5test.pas
1030_filepilot2.txt
1030_filepilot21.txt
1030_StrUtil2.pas
1026_SVC_BitVision_MLPrimer2.py
1031_Bitcoinaddress_validation2.txt
1032_population_count.psb
1032_pascalcoin_validation.txt
1035_asserts_safe.pas
m85_covid6_worldwide_192.png
1035_covidapp3.txt
652_graph3Dspiral.pas
1036_p4d_routines.txt
1038_FileReplaceString.txt
covidapp31.psb
1039_pcatest2.pas
1039_match.pas
1036_p4d_routines2.txt
1016_newsfeed_sentiment_integrate2.txt
1040_htmlhelper.pas
1042_math_mX4.txt
1043_weatherconsole_mX4.txt
1034_Bern_0pq_single.png
1044_SimpMath.pas
1045_testxmlreaderunitmx47.pas
1045_testxmlreaderunitmx47.psb
1045_testxmlreaderunitmx475.pas
1045_testxmlreaderunitmx47520.pas
openweather.txt
1046_openweather_integrate2.txt
763_google_rest_api_json2.txt
1048_uCommonFunctions.pas
100_weather_australia_secure21.txt

```
37127: 1049_InetUtils_Callback.TXT                          336_digiclock3resource.txt
37128: 1050_saint_Pythonengine.txt                          1050_saint_Pythonengine2.txt
37129: 1050_pydemo2.txt                                     1050_pydemo21.txt
37130: 974_64bit_Routines2.txt                              1050_pydemo_51.txt
37131: 1051_semaphorgrid.txt                                778_advapi32_dll_SHA512_plus.txt
37132: 778_advapi32_dll_SHA512_plusRefactor.txt             1052_delphi_arduino21.png
37133: 1052_delphi_arduino2.txt                             1052_delphi_arduino.txt
37134: 1052_delphi_arduino2.png                             1052_OoMiscpas2.txt
37135: 1052_delphi_arduino3.txt                             1052_delphi_arduino31_2.png
37136: 1052_delphi_arduino31.txt                            1052_delphi_arduino31.png
37137: 1052_delphi_arduino32.ini                            1052_delphi_arduino32.txt
37138: 1052_bodmas.txt                                      1050pytest21_5powershttpserve.txt
37139: 1050_pydemo_5_powerliners.txt                        1050_pydemo_5_powerliners23.txt
37140: 1052_OoMiscpas2PyHTTPSServer2LAB.txt                 1052_OoMiscpas2output.txt
37141: 1052_pyhttpserver_integrate22.py                     1050_pydemo_10_pythontips.txt
37142: 1053_U_TimeZoneDemo2.pas                             1054_U_SoundGen_mcisendstring31.pas
37143: 1054_U_SoundGen31.pas                                1055_WaveStorage.pas
37144: 1054_U_SoundGen_mcisendstring32.pas                  1055_U_GridWordHighlight.pas
37145: 1055_U_IntListtest.pas                               1055_tintlistdemo2.png
37146: 1055_tintlistdemo.png                                1055_U_IntListtest.psb
37147: 1052_pydemo_pythonversion.txt                        1052_pydemo_pythonversion2.txt
37148: 1056_neuralvolume_routines.pas                       1019_math_source_cap5_pulse.pas
37149: 1057_pydemo_7_pyIO.txt                               1057_pydemo_7_pyIO.psb
37150: 1058_Arduino.pas                                     1060_pypas_demo10.txt
37151: 1060_pypas_demo10.psb                                1060_pypas_demo10.htm
37152: 1059_pypas_demo11.psb                                1059_pypas_demo11.htm
37153: 1059_pypas_demo11.txt                                1059_pydemo11tester.png
37154: 1060_arduinoUnit1.pas                                1061neuraldatasets_pretester.pas
37155: 1062_utils.py                                        1062_3DCellularUnit1.pas
37156: 1061_gameoflifeUNIT1.PAS                             1062_intervallunit.txt
37157: 1062_intervallunit2.txt                              1062_neuralnet_vectors.txt
37158: 87_1062_neuraltrainer.png                            1063_vector_lists.txt
37159: 1062_neuralnet_vectors2.txt                          1064_vectorclassroutines.txt
37160: 1064_httpd24.pas                                     httpd24.pas
37161: 1065_neuralnetwork_CAI.txt                           1065_neuralnetwork_CAI_2.txt
37162: 1066_KMeans_tester.txt                               1066_neuralvolume_tester.txt
37163: 1067_py_planet_moon.txt                              1052_pydemo_pythonversion31.txt
37164: 1068_vector_matrix_tester.txt                        1068_vector_matrix_tester2.txt
37165: 1068_floattostr_tester.txt                           1063_vector_lists3.txt
37166: 1063_array_lists3.txt                                1070_U_CardProbability2.pas
37167: 1070_U_PerfectDeck.pas                               1070_tshirt_prime.txt
37168: 1070_tprime_tshirt.png                               1069_pydemo30_performance.txt
37169: 1070_U_Keno2mX4.pas                                  1065_neuralnetwork_CAI_2_data.txt
37170: 1065__CAI_2_SimpleImageClassifier.txt                1066_KMeans_tester2.txt
37171: 1065__CAI_2_SimpleImageClassifier2.txt               1070_U_Keno2mX4Form.pas
37172: 1071_Newadds_V47590_Modules.pas                      1070_U_Keno2mX4_Forms2.pas
37173: 1070_tshirt_prime2_tutor88.txt                       1070_U_PrimesFromDigits.pas
37174: 1071_Newadds_V47590_Modules.pas                      1071_Newadds_V47590_Modules2.pas
37175: 1065__CAI_2_SiImageClassifier21_Tutor_89_test3.txt   1072_captcha.txt
37176: 1072_captcha2.txt                                    weatherapp476.txt
37177: 1073__CAI_3_LearnerClassifier22_Tutor_89_2.txt       1074__CAI_3_visualautoencoder.txt
37178: 1074__CAI_3_visualautoencoder2.txt                   1071_Newadds_V47610_Modules3.pas
37179: 1074_U_Systools_mX4Similarity.pas                    1074_U_Systools_mX4Similarity2.pas
37180: 1075_primes30_performance5.txt                       1065__CAI_2_SimpleImageClassifier21_EKON_blogstarter.txt
37181: 1076_CAI_Ugradientascent.pas                         1076_CAI_UVisualGAN2.pas
37182: 1076_CAI_UVisualGAN2tester_MNIST.pas                 1076_CAI_UVisualGAN2tester_MNIST2.pas
37183: 1077_U_Catapult4.pas                                 1077_U_Catapult41.pas
37184: 1078_create_DB.pas                                   1077_U_Catapult41forms.pas
37185: 1077_U_Catapult42forms.pas                           1077_U_Catapult43forms.pas
37186: 1077_U_metronome2.pas                                1078_bell_numbers.pas
37187: 1079_Fregan.pas                                      1079_Test.txt
37188: 1077_U_Catapult43forms2.pas                          1077_U_metronome21.pas
37189: 1080_trend.pas                                       1080_hist.txt
37190: 1080_hist.exe                                        1080_histdatgui.png
37191: 1080_tprob.pas                                       1080_fmTraceRouteMainU.pas
37192: 1080_VCLScanner2021.txt                              1080_VCLScanner2021.uc
37193: 1080_syndat_concept.png                              1081_syndat_pydemo32_2.txt
37194: 1081_syndat_pydemo32_2.psb                           781u_LatLonDistanceTestmX21.pas
37195: 1082_Idiomatically.txt                               1082_ToolsUtils.pas
37196: 1082_ToolsUtils2.pas                                 1071_Newadds_V47610_Modules42.pas
37197: 1083_Deltics_TCommandline.pas                        1084_SeAES256_explain.pas
37198: 1084_DelticsStrUtils.pas                             602_moonbug25.txt
37199: 1085_kmemo_dictionary.pas                            1085_kmemo_dictionary2taunumber.pas
37200: 1086_bigdecimals2.pas                                1086_bigdecimals2python.pas
37201: 1086_bigdecimals2python2.pas                         1087_HQ9.pas
37202: 1088__CAI_3_Hypotenuse.txt                           1088__CAI_3_DenseNetBCL40.txt
37203: 1088__CAI_3_DenseNetBCL40_2.txt                      1050_pydemo_5_powerliners_def_2.txt
37204: 1089_RSA_Example_crypto.htm                          1089_RSA_Example_crypto.txt
37205: 1090_simpleneuralwebserver.txt                       1089_RSA_Example_crypto_Herdt.txt
37206: 1090_simple_neuralwebserver2.txt                     1090_uarduinoformworkspace.pas
37207: 1090_uformlamwinosettingspaths.pas                   1090_Fractran_test.txt
37208: 1090_Fractran_test_Python.txt                        1090_U_Splines.pas
37209: 1090_U_Numbrix2.pas                                  1090_numbrix_GUI.png
37210: 1090_Numbrix2.txt                                    1090_U_Numbrix21.pas
37211: 1090_numbrix_GUI_24.png                              1090_Numbrix2data.txt
37212: 1091_U_SpringMass21.pas                              1091_springmassdemo.png
37213: 1091_U_SpringMass2.pas                               1092_Rollercoaster_Simulator.pas
37214: 1092_Rollercoaster_Simulator21.pas                   1092_Rollercoaster_Simulator21.htm
37215: 1093_XMLUtils_Tutor92tester.txt                      1094_pydemo13tutor_PAS2JS.txt
```

```
37216: 1095_faker_tutor_PAS2JS.txt                          1096_MainFormSVG_2.pas
37217: 1096_VCL4Python.pas                                  1096_P4D_VCL4Python.pas
37218: 1096_MainFormSVG_2test4Python.pas                    1093_CAI_IdentityShortcutConnection2.pas
37219: 1096_pychats_demo4VCL4Python.png                     1096_standalonepyvclexe.png
37220: 1096_MainFormSVG_2test4Python_VCL4Python.pas         1096_MainFormSVG_2test4Python_VCL4Python2.pas
37221: 1100_901_turtle_purple_circle_EKON23.txt             1100_VectorProduct.pas
37222: 1100_U_Compass.pas                                   1100_U_Compass2.pas
37223: 1097_Pierpont_primes.pas                             1097_Pierpont_primes2.pas
37224: 1098_image_noise.pas                                 1098_jaro_similarity.txt
37225: 1098_image_noise2xmlnodes.pas                        1099_rest_localdb.pas
37226: 1001_ESP32_Loop.TXT                                  1069_pydemo30_matrixperformance4.txt
37227: 1102_matrixmultiplication4.txt                       781u_LatLonDistanceTestmX22.pas
37228: 1102_matrixmultiplication41.txt                      1101_PJConsole.txt
37229: 1101_PJConsole1.txt                                  1103_DudsCommonUtils.pas
37230: 1104_TNeuronClass.txt                                1104_TNeuronClass2.txt
37231: 281_picturepuzzle31res.txt                           1101_PJConsole12.txt
37232: 1101_PJConsoleApp_Echoer.png                         Echoer.exe
37233: 1101_PJConsole12Json.txt                             1105_HTML_Builder.txt
37234: 1105_HTML_Builder2DB.txt                             1101_PJConsoleApp_Echoer2.png
37235: 1101_PJConsole1ex2.txt                               1101_PJConsole1_demo9.txt
37236: 1105_HTML_Builder2Modules.txt                        1106_OnlineRandomizer.pas
37237: 1107_Iris_Console.pas                                1107_bezdekIris.fann.data.txt
37238: 1107_Iris_Console2.pas                               1106_OnlineRandomizer_Random_Main.pas
37239: 752_U_AstroDemo_mX4_41.pas                           752_astronomy_moon23.png
37240: 1108_CAI_Geocoding.pas                               1108_geocoding_api.png
37241: 1109_winapidownload.pas                              1110_JSONObject.txt
37242: 1111_lifeappearance_Unit1.pas                        1111_uvisualcifar10animalmachine.pas
37243: 1111_UVPlan1test_QR_Code.pas                         1111_URSOM_Network.pas
37244: 1071_Newadds_V47610_Modules47.pas                    100U_CoinSearch.pas
37245: 1012_UTicTacToeCount.pas                             1012_delphi-utils.pas
37246: 1108_CAI_GeocodingTWininetAPI.pas                    1108_CAI_GeocodingTWininetAPI_only.pas
37247: 1112_lastUtils.pas                                   1112_lastUtils2.pas
37248: 100U_CoinSearch2.pas                                 1111_machineanimal.png
37249: 1111_animalmachine_pictures12.pas                    1011_U_CoinSearc_form.pas
37250: 1011_graphnodes_solution.png                         1011_U_CoinSearch_form.pas
37251: 1011_graphnodes_solution1.png                        1112_U_Wordgrid.pas
37252: 1112_U_Wordgrid.htm                                  1112_wordgrid.png
37253: 1112_wordgrid1.png                                   1112_full.dic
37254: 1113_composition_tree.pas                            1113_composition_tree1.pas
37255: 1071_Newadds_V47610_Modules47tester.pas              ubuntu16_session_maxbox4_cloud2022_full2.png
37256: 1114_polynomial_synthetic_division.txt               1115_Wireworld.txt
37257: 1115_unitKWKalender.pas                              1115_scriptDatenbankdiagnose.pas
37258: 1088__CAI_3_supersimplecorrelation.txt               1088__CAI_3_supersimplehyperbolictangent.txt
37259: 1115_gecodingapi_tee_helvetia_json.txt               1116_U_ChallengingmathTeasers.pas
37260: 1117_sunpos.png                                      1117_sunpos_bern.png
37261: 1117_U_SolarPos2Bern.pas                             1117_U_SolarPos2.pas
37262: 1117_sunpos_bern2.png                                1117_U_SolarPos2Bern2.pas
37263: 1118_Cumulative_standard_deviations.pas              1118_Cumulative_standard_deviations.htm
37264: 1108_CAI_GeocodingTWininetAPI_Maponly.pas            1117_StAstro.pas
37265: 1119_FloatToBinTests.pas                             1120_Department_numbers.pas
37266: 93_1108_CAI_GeocodingTWininetAPI_Maponly.pas.png     1120_DammAlgorithm.pas
37267: 1120_DammAlgorithm12.pas                             sentiment3.txt, sentiment4.txt
37268: 1121_sentiment_api3_bbc_newsfeed4rec.txt             1121_sentiment_api3_bbc_newsfeed4.txt
37269: 1121_sentiment_api3_bbc_newsfeed4rec2.txt            1122umain.pas
37270: 1121_sentiment_api3_bbc_newsfeed4rec21.rtf           1121_sentiment_api3_bbc_newsfeed4rec21.txt
37271: 1122_langtranslator.txt                              1122_langtranslator1.txt
37272: 1123_IntegerArraySort.pas                            1123_MATHBOX.PAS
37273: 1123_CODEWIZU.PAS                                    1123_CODEWIZU_2.PAS
37274: 1124_CODEReflection.PAS                              1125_Forward_Difference.pas
37275: 1124_CODEReflection1.PAS                             1125_unitGlobal.pas
37276: 1125_unitGlobal1.pas                                 1125_unitGlobal12.pas
37277: 1125_unitGlobal13.pas                                1100_U_Compass1_deltics_sysutils.pas
37278: 1126_Projectile_Physics.pas                          1126_Projectile_Physics2.pas
37279: 1126_Projectile_Physics21.pas                        1126_projectile2.png
37280: 1126_fetchapi1.png                                   1126_fetch_api1.txt
37281: 727_mondrian_next.png                                727_mondrian_next21.png
37282: 727_Canvas_Artist_mondrian_gen.TXT                   727_Canvas_Artist_mondrian_gen2.TXT
37283: 1127_RegistryWatchDog.pas                            727_Canvas_Artist_mondrian_gen21.TXT
37284: 1128_fpvutils.pas                                    1128_mean_angle1.txt
37285: 1129_urlscan_api_json.txt                            1129_urlscan_api_json1.txt
37286: 1130_Scanner_gen21.TXT                               1130_mondrian_QRCode_P.png
37287: 1130_mondrian_QRCode_P2.png                          1130_Scanner_QR_21.TXT
37288: 1130_urlscan_api_json1_geoip.txt                     1130_urlscan_api_json1_geoip1.txt
37289: 1131_rosetta_numerical_integration.txt               1132_rosetta_Farey_sequence.txt
37290: 1133_klib_utilities.txt                              1133_klib_utilities2.txt
37291: 1134_syn_dat_pop.txt                                 1134_Cifar10Resize.pas
37292: 1134_Cifar10Resize_win_lessthread.pas                1135_Cifar10SeparableConvolution.pas
37293: 1135_Cifar10SeparableConvolution_50.pas              1136_hamming_numbers1.txt
37294: 1136_Chowla_numbers1.txt                             1135_uvisualcifar10test_mX4_1.psb
37295: 1135_visualCNN5.png                                  1135_uvisualcifar10test_mX4_1.pas
37296: SimpleSeparableImageClassifier124_50_21.nn           1137_gauss_legendre.pas
37297: 1138_ocr_tesseract.txt                               1138_testmap2.png
37298: 1138_bmp.bmp                                         1139_pentagram.txt
37299: 1140_price_fraction.txt                              1140_Statistics.pas
37300: 1140_Statistics1.pas                                 1140_USolarSystem.pas
37301: 1141_adafruit_tempsensor21.txt                       1140_PlanetData.pas
37302: 1140_Om_AstronomicalAlgorithms.pas                   1140_doubleVector3D.pas
37303: 1140_quaternion_3D.pas                               1141_fast_sin.pas
37304: 1142_anti_primes.pas                                 1139_pentagram1.txt
```

```
37305: 1142_list_virtualconstructors_test.pas          1142_setbit_comparison.pas
37306: 1142_list_collections_pydemo42.txt              1140_doubleVector3D_collection.pas
37307: 1143_KLibVC_Redist.pas                          1143_KLibVC_Redist2.pas
37308: 1071_Newadds_V47610_Modules48.pas               1071_Newadds_V47610_Modules49.pas
37309: 1145_271_closures_study_op_sys_tutor97.txt      1142_list_collections_pydemo42.txt
37310: 1144_AzuliaUtils_mX4.pas                        1145_271_closures_study_op_sys_tutor97_1.txt
37311: 1146_filesize_distribution.txt                  1147_practicalnumbers_2.txt
37312: 1148_julia_set.txt                              1148_julia_set.png
37313: 1148_julia-set.bmp                              1149_sum_to_100.txt
37314: 1150_DigitalRoot.txt                            1150_pydemo42httpheader.txt
37315: 1150_pydemo42httpheader1.txt                    1150_Partition_function_P.txt
37316: 1150_Partition_function_P1.txt                  1151_rapid_api_proto.txt
37317: 1151_sentiment_api4simple.txt                   1151_sentiment_api4simple1.txt
37318: 1151_rapid_api_proto1.txt                       1150_pydemo42httpheader2openssl.txt
37319: 1152_dos_output_callback.txt                    629_Word_FORMULA2super.PAS
37320: 1151_rapid_api_proto_googleapi12.txt            1152_shell_composer.txt
37321: 1153_space_invaders3.txt                        170_4gewinnt_main21.txt
37322: 170_4gewinnt_main21.htm                         170_4gewinnt_main31.txt
37323: 1142_list_collections_pydemo42_1.txt            1154_energy_api_agsi.txt
37324: 1154_energy_api_agsi_plot.txt                   1154_energy_api_agsi_plot1.txt
37325: 1154_energy_api_agsi_plot12.txt                 1154_energy_api_agsi_plot13.txt
37326: 1155_census_data.txt                            1155_census_data_graph.txt
37327: 1155_Utils47620.pas                             1155_dynmatrix47620.pas
37328: 1156_SearchUtils.pas                            1157_wininet_internetreadfile_dll.pas
37329: 1058_teetest_screen21map.TXT                    1158_resourceloader2.pas
37330:
37331: 685 uPSI_ALQuickSortList;                      //alcinoe for list of double fast
37332: with TALDoubleList.create do begin  //singlelist
37333:   with CL.AddClassN(CL.FindClass('TALBaseQuickSortList'),'TALDoubleList') do begin
37334:     Function IndexOf( Item : Double) : Integer');
37335:     Function IndexOfObject( AObject : TObject) : Integer');
37336:     Function Add( const Item : Double) : Integer');
37337:     Function AddObject( const Item : Double; AObject : TObject) : Integer');
37338:     Function Find( item : Double; var Index : Integer) : Boolean');
37339:     Procedure Insert( Index : Integer; const item : Double);
37340:     Procedure InsertObject( Index : Integer; const item : Double; AObject : TObject);
37341:     RegisterProperty('Items', 'Double Integer', iptrw);
37342:     SetDefaultPropery('Items');
37343:     RegisterProperty('Objects', 'TObject Integer', iptrw);
37344:   end;
37345:
37346: https://sourceforge.net/projects/maxbox/files/Examples/13_General/
37347: http://www.softwareschule.ch/examples/signtoolbatch.txt
37348: http://www.softwareschule.ch/examples/signtoolbatch.htm
37349:
37350: https://cdn.top4download.com/img/award_120x60_pick.gif
37351: http://www.kleiner.com/boxart.htm
37352: http://max.kleiner.com/boxart.htm
37353: http://www.kleiner.ch/kleiner/boxart7.htm
37354:
37355: All 32 Articles of maXbox Pascal in Blaise Pascal Magazine!
37356: ------------------------------------------------------------
37357: Using Examples and Sources from Magazine CatalogX
37358: Art of Coding V.31/32  5/6 2013
37359: DLL for All V.33
37360: QR Code Coding V.34
37361: GEO Maps Nav V.35
37362: 3D Printing Lab V.36
37363: RegEx Report REX V.37/38
37364: Func Testing V.40
37365: Arduino Coding V.45/46
37366: Time is on my side V.48
37367: Compute Big Numbers V.52
37368: Code Metrics V.57
37369: Work with WineHQ V.60
37370: Work with Win WMI V.63
37371: Work with Win WMI II V.64
37372: Work MS CryptoAPI V.65
37373: The Perceptron! V.72
37374: No GUI Shell V.80
37375: Machine L. with CAI V.87/88
37376: Portable Pixmap PPM V.90
37377: Fundamentals Lib V.91
37378: RSS Feed of BBC News V.92
37379: RSS Feed Sentiment V.93
37380: JSON4Pascal V.94/95
37381: Python4Delphi 96
37382: Python4Delphi II 97
37383: Python4maXbox III 98
37384: Image Classification for Lazarus 99/100 !
37385: Faker SynDat 101
37386: Catapult, Coaster Simulator, VCL4Python 102
37387: Geocoding II 103/104
37388: CIFAR 10 Image Classifier 105
37389: Energy_API_agsi_Plot 106
37390:
37391: Proposal for Boolean Logic Abreviation Symbol:
37392: ---------------------------------------------------
37393:
```

```
37394:   01 FALSE    //Contradiction
37395:
37396:   02 AND      //Conjunction x*y
37397:
37398:   03 INHIB    //Inhibition x*^y
37399:
37400:   04 PRAEP    //Praependence x
37401:
37402:   05 PRAE     //Praesection ^x*y
37403:
37404:   06 POST     //Postpendence y
37405:
37406:   07 XOR      //Exclusive OR x*^y+^x*y
37407:
37408:   08 OR       //Disjunction OR = x+y
37409:
37410:   09 NOR      //Rejection
37411:
37412:   10 AEQ      //Aequivalence x<-->y, ^x*^y+x*y
37413:
37414:   11 NEGY     //YNegation ^y
37415:
37416:   12 IMPY     //YImplication y-->x; x+^y
37417:
37418:   13 NEGX     //Xnegation ^x
37419:
37420:   14 IMPX     //XImplication x-->y; ^x+y
37421:
37422:   15 NAND     //Exclusion
37423:
37424:   16 TRUE     //TRUE Tautologic
37425: ----------------------------------------------------------
37426:
37427: http://sourceforge.net/projects/maxbox/files/Examples/
37428: http://sourceforge.net/projects/maxbox/files/Examples/
37429:
37430: Help Online: http://www.softwareschule.ch/maxbox_functions.txt
37431:
37432: WebScript Examples: Get Web Script
37433: http://www.softwareschule.ch/examples/performer.txt;
37434: http://www.softwareschule.ch/examples/turtle.txt;
37435: http://www.softwareschule.ch/examples/SQLExport.txt;
37436: http://www.softwareschule.ch/examples/Richter.txt;
37437: http://www.softwareschule.ch/examples/checker.txt;
37438: http://www.softwareschule.ch/examples/demoscript.txt;
37439: http://www.softwareschule.ch/examples/ibzresult.txt;
37440: http://www.softwareschule.ch/examples/performindex.txt
37441: http://www.softwareschule.ch/examples/processlist.txt
37442: http://www.softwareschule.ch/examples/game.txt
37443: http://www.softwareschule.ch/examples/GEOGPS.txt
37444: http://www.softwareschule.ch/examples/turtle2.txt
37445: http://www.softwareschule.ch/examples/turtle3.txt
37446: http://www.softwareschule.ch/examples/asyncterminal.txt
37447: http://www.softwareschule.ch/examples/snowflake.txt
37448: http://www.softwareschule.ch/examples/arduinoled.txt
37449: http://www.softwareschule.ch/examples/moon2.txt
37450: http://www.softwareschule.ch/examples/cockpit.txt
37451: http://www.softwareschule.ch/examples/tartaruga.txt
37452: http://www.softwareschule.ch/examples/surprise.txt
37453: http://www.softwareschule.ch/examples/weatherapp2.txt
37454: http://www.softwareschule.ch/examples/weatherapp3.txt
37455: http://www.softwareschule.ch/examples/bigint.txt
37456: http://www.softwareschule.ch/examples/weatherapp.txt
37457: http://www.softwareschule.ch/examples/weatherapp.txt
37458: http://www.softwareschule.ch/examples/crypto.txt
37459: http://www.softwareschule.ch/examples/stack.htm
37460: http://www.softwareschule.ch/examples/fisher.htm
37461: http://www.softwareschule.ch/examples/arduinotemp.htm
37462: http://www.softwareschule.ch/examples/web.txt
37463: http://www.softwareschule.ch/examples/web3.txt
37464: http://www.softwareschule.ch/examples/web4.txt
37465: http://www.softwareschule.ch/examples/cryptocomplete.txt
37466: http://www.softwareschule.ch/examples/web4arduino.txt
37467: http://www.softwareschule.ch/examples/adafruit.txt
37468: http://www.softwareschule.ch/examples/stack.txt
37469: http://www.softwareschule.ch/examples/stack.htm
37470: http://www.softwareschule.ch/examples/billiard.txt
37471: http://www.softwareschule.ch/examples/systeminfo_tester.htm
37472: http://www.softwareschule.ch/examples/systeminfo.htm
37473: http://www.softwareschule.ch/examples/elevator.txt
37474: http://www.softwareschule.ch/examples/sha256.txt
37475: http://www.softwareschule.ch/examples/robo.txt
37476: http://www.softwareschule.ch/examples/robo2.txt
37477: http://www.softwareschule.ch/examples/font.txt
37478: http://www.softwareschule.ch/examples/spider.txt
37479: http://www.softwareschule.ch/examples/totient.txt
37480: http://www.softwareschule.ch/examples/811_mXpcatest_dmath_datascience.txt
37481: http://www.softwareschule.ch/examples/datascience.txt
37482: http://www.softwareschule.ch/examples/pi.txt
```

37483: http://www.softwareschule.ch/examples/excelexport.txt
37484: http://www.softwareschule.ch/examples/tokenizer.htm
37485: http://www.softwareschule.ch/examples/snake.txt
37486: http://www.softwareschule.ch/examples/asteroid.txt
37487: http://www.softwareschule.ch/examples/weatherapp46.txt
37488: http://www.softwareschule.ch/examples/cryptocomplete.txt
37489: http://www.softwareschule.ch/examples/web4arduino.txt
37490: http://www.softwareschule.ch/examples/wmi.txt
37491: http://www.softwareschule.ch/examples/wmi2.txt
37492: http://www.softwareschule.ch/examples/reverse.htm
37493: http://www.softwareschule.ch/examples/weatherapp4.txt
37494: http://www.softwareschule.ch/examples/tictac.txt
37495: http://www.softwareschule.ch/examples/tictac3.txt
37496: http://www.softwareschule.ch/examples/mapbox2.txt
37497: http://www.softwareschule.ch/examples/neuralnetwork.txt
37498: http://www.softwareschule.ch/examples/datascience.txt
37499: http://www.softwareschule.ch/examples/billiard.txt
37500: http://www.softwareschule.ch/examples/sentiment2.txt
37501: http://www.softwareschule.ch/examples/python.htm
37502: http://www.softwareschule.ch/examples/asteroid.txt
37503: http://www.softwareschule.ch/examples/penny.txt
37504: http://www.softwareschule.ch/examples/snake3.txt
37505: http://www.softwareschule.ch/examples/unittest.txt
37506: http://www.softwareschule.ch/examples/410_titanic_keras_predictor.txt
37507: http://www.softwareschule.ch/examples/410_titanic_keras_predictor.htm
37508: http://www.softwareschule.ch/examples/histogram.txt
37509: http://www.softwareschule.ch/examples/odometer.txt
37510: http://www.softwareschule.ch/examples/chartapi.txt
37511: http://www.softwareschule.ch/examples/geoloc.txt
37512: http://www.softwareschule.ch/examples/xml.txt
37513: http://www.softwareschule.ch/examples/4gewinnt.txt
37514: http://www.softwareschule.ch/examples/cgi.txt
37515: http://www.softwareschule.ch/examples/singlesamplepredict.py
37516: http://www.softwareschule.ch/examples/perform.txt
37517: http://www.softwareschule.ch/examples/statistic.txt
37518: http://www.softwareschule.ch/examples/picbox.txt
37519: http://www.softwareschule.ch/examples/memory2.txt
37520: http://www.softwareschule.ch/examples/getgeomap.txt
37521: http://www.softwareschule.ch/examples/getzip.txt
37522: http://www.softwareschule.ch/examples/getrsa.txt
37523: http://www.softwareschule.ch/examples/getrsa5.txt
37524: http://www.softwareschule.ch/examples/getrsa5.htm
37525: http://www.softwareschule.ch/examples/scramble.txt
37526: http://www.softwareschule.ch/examples/factor2.txt
37527: http://www.softwareschule.ch/examples/bug.txt
37528: http://www.softwareschule.ch/examples/smalldic.txt
37529: http://www.softwareschule.ch/examples/metadefender.txt
37530: http://www.softwareschule.ch/examples/pythonoma.txt
37531: http://www.softwareschule.ch/examples/992_detector21_wget_integrate2.htm
37532: http://www.softwareschule.ch/examples/perceptron.txt
37533: http://www.softwareschule.ch/examples/perceptron2.txt
37534: http://www.softwareschule.ch/examples/perceptron2.htm
37535: http://www.softwareschule.ch/examples/sphere.txt
37536: http://www.softwareschule.ch/examples/entropy.txt
37537: http://www.softwareschule.ch/examples/protein.txt
37538: http://www.softwareschule.ch/examples/wordwheel.txt
37539: http://www.softwareschule.ch/examples/soundcloud.txt
37540: http://www.softwareschule.ch/examples/geometry.txt
37541: http://www.softwareschule.ch/examples/time.txt
37542: http://www.softwareschule.ch/examples/sphere2.txt
37543: http://www.softwareschule.ch/examples/sphere2.htm
37544: http://www.softwareschule.ch/examples/unittests.txt
37545: http://www.softwareschule.ch/examples/newssentiment.txt
37546: http://www.softwareschule.ch/examples/newssentiment.htm
37547: http://www.softwareschule.ch/examples/bbcnews.txt
37548: http://www.softwareschule.ch/examples/mathtool.txt
37549: http://www.softwareschule.ch/examples/newssentiment2.txt
37550: http://www.softwareschule.ch/examples/covid2.txt
37551: http://www.softwareschule.ch/examples/covid3.txt
37552: http://www.softwareschule.ch/examples/covidapp3.txt
37553: http://www.softwareschule.ch/examples/signtoolbatch.txt
37554: http://www.softwareschule.ch/examples/openweather.txt
37555: http://www.softwareschule.ch/examples/weatherbox.txt
37556: http://www.softwareschule.ch/examples/wininet.txt
37557: http://www.softwareschule.ch/examples/digiclock.txt
37558: http://www.softwareschule.ch/examples/pydemo.txt
37559: http://www.softwareschule.ch/examples/pydemo2.txt
37560: http://www.softwareschule.ch/examples/pydemo3.txt
37561: http://www.softwareschule.ch/examples/pydemo13.txt
37562: http://www.softwareschule.ch/examples/pydemo31.txt
37563: http://www.softwareschule.ch/examples/catapult.txt
37564: http://www.softwareschule.ch/examples/pydemo33.txt
37565: http://www.softwareschule.ch/examples/syndat.png
37566: http://www.softwareschule.ch/examples/cryptosystem2.txt
37567: http://www.softwareschule.ch/examples/cryptosystem3.txt
37568: http://www.softwareschule.ch/examples/springdemo.txt
37569: http://www.softwareschule.ch/examples/coaster.txt
37570: http://www.softwareschule.ch/examples/pydemo38.txt
37571: http://www.softwareschule.ch/examples/pydemo40.txt

```
http://www.softwareschule.ch/examples/sunpos.txt
http://www.softwareschule.ch/examples/geocoding.txt
http://www.softwareschule.ch/examples/sentiment3.txt
http://www.softwareschule.ch/examples/sentiment4.txt
http://www.softwareschule.ch/examples/qrcode.txt
http://www.softwareschule.ch/examples/json3.txt
http://www.softwareschule.ch/examples/json4.txt
http://www.softwareschule.ch/examples/shellcomposer.txt
http://www.softwareschule.ch/examples/4gewinnt2.txt
http://www.softwareschule.ch/examples/space2.txt

https://maxbox4.wordpress.com">maxbox4.wordpress.com
https://softwareschule.code.blog">softwareschule.code.blog
https://my6.code.blog">my6.code.blog
https://github.com/maxkleiner/maXbox4/blob/master/newstoday_sentiment.ipynb
http://www.softwareschule.ch/download/exampleedition2016.zip
https://github.com/maxkleiner/fundamentals5
https://thenewstack.io/which-programming-languages-use-the-least-electricity/
https://www.clevercomponents.com/articles/article052/


Delphi Basics Run Time Library listing
******************************************************************************
A
Compiler Directive $A Determines whether data is aligned or packed
Compiler Directive $Align Determines whether data is aligned or packed
Compiler Directive $AppType Determines the application type : GUI or Console
Proc SysUtils Abort Aborts the current processing with a silent exception
Func System Abs Gives the absolute value of a number (-ve sign is removed)
Directive Abstract Defines a class method only implemented in subclasses
Variable System AbstractErrorProc Defines a proc called when an abstract method is called
Func System Addr Gives the address of a variable, Func or procedure
Keyword And Boolean and or bitwise and of two arguments
Type System AnsiChar A character type guaranteed to be 8 bits in size
Func SysUtils AnsiCompareStr Compare two strings for equality
Func SysUtils AnsiCompareText Compare two strings for equality, ignoring case
Func StrUtils AnsiContainsStr Returns true if a string contains a substring
Func StrUtils AnsiEndsStr Returns true if a string ends with a substring
Func StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
Func StrUtils AnsiLeftStr Extracts characters from the left of a string
Func SysUtils AnsiLowerCase Change upper case characters in a string to lower case
Func StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
Func StrUtils AnsiMidStr Returns a substring from the middle characters of a string
Func StrUtils AnsiPos Find the position of one string in another
Func StrUtils AnsiReplaceStr Replaces a part of one string with another
Func StrUtils AnsiReverseString Reverses the sequence of letters in a string
Func StrUtils AnsiRightStr Extracts characters from the right of a string
Func StrUtils AnsiStartsStr Returns true if a string starts with a substring
Type System Ansistr A data type that holds a string of AnsiChars
Func SysUtils AnsiUpperCase Change lower case characters in a string to upper case
Proc System Append Open a text file to allow appending of text to the end
Proc SysUtils AppendStr Concatenate one string onto the end of another
Func Math ArcCos The Arc Cosine of a number, returned in radians
Func Math ArcSin The Arc Sine of a number, returned in radians
Func System ArcTan The Arc Tangent of a number, returned in radians
Keyword Array A data type holding indexable collections of data
Keyword As Used for casting object references
Proc System Assign Assigns a file handle to a binary or text file
Func System Assigned Returns true if a reference is not nil
Proc System AssignFile Assigns a file handle to a binary or text file
Proc Printers AssignPrn Treats the printer as a text file - an easy way of printing text

B
Compiler Directive $B Whether to short cut and and or operations
Compiler Directive $BoolEval Whether to short cut and and or operations
Proc SysUtils Beep Make a beep sound
Keyword Begin Keyword that starts a statement block
Func System BeginThread Begins a separate thread of code execution
Proc System BlockRead Reads a block of data records from an untyped binary file
Proc System BlockWrite Writes a block of data records to an untyped binary file
Type System Boolean Allows just True and False values
Func Classes Bounds Create a TRect value from top left and size values
Proc System Break Forces a jump out of a single loop
Type System Byte An Int type supporting values 0 to 255

C
Type System Cardinal The basic unsigned Int type
Keyword Case A mechanism for acting upon different values of an Ordinal
Func StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
Func SysUtils ChangeFileExt Change the extension part of a file name
Type System Char Variable type holding a single character
Proc System ChDir Change the working drive plus path for a specified drive
Func System Chr Convert an Int into a character
Keyword Class Starts the declaration of a type of object class
Proc System Close Closes an open file
Proc System CloseFile Closes an open file
Variable System CmdLine Holds the execution text used to start the current program
Type System Comp A 64 bit signed Int
Func SysUtils CompareStr Compare two strings to see which is greater than the other
```

```
37661: Func SysUtils CompareText Compare two strings for equality, ignoring case
37662: Func Math CompareValue Compare numeric values with a tolerance
37663: Func System Concat Concatenates one or more strings into one string
37664: Keyword Const Starts the definition of fixed data values
37665: Keyword Constructor Defines the method used to create an object from a class
37666: Proc System Continue Forces a jump to the next iteration of a loop
37667: Func ConvUtils Convert Convert one measurement value to another
37668: Func System Copy Create a copy of part of a string or an array
37669: Func System Cos The Cosine of a number
37670: Func SysUtils CreateDir Create a directory
37671: Type System Currency A floating point type with 4 decimals used for financial values
37672: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
37673: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
37674: Variable SysUtils CurrencyString The currency string used in currency display functions
37675: Func SysUtils CurrToStr Convert a currency value to a string
37676: Func SysUtils CurrToStrF Convert a currency value to a string with formatting
37677:
37678: D
37679: Compiler Directive $D Determines whether application debug information is built
37680: Compiler Directive $DebugInfo Determines whether application debug information is built
37681: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
37682: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
37683: Func SysUtils Date Gives the current date
37684: Variable SysUtils DateSeparator The character used to separate display date fields
37685: Func SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
37686: Func SysUtils DateTimeToStr Converts TDateTime date and time values to a string
37687: Proc SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
37688: Func SysUtils DateToStr Converts a TDateTime date value to a string
37689: Func DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
37690: Func DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
37691: Func DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
37692: Func SysUtils DayOfWeek Gives day of week index for a TDateTime value
37693: Func DateUtils DaysBetween Gives the whole number of days between 2 dates
37694: Func DateUtils DaysInAMonth Gives the number of days in a month
37695: Func DateUtils DaysInAYear Gives the number of days in a year
37696: Func DateUtils DaySpan Gives the fractional number of days between 2 dates
37697: Proc System Dec Decrement an ordinal variable
37698: Variable SysUtils DecimalSeparator The character used to display the decimal point
37699: Proc SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
37700: Proc DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
37701: Proc SysUtils DecodeTime Break a TDateTime value into individual time values
37702: Directive Default Defines default processing for a property
37703: Func Math DegToRad Convert a degrees value to radians
37704: Proc System Delete Delete a section of characters from a string
37705: Func SysUtils DeleteFile Delete a file specified by its file name
37706: Keyword Destructor Defines the method used to destroy an object
37707: Func SysUtils DirectoryExists Returns true if the given directory exists
37708: Func SysUtils DiskFree Gives the number of free bytes on a specified drive
37709: Func SysUtils DiskSize Gives the size in bytes of a specified drive
37710: Proc System Dispose Dispose of storage used by a pointer type variable
37711: Keyword Div Performs Int division, discarding the remainder
37712: Keyword Do Defines the start of some controlled action
37713: Type System Double A floating point type supporting about 15 digits of precision
37714: Keyword DownTo Prefixes an decremental for loop target value
37715: Func StrUtils DupeString Creates a string containing copies of a substring
37716: Directive Dynamic Allows a class method to be overriden in derived classes
37717:
37718: E
37719: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
37720: Compiler Directive $EndIf Terminates conditional code compilation
37721: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
37722: Keyword Else Starts false section of if, case and try statements
37723: Func SysUtils EncodeDate Build a TDateTime value from year, month and day values
37724: Func DateUtils EncodeDateTime Build a TDateTime value from day and time values
37725: Func SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
37726: Keyword End Keyword that terminates statement blocks
37727: Func DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
37728: Func DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
37729: Proc System EndThread Terminates a thread with an exit code
37730: Func System Eof Returns true if a file opened with Reset is at the end
37731: Func System Eoln Returns true if the current text file is pointing at a line end
37732: Proc System Erase Erase a file
37733: Variable System ErrorAddr Sets the error address when an application terminates
37734: Keyword Except Starts the error trapping clause of a Try statement
37735: Proc System Exclude Exclude a value in a set variable
37736: Proc System Exit Exit abruptly from a Func or procedure
37737: Variable System ExitCode Sets the return code when an application terminates
37738: Func System Exp Gives the exponent of a number
37739: Directive System Export Makes a Func or Proc in a DLL externally available
37740: Type System Extended The floating point type with the highest capacity and precision
37741: Func SysUtils ExtractFileDir Extracts the dir part of a full file name
37742: Func SysUtils ExtractFileDrive Extracts the drive part of a full file name
37743: Func SysUtils ExtractFileExt Extracts the extension part of a full file name
37744: Func SysUtils ExtractFileName Extracts the name part of a full file name
37745: Func SysUtils ExtractFilePath Extracts the path part of a full file name
37746:
37747: F
37748: Func StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
37749: Keyword File Defines a typed or untyped file
```

```
37750: Func SysUtils FileAge Get the last modified date/time of a file without opening it
37751: Func SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
37752: Func SysUtils FileExists Returns true if the given file exists
37753: Func SysUtils FileGetAttr Gets the attributes of a file
37754: Variable System FileMode Defines how Reset opens a binary file
37755: Func System FilePos Gives the file position in a binary or text file
37756: Func SysUtils FileSearch Search for a file in one or more directories
37757: Func SysUtils FileSetAttr Sets the attributes of a file
37758: Func SysUtils FileSetDate Set the last modified date and time of a file
37759: Func System FileSize Gives the size in records of an open file
37760: Proc System FillChar Fills out a section of storage with a fill character or byte value
37761: Keyword Finally Starts the unconditional code section of a Try statement
37762: Func SysUtils FindClose Closes a successful FindFirst file search
37763: Func SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
37764: Func SysUtils FindFirst Finds all files matching a file mask and attributes
37765: Func SysUtils FindNext Find the next file after a successful FindFirst
37766: Func SysUtils FloatToStr Convert a floating point value to a string
37767: Func SysUtils FloatToStrF Convert a floating point value to a string with formatting
37768: Proc System Flush Flushes buffered text file data to the file
37769: Keyword For Starts a loop that executes a finite number of times
37770: Func SysUtils ForceDirectories Create a new path of directories
37771: Func SysUtils Format Rich formatting of numbers and text into a string
37772: Func SysUtils FormatCurr Rich formatting of a currency value into a string
37773: Func SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
37774: Func SysUtils FormatFloat Rich formatting of a floating point number into a string
37775: Func System Frac The fractional part of a floating point number
37776: Proc SysUtils FreeAndNil Free memory for an object and set it to nil
37777: Proc System FreeMem Free memory storage used by a variable
37778: Keyword System Func Defines a subroutine that returns a value
37779:
37780: G
37781: Func SysUtils GetCurrentDir Get the current directory (drive plus directory)
37782: Proc System GetDir Get the default directory (drive plus path) for a specified drive
37783: Func System GetLastError Gives the error code of the last failing Windows API call
37784: Proc SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
37785: Func System GetMem Get a specified number of storage bytes
37786: Keyword Goto Forces a jump to a label, regardless of nesting
37787:
37788: H
37789: Compiler Directive $H Treat string types as Ansistr or ShortString
37790: Compiler Directive $Hints Determines whether Delphi shows compilation hints
37791: Proc System Halt Terminates the program with an optional dialog
37792: Func System Hi Returns the hi-order byte of a (2 byte) Int
37793: Func System High Returns the highest value of a type or variable
37794:
37795: I
37796: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
37797: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
37798: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
37799: Compiler Directive $IfOpt Tests for the state of a Compiler directive
37800: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
37801: Compiler Directive $IOChecks When on, an IO operation error throws an exception
37802: Keyword If Starts a conditional expression to determine what to do next
37803: Keyword Implementation Starts the implementation (code) section of a Unit
37804: Keyword In Used to test if a value is a member of a set
37805: Proc System Inc Increment an ordinal variable
37806: Func DateUtils IncDay Increments a TDateTime variable by + or - number of days
37807: Proc System Include Include a value in a set variable
37808: Func DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
37809: Func DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
37810: Func SysUtils IncMonth Increments a TDateTime variable by a number of months
37811: Func DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
37812: Func DateUtils IncYear Increments a TDateTime variable by a number of years
37813: Directive Index Principally defines indexed class data properties
37814: Constant Math Infinity Floating point value of infinite size
37815: Keyword Inherited Used to call the parent class constructor or destructor method
37816: Variable System Input Defines the standard input text file
37817: Func Dialogs InputBox Display a dialog that asks for user text input, with default
37818: Func Dialogs InputQuery Display a dialog that asks for user text input
37819: Proc System Insert Insert a string into another string
37820: Func System Int The Int part of a floating point number as a float
37821: Type System Int64 A 64 bit sized Int - the largest in Delphi
37822: Type System Int The basic Int type
37823: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
37824: Func SysUtils IntToHex Convert an Int into a hexadecimal string
37825: Func SysUtils IntToStr Convert an Int into a string
37826: Func System IOResult Holds the return code of the last I/O operation
37827: Keyword Is Tests whether an object is a certain class or ascendant
37828: Func Math IsInfinite Checks whether a floating point number is infinite
37829: Func SysUtils IsLeapYear Returns true if a given calendar year is a leap year
37830: Func System IsMultiThread Returns true if the code is running multiple threads
37831: Func Math IsNaN Checks to see if a floating point number holds a real number
37832:
37833: L
37834: Compiler Directive $L Determines what application debug information is built
37835: Compiler Directive $LocalSymbols Determines what application debug information is built
37836: Compiler Directive $LongStrings Treat string types as Ansistr or ShortString
37837: Func SysUtils LastDelimiter Find the last position of selected characters in a string
37838: Func System Length Return the number of elements in an array or string
```

```
37839: Func System Ln Gives the natural logarithm of a number
37840: Func System Lo Returns the low-order byte of a (2 byte) Int
37841: Func Math Log10 Gives the log to base 10 of a number
37842: Variable SysUtils LongDateFormat Long version of the date to string format
37843: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
37844: Type System LongInt An Int whose size is guaranteed to be 32 bits
37845: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
37846: Variable SysUtils LongTimeFormat Long version of the time to string format
37847: Type System LongWord A 32 bit unsigned Int
37848: Func System Low Returns the lowest value of a type or variable
37849: Func SysUtils LowerCase Change upper case characters in a string to lower case
37850:
37851: M
37852: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
37853: Func Math Max Gives the maximum of two Int values
37854: Constant System MaxInt The maximum value an Int can have
37855: Constant System MaxLongInt The maximum value an LongInt can have
37856: Func Math Mean Gives the average for a set of numbers
37857: Func Dialogs MessageDlg Displays a message, symbol, and selectable buttons
37858: Func Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
37859: Func Math Min Gives the minimum of two Int values
37860: Constant SysUtils MinsPerDay Gives the number of minutes in a day
37861: Proc System MkDir Make a directory
37862: Keyword Mod Performs Int division, returning the remainder
37863: Constant SysUtils MonthDays Gives the number of days in a month
37864: Func DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
37865: Proc System Move Copy bytes of data from a source to a destination
37866:
37867: N
37868: Constant Math NaN Not a real number
37869: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
37870: Proc System New Create a new pointer type variable
37871: Constant System Nil A pointer value that is defined as undetermined
37872: Keyword Not Boolean Not or bitwise not of one arguments
37873: Func SysUtils Now Gives the current date and time
37874: Variable Variants Null A variable that has no value
37875:
37876: O
37877: Compiler Directive $O Determines whether Delphi optimises code when compiling
37878: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
37879: Compiler Directive $OverFlowChecks Determines whether Delphi checks Int and enum bounds
37880: Keyword System Object Allows a subroutine data type to refer to an object method
37881: Func System Odd Tests whether an Int has an odd value
37882: Keyword Of Linking keyword used in many places
37883: Keyword On Defines exception handling in a Try Except clause
37884: Keyword Or Boolean or or bitwise or of two arguments
37885: Func System Ord Provides the Ordinal value of an Int, character or enum
37886: Directive Out Identifies a routine parameter for output only
37887: Variable System Output Defines the standard output text file
37888: Directive Overload Allows 2 or more routines to have the same name
37889: Directive Override Defines a method that replaces a virtual parent class method
37890:
37891: P
37892: Keyword Packed Compacts complex data types into minimal storage
37893: Type System PAnsiChar A pointer to an AnsiChar value
37894: Type System PAnsistr Pointer to an Ansistr value
37895: Func System ParamCount Gives the number of parameters passed to the current program
37896: Func System ParamStr Returns one of the parameters used to run the current program
37897: Type System PChar A pointer to an Char value
37898: Type System PCurrency Pointer to a Currency value
37899: Type System PDateTime Pointer to a TDateTime value
37900: Type System PExtended Pointer to a Extended floating point value
37901: Func System Pi The mathematical constant
37902: Type System PInt64 Pointer to an Int64 value
37903: Func Classes Point Generates a TPoint value from X and Y values
37904: Type System Pointer Defines a general use Pointer to any memory based data
37905: Func Classes PointsEqual Compares two TPoint values for equality
37906: Func System Pos Find the position of one string in another
37907: Func System Pred Decrement an ordinal variable
37908: Func Printers Printer Returns a reference to the global Printer object
37909: Directive Private Starts the section of private data and methods in a class
37910: Keyword System Proc Defines a subroutine that does not return a value
37911: Proc FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
37912: Keyword System Program Defines the start of an application
37913: Func Dialogs PromptForFileName Shows a dialog allowing the user to select a file
37914: Keyword System Property Defines controlled access to class fields
37915: Directive Protected Starts a section of class private data accesible to sub-classes
37916: Type System PShortString A pointer to an ShortString value
37917: Type System PString Pointer to a String value
37918: Func Types PtInRect Tests to see if a point lies within a rectangle
37919: Directive Public Starts an externally accessible section of a class
37920: Directive Published Starts a published externally accessible section of a class
37921: Type System PVariant Pointer to a Variant value
37922: Type System PWideChar Pointer to a WideChar
37923: Type System PWideString Pointer to a WideString value
37924:
37925: Q
37926: Compiler Directive $Q Determines whether Delphi checks Int and enum bounds
37927:
```

```
37928: R
37929: Compiler Directive $R Determines whether Delphi checks array bounds
37930: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
37931: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
37932: Compiler Directive $Resource Defines a resource file to be included in the application linking
37933: Func Math RadToDeg Converts a radian value to degrees
37934: Keyword Raise Raise an exception
37935: Func System Random Generate a random floating point or Int number
37936: Proc System Randomize Reposition the Random number generator next value
37937: Func Math RandomRange Generate a random Int number within a supplied range
37938: Variable System RandSeed Reposition the Random number generator next value
37939: Proc System Read Read data from a binary or text file
37940: Proc System ReadLn Read a complete line of data from a text file
37941: Type System Real A floating point type supporting about 15 digits of precision
37942: Type System Real48 The floating point type with the highest capacity and precision
37943: Proc System ReallocMem Reallocate an existing block of storage
37944: Func DateUtils RecodeDate Change only the date part of a TDateTime variable
37945: Func DateUtils RecodeTime Change only the time part of a TDateTime variable
37946: Keyword Record A structured data type - holding fields of data
37947: Func Classes Rect Create a TRect value from 2 points or 4 coordinates
37948: Func SysUtils RemoveDir Remove a directory
37949: Proc System Rename Rename a file
37950: Func SysUtils RenameFile Rename a file or directory
37951: Keyword Repeat Repeat statements until a ternmination condition is met
37952: Proc SysUtils ReplaceDate Change only the date part of a TDateTime variable
37953: Proc SysUtils ReplaceTime Change only the time part of a TDateTime variable
37954: Proc System Reset Open a text file for reading, or binary file for read/write
37955: Variable System Result A variable used to hold the return value from a function
37956: Proc System ReWrite Open a text or binary file for write access
37957: Proc System RmDir Remove a directory
37958: Func System Round Rounds a floating point number to an Int
37959: Proc System RunError Terminates the program with an error dialog
37960:
37961: S
37962: Constant SysUtils SecsPerDay Gives the number of seconds in a day
37963: Proc System Seek Move the pointer in a binary file to a new record position
37964: Func System SeekEof Skip to the end of the current line or file
37965: Func System SeekEoln Skip to the end of the current line or file
37966: Func FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
37967: Variable System Self Hidden parameter to a method - refers to the containing object
37968: Keyword Set Defines a set of up to 255 distinct values
37969: Func SysUtils SetCurrentDir Change the current directory
37970: Proc System SetLength Changes the size of a string, or the size(s) of an array
37971: Proc System SetString Copies characters from a buffer into a string
37972: Keyword Shl Shift an Int value left by a number of bits
37973: Variable SysUtils ShortDateFormat Compact version of the date to string format
37974: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
37975: Type System ShortInt An Int type supporting values -128 to 127
37976: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
37977: Type System ShortString Defines a string of up to 255 characters
37978: Variable SysUtils ShortTimeFormat Short version of the time to string format
37979: Proc Dialogs ShowMessage Display a string in a simple dialog with an OK button
37980: Proc Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
37981: Proc Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
37982: Keyword Shr Shift an Int value right by a number of bits
37983: Func System Sin The Sine of a number
37984: Type System Single The smallest capacity and precision floating point type
37985: Func System SizeOf Gives the storage byte size of a type or variable
37986: Func System Slice Creates a slice of an array as an Open Array parameter
37987: Type System SmallInt An Int type supporting values from -32768 to 32767
37988: Func System Sqr Gives the square of a number
37989: Func System Sqrt Gives the square root of a number
37990: Proc System Str Converts an Int or floating point number to a string
37991: Type System String A data type that holds a string of characters
37992: Func System StringOfChar Creates a string with one character repeated many times
37993: Func SysUtils StringReplace Replace one or more substrings found within a string
37994: Func System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
37995: Func SysUtils StrScan Searches for a specific character in a constant string
37996: Func SysUtils StrToCurr Convert a number string into a currency value
37997: Func SysUtils StrToDate Converts a date string into a TDateTime value
37998: Func SysUtils StrToDateTime Converts a date+time string into a TDateTime value
37999: Func SysUtils StrToFloat Convert a number string into a floating point value
38000: Func SysUtils StrToInt Convert an Int string into an Int value
38001: Func SysUtils StrToInt64 Convert an Int string into an Int64 value
38002: Func SysUtils StrToInt64Def Convert a string into an Int64 value with default
38003: Func SysUtils StrToIntDef Convert a string into an Int value with default
38004: Func SysUtils StrToTime Converts a time string into a TDateTime value
38005: Func StrUtils StuffString Replaces a part of one string with another
38006: Func System Succ Increment an ordinal variable
38007: Func Math Sum Return the sum of an array of floating point values
38008:
38009: T
38010: Func Math Tan The Tangent of a number
38011: Type Classes TBits An object that can hold an infinite number of Boolean values
38012: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
38013: Type ConvUtils TConvType Defines a measurement type as used by Convert
38014: Type System TDateTime Data type holding a date and time value
38015: Type System Text Defines a file as a text file
38016: Type System TextFile Declares a file type for storing lines of text
```

```
38017: Type SysUtils TFloatFormat Formats for use in floating point number display functions
38018: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
38019: Keyword Then Part of an if statement - starts the true clause
38020: Variable SysUtils ThousandSeparator The character used to display the thousands separator
38021: Keyword ThreadVar Defines variables that are given separate instances per thread
38022: Func SysUtils Time Gives the current time
38023: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
38024: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
38025: Variable SysUtils TimeSeparator The character used to separate display time fields
38026: Func SysUtils TimeToStr Converts a TDateTime time value to a string
38027: Type Classes TList General purpose container of a list of objects
38028: Keyword To Prefixes an incremental for loop target value
38029: Type System TObject The base class type that is ancestor to all other classes
38030: Func DateUtils Tomorrow Gives the date tomorrow
38031: Type Dialogs TOpenDialog Displays a file selection dialog
38032: Type Types TPoint Holds X and Y Int values
38033: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
38034: Type Types TRect Holds rectangle coordinate values
38035: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
38036: Func SysUtils Trim Removes leading and trailing blanks from a string
38037: Func SysUtils TrimLeft Removes leading blanks from a string
38038: Func SysUtils TrimRight Removes trailing blanks from a string
38039: Func System Trunc The Int part of a floating point number
38040: Proc System Truncate Truncates a file size - removes all data after the current position
38041: Keyword Try Starts code that has error trapping
38042: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
38043: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
38044: Type Classes TStringList Holds a variable length list of strings
38045: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
38046: Type System TThreadFunc Defines the Func to be called by BeginThread
38047: Var SysUtils TwoDigitYearCenturyWindow Sets century threshold for 2 digit year string conversions
38048: Keyword Type Defines a new category of variable or process
38049:
38050: U
38051: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
38052: Keyword Unit Defines the start of a unit file - a Delphi module
38053: Keyword Until Ends a Repeat control loop
38054: Func System UpCase Convert a Char value to upper case
38055: Func SysUtils UpperCase Change lower case characters in a string to upper case
38056: Keyword Uses Declares a list of Units to be imported
38057:
38058: V
38059: Proc System Val Converts number strings to Int and floating point values
38060: Keyword Var Starts the definition of a section of data variables
38061: Type System Variant A variable type that can hold changing data types
38062: Func Variants VarType Gives the current type of a Variant variable
38063: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
38064: Directive Virtual Allows a class method to be overriden in derived classes
38065:
38066: W
38067: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
38068: Keyword While Repeat statements whilst a continuation condition is met
38069: Type System WideChar Variable type holding a single International character
38070: Func System WideCharToString Copies a null terminated WideChar string to a normal string
38071: Type System WideString A data type that holds a string of WideChars
38072: Keyword With A means of simplifying references to structured variables
38073: Type System Word An Int type supporting values 0 to 65535
38074: Func SysUtils WrapText Add line feeds into a string to simulate word wrap
38075: Proc System Write Write data to a binary or text file
38076: Proc System WriteLn Write a complete line of data to a text file
38077:
38078: X
38079: Compiler Directive $X Controls some Pascal extension handling
38080: Keyword Xor Boolean Xor or bitwise Xor of two arguments
38081: Y
38082: Compiler Directive $Y Determines whether application symbol information is built
38083: Func DateUtils Yesterday Gives the date yesterday
38084: Z
38085: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
38086:
38087:
38088: Proc SIRegister_uPSUtils(CL: TPSPascalCompiler);
38089: begin     //'TPSBaseType','').SetString( Byte);
38090:  PSMainProcName','String').SetString( '!MAIN;
38091:  PSMainProcNameOrg','String').SetString( 'Main Proc;
38092: 'PSLowBuildSupport','LongInt').SetInt(12);
38093: 'PSCurrentBuildNo','LongInt').SetInt(23);
38094: 'PSCurrentversion','String').SetString('1.31;
38095: 'PSValidHeader','LongInt').SetInt(1397769801);
38096: 'PSAddrStackStart','LongInt').SetInt(1610612736);
38097: 'PSAddrNegativeStackStart','LongInt').SetInt(1073741824);
38098: 'btReturnAddress','LongInt').SetInt(0);
38099: 'btU8','LongInt').SetInt(1);
38100: 'btS8','LongInt').SetInt(2);
38101: 'btU16','LongInt').SetInt(3);
38102: 'btS16','LongInt').SetInt(4);
38103: 'btU32','LongInt').SetInt(5);
38104: 'btS32','LongInt').SetInt(6);
38105: 'btSingle','LongInt').SetInt(7);
```

```
38106: 'btDouble','LongInt').SetInt(8);
38107: 'btExtended','LongInt').SetInt(9);
38108: 'btString','LongInt').SetInt(10);
38109: 'btRecord','LongInt').SetInt(11);
38110: 'btArray','LongInt').SetInt(12);
38111: 'btPointer','LongInt').SetInt(13);
38112: 'btPChar','LongInt').SetInt(14);
38113: 'btResourcePointer','LongInt').SetInt(15);
38114: 'btVariant','LongInt').SetInt(16);
38115: 'btS64','LongInt').SetInt(17);
38116: 'btU64','LongInt').SetInt(30);
38117: 'btChar','LongInt').SetInt(18);
38118: 'btWideString','LongInt').SetInt( 19);
38119: 'btWideChar','LongInt').SetInt( 20);
38120: 'btProcPtr','LongInt').SetInt( 21);
38121: 'btStaticArray','LongInt').SetInt( 22);
38122: 'btSet','LongInt').SetInt( 23);
38123: 'btCurrency','LongInt').SetInt( 24);
38124: 'btClass','LongInt').SetInt( 25);
38125: 'btInterface','LongInt').SetInt( 26);
38126: 'btNotificationVariant','LongInt').SetInt( 27);
38127: 'btUnicodeString','LongInt').SetInt( 28);
38128: 'btType','LongInt').SetInt( 130);
38129: 'btEnum','LongInt').SetInt( 129);
38130: 'btExtClass','LongInt').SetInt( 131);
38131: 'CM_A','LongInt').SetInt( 0);
38132: 'CM_CA','LongInt').SetInt( 1);
38133: 'CM_P','LongInt').SetInt( 2);
38134: 'CM_PV','LongInt').SetInt( 3);
38135: 'CM_PO','LongInt').SetInt( 4);
38136: 'Cm_C','LongInt').SetInt( 5);
38137: 'Cm_G','LongInt').SetInt( 6);
38138: 'Cm_CG','LongInt').SetInt( 7);
38139: 'Cm_CNG','LongInt').SetInt( 8);
38140: 'Cm_R','LongInt').SetInt( 9);
38141: 'Cm_ST','LongInt').SetInt( 10);
38142: 'Cm_Pt','LongInt').SetInt( 11);
38143: 'CM_CO','LongInt').SetInt( 12);
38144: 'Cm_cv','LongInt').SetInt( 13);
38145: 'cm_sp','LongInt').SetInt( 14);
38146: 'cm_bn','LongInt').SetInt( 15);
38147: 'cm_vm','LongInt').SetInt( 16);
38148: 'cm_sf','LongInt').SetInt( 17);
38149: 'cm_fg','LongInt').SetInt( 18);
38150: 'cm_puexh','LongInt').SetInt( 19);
38151: 'cm_poexh','LongInt').SetInt( 20);
38152: 'cm_in','LongInt').SetInt( 21);
38153: 'cm_spc','LongInt').SetInt( 22);
38154: 'cm_inc','LongInt').SetInt( 23);
38155: 'cm_dec','LongInt').SetInt( 24);
38156: 'cm_nop','LongInt').SetInt( 255);
38157: 'Cm_PG','LongInt').SetInt( 25);
38158: 'Cm_P2G','LongInt').SetInt( 26);
38159: TypeS('TbtU8', 'Byte;
38160: TypeS('TbtS8', 'ShortInt;
38161: TypeS('TbtU16', 'Word;
38162: TypeS('TbtS16', 'SmallInt;
38163: TypeS('TbtU32', 'Cardinal;
38164: TypeS('TbtS32', 'Longint;
38165: TypeS('TbtSingle', 'Single;
38166: TypeS('TbtDouble', 'double;
38167: TypeS('TbtExtended', 'Extended;
38168: TypeS('tbtCurrency', 'Currency;
38169: TypeS('tbts64', 'int64;
38170: TypeS('Tbtu64', 'uint64;
38171: TypeS('TbtString', 'string;
38172: Func MakeHash( const s : TbtString) : Longint;
38173: // TbtString = {$IFDEF DELPHI2009UP}Ansistr{$ELSE}String{$ENDIF};
38174: //'PointerSize','LongInt').SetInt( IPointer ( 8 4 ));
38175: end:
38176:
38177: ------------------------------------------------------------------------------
38178: mapX:
38179: if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
38180:  writeln('cologne map found;
38181:  GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
38182:  writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
38183:  OpenMapX('church trier;
38184:  GetGeoCode(C_form,apath:Str; const data:Str; sfile:Bool):Str;
38185:  writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));}
38186:  >>>  //latitude: '50.94133705'  longitude: '6.95812076100766'
38187:  // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
38188:  writeln(GetGeoCoord('xml','church cefalu sicily',true))
38189:  CoordinateStr(Idx: Int; PosInSec: Double; PosLn: TPos):Str;
38190:  Func SendInput( cInputs : UINT; var pInputs : TInput; cbSize : Int) : UINT;
38191:  Func GetLastInputInfo( var plii : TLastInputInfo) : BOOL;
38192:  Proc JvErrorIntercept;
38193:  writeln(GetGeoInfo4('178.196.192.131', UrlGeoLookupInfo3));
38194:
```

```
38195:    {$IFDEF MSWINDOWS}
38196:    Proc Process;
38197:    var
38198:      Msg: TMsg;
38199:    begin
38200:      if ApplicationHasPriority then begin
38201:        Application.ProcessMessages;
38202:      end else begin
38203:        // This guarantees it will not ever call Application.Idle
38204:        if PeekMessage(Msg, 0, 0, 0, PM_NOREMOVE) then begin
38205:          Application.HandleMessage;
38206:        end;
38207:      end;
38208:    end;
38209:
38210:
38211: Ref:
38212:  1. writeln(SHA1(Exepath+'\maxbox4.exe'))
38213:  2. shdig: TSHA1Digest;
38214:     shdig:= GetSHA1OfFile(false,exepath+'\maxbox4.exe;
38215:             for i:= 0 to 19 do write(BytetoHex(shdig[i]));
38216:
38217:  3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox4.exe'),4));
38218:  4. writeln(sha1tohex(SHA1ofStr(loadStringofFile(exepath+'maxbox4.exe'))))
38219:
38220: I have to get only the numbers, without the hyphen and the dots, to my calcs worth.
38221:     sr := '123.456.789-00'
38222:     writeln(ReplaceRegExpr('\D',sr,'',true))
38223:
38224: Func GetLocalStream(C_form:Str; const lat,long:Str):boolean;
38225:  var encodedURL, UrlMapQuestAPI, bufstr:Str;
38226:      mStream: TMemoryStream;
38227:      idhttp: TIdHttp;// THTTPSend;
38228:      len: Int;
38229:  begin
38230:    UrlMapQuestAPI:= 'http://192.168.1.53:9000';
38231:    encodedURL:= UrlMapQuestAPI;
38232:    idHTTP:= TIdHTTP.Create(NIL)
38233:    mStream:= TMemoryStream.create;
38234:    try
38235:      //HttpGet(EncodedURL, mStream);  {WinInet}
38236:      idHTTP.Get1(EncodedURL, mStream)
38237:      mStream.Position:= 0;
38238:      writeln('stream size: '+inttostr(mStream.size));
38239:      //mStream.memory;
38240:      len:= mStream.Size - mStream.Position;
38241:      SetLength(bufstr, len);
38242:      mStream.readbuffer(bufstr, len)
38243:      writeln('debug stream local back: '+bufstr)
38244:    finally
38245:      encodedURL:= '';
38246:      mStream.Free;
38247:      idHTTP.Disconnect
38248:      idHTTP.Free;
38249:      result:= true;
38250:    end;
38251:  end;
38252:
38253: 39/40 PE imports of V maXbox 4.2.5.10
38254:
38255: 1[+] AVICAP32.DLL  2[+] AVICAP32.dll   3[+] GLU32.dll     40[++]dmath.dll
38256: 4[+] IMAGEHLP.DLL  6[+] MSVCRT.DLL     7[+] MSVFW32.DLL
38257: 8[+] OpenGL32.dll  9[+] SHFolder.dll  11[+] URLMON.DLL    10[+] SHfolder.dll
38258: 12[+] advapi32.dll 13[+] comctl32.dll 14[+] comdlg32.dll
38259: 15[+] gdi32.dll    16[+] imagehlp.dll 17[+] imm32.dll
38260: 18[+] iphlpapi.dll 19[+] kernel32.dll 20[+] mpr.dll
38261: 21[+] msacm32.dll  24[+] ole32.dll    25[+] oleacc.dll
38262: 26[+] oleaut32.dll 27[+] oledlg.dll   28[+] opengl32.dll
38263: 29[+] shell32.dll  30[+] shlwapi.dll  31[+] user32.dll
38264: 32[+] usp10.dll    33[+] version.dll  34[+] winhttp.dll
38265: 35[+] wininet.dll  36[+] winmm.dll    37[+] winspool.drv   5[+] KERNEL32.DLL
38266: 38[+] ws2_32.dll   39[+] wsock32.dll  22[+] msimg32.dll   23[+] netapi32.dll
38267:
38268: Magic literal PE32 executable for MS Windows (GUI) Intel 80386 32-bit
38269: ******************************************************************************
38270: Microsoft Windows SDK for Windows 7 and .NET Framework 4 Interface (UI Automation)
38271: Overall test results Number of tests 68 Verify
38272: Passed 67 Failed 1
38273: Unexpected Error 0
38274:
38275: UI Tests caused Unexpected Error: 0
38276: Tests Failed: 1
38277: Test Name : ControlType.Window:SetFocus.1
38278: Result Status: Failed
38279: Test Priority: Pri1
38280: Description: Precondition: Some controls are not persisted after they loose focus (ie, Floating edit in
       Excel), don't tests these
38281: !! Exception !!
38282: Message: Step 12 : Focus event was not fired and was expected to be fired
```

```
38283: Unexpected: false Known Bug: false Incorrect Configuration: false
38284:
38285: 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)
38286:
38287: Tests Passed:67/68
38288: Test Name:ControlType.Window:FocusedElement.1
38289: Test Name:ControlType.Window:GetRuntimeIdType.1
38290: Test Name:ControlType.Window:GetSupportedPatterns.1
38291: Test Name:ControlType.Window:GetSupportedProperties.1
38292: Test Name:ControlType.Window:GetType.1
38293: Test Name:ControlType.Window:HwndWindowRect.1.MSAA
38294: Test Name:ControlType.Window:HwndWindowRect.2.UIA
38295: Test Name:ControlType.Window:LocalizedControlType.1.8.1
38296: Test Name:ControlType.Window:ToString.1
38297: Test Name:ControlType.Window:AutomationIdProperty.1.7.1
38298: Test Name:ControlType.Window:AutomationIdProperty.1.7.8
38299: Test Name:ControlType.Window:BoundingRect.1
38300: Test Name:ControlType.Window:ClickablePointProperty.1
38301: Test Name:ControlType.Window:ControlTypeProperty.1
38302: Test Name:ControlType.Window:GetClickablePoint.2
38303: Test Name:ControlType.Window:GetHashCode.1
38304: Test Name:ControlType.Window:IsKeyboardFocusable.1.1.1
38305: Test Name:ControlType.Window:IsPasswordProperty.1.3.2
38306: Test Name:ControlType.Window:KeyboardFocusable.1.1.2
38307: Test Name:ControlType.Window:Navigation.1
38308: Test Name:ControlType.Window:Navigation.10
38309: Test Name:ControlType.Window:Navigation.11
38310: Test Name:ControlType.Window:Navigation.2
38311: Test Name:ControlType.Window:Navigation.3
38312: Test Name:ControlType.Window:Navigation.4
38313: Test Name:ControlType.Window:Navigation.5
38314: Test Name:ControlType.Window:Navigation.6
38315: Test Name:ControlType.Window:Navigation.7
38316: Test Name:ControlType.Window:Navigation.8
38317: Test Name:ControlType.Window:Navigation.9
38318: Test Name:ControlType.Window:Navigation.ControlFirstChild.1
38319: Test Name:ControlType.Window:Navigation.ControlLastChild.1
38320: Test Name:ControlType.Window:Navigation.ControlNextSibling.1
38321: Test Name:ControlType.Window:Navigation.ControlParent.1
38322: Test Name:ControlType.Window:Navigation.ControlPreviousSibling.1
38323: Test Name:ControlType.Window:SetFocus.10
38324: Test Name:ControlType.Window:SetFocus.3
38325: Test Name:ControlType.Window:AutomationElement.PropertyChange.Enabled.1
38326: Test Name:ControlType.Window:AutomationElement.PropertyChange.Name.1
38327: Test Name:ControlType.Window:GetCurrentPattern.1
38328: Test Name:ControlType.Window:GetCurrentPropertyValue.1
38329: Test Name:ControlType.Window:SetFocus.4
38330: Test Name:ControlType.Window:BulkAdd.1
38331: Test Name:ControlType.Window:BulkRemove.1
38332: Test Name:ControlType.Window:ChildAdd.1
38333: Test Name:ControlType.Window:ChildRemove.1
38334: Test Name:ControlType.Window:Invalidate.1
38335: Test Name:ControlType.Window:Reorder.1
38336: Test Name:ControlType.Window:TestContentView
38337: Test Name:ControlType.Window:TestControlPatterns
38338: Test Name:ControlType.Window:TestControlProperties
38339: Test Name:ControlType.Window:TestControlView
38340: Test Name:ControlType.Window:CanRotate.1.6
38341: Test Name:ControlType.Window:Move.2.1
38342: Test Name:ControlType.Window:Move.2.2
38343: Test Name:ControlType.Window:Move.2.8
38344: Test Name:ControlType.Window:Move.2.13
38345: Test Name:ControlType.Window:Move.2.17
38346: Test Name:ControlType.Window:Move.2.18
38347: Test Name:ControlType.Window:Move.2.19
38348: Test Name:ControlType.Window:Move.2.3
38349: Test Name:ControlType.Window:Move.2.4
38350: Test Name:ControlType.Window:Move.2.5
38351: Test Name:ControlType.Window:Window.MaximizableProperty.S.6.1
38352: Test Name:ControlType.Window:Window.MaximizableProperty.S.6.2
38353: Test Name:ControlType.Window:Window.MinimizableProperty.S.7.2
38354: Test Name:ControlType.Window:Window.ModalProperty.S.10.1
38355:
38356:  Func GetCachedFileFromURL(strUL:Str; var strLocalFile:Str):Bool;
38357:  Func IAddrToHostName(const IP:Str):Str;
38358:  Func GetIEHandle(WebBrowser: TWebbrowser; ClassName:Str): HWND;
38359:  Func GetTextFromHandle(WinHandle: THandle):Str;
38360:  Proc Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
38361:  Func FillWebForm(WebBrowser:TWebBrowser;FieldName:str;Value:str):Bool;
38362:  Proc WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode:Str);
38363:  Func NetSend(dest, Source, Msg:Str): Longint; overload;
```

```
38364:  Func RecordsetFromXML2(const XML:Str): variant;;
38365:  Func RecordsetToXML2(const Recordset: variant):Str;;
38366:  Func GetCharEncoding( alias :Str; var _name :Str) : Int;
38367:  Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
38368:  Func MicrosoftLangageCodeToISOCode( langcode : Int) :Str;
38369:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str = '';
38370:  Func RFC1123ToDateTime(Date:Str): TDateTime;
38371:  Func DateTimeToRFC1123(aDate: TDateTime):Str;
38372:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str);;
38373:  Proc DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);;
38374:  Proc DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);;
38375:  Proc DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);;
38376:  Func checkSystem:Str;
38377:  Func getSystemReport:Str;
38378:
38379: //Commonly used Delphi WinAPI routines
38380: http://www.rosseeld.be/DRO/Delphi/Delphi%20WinAPI.htm
38381:
38382: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
38383:
38384: http://www.slideshare.net/maxkleiner1/codesign-2015
38385: http://basta_2015_speaker_max_kleiner
38386: http://www.arduino.cc/en/Tutorial/ASCIITable
38387: http://www.vwlowen.co.uk/arduino/usb-digital/pc-control.htm
38388: http://www.yunqa.de/delphi/doku.php/products/regex/syntax#quantifiers
38389: http://elib.uni-stuttgart.de/opus/volltexte/2008/3440/pdf/diss_kroell_hs.pdf
38390: http://www.softwareschule.ch/download/exampleedition2016.zip
38391:
38392: Published Doc maXbox Example Edition 2015/2016/2017/2018
38393:
38394:   \example_edition\01_Algorithm;
38395:   \example_edition\02_Graphics;
38396:   \example_edition\03_Games;
38397:   \example_edition\04_Multimedia;
38398:   \example_edition\05_Internet;
38399:   \example_edition\06_Communication;
38400:   \example_edition\07_Geographical;
38401:   \example_edition\08_Operating;
38402:   \example_edition\09_Database;
38403:   \example_edition\10_Science;
38404:   \example_edition\11_Embedded;
38405:   \example_edition\12_Security;
38406:   \example_edition\13_General;
38407:   \example_edition\14_Energy;
38408:   \example_edition\15_Transport;
38409:   \example_edition\16_Robotics;
38410:
38411: Ref:
38412:   if IsValidPeFile(exepath+'maxbox3.exe') then begin
38413:     x1:= ComputePEChecksum(exepath+'maxbox3.exe; // original filename
38414:     x2:= ComputePEChecksum(exepath+'maxbox3.exe;
38415:   end;
38416:   WriteLn('Checksum 1: '+ itoa(x1)+ #13#10+'Checksum 2: '+ itoa(x2));
38417:
38418:   if ConnectDrive('Z:','\\MAXBOX8\Users\Public', True,True) = NO_ERROR then
38419:          writeln('Net Share Z:\ Connected;
38420:      DisconnectNetDrive('Z:', True, True, True);
38421:
38422:     ComTerminal:= TCustomComTerminal.create(self);
38423:       EditComTerminal(comterminal);  //TComTrmSetForm  comterminal.Free;
38424:     LastSysErrorMessage :Str; LastSysErrorMessageA : Ansistr;
38425:
38426:   writeln(sha1tohex(SHA1ofStr(loadstringj(exepath+'maxbox4.exe'))))
38427:   writeln(sha1tohex(SHA1ofStr(loadstringoffile(exepath+'maxbox4.exe'))))
38428:   writeln(strtoHex1(SHA1ofStr(lsof(exepath+'maxbox4.exe'))))
38429:   writeln(sha1tohex(SHA1ofStr(filetoString(exepath+'maxbox4.exe'))))
38430:
38431:   SHA512 sr:= filetostring(Exepath+'maXbox4.exe')
38432:   writeln(uppercase(SHA512DigesttoHEX(CalcSHA512(sr))))
38433:   writeln(Base64EncodeStr((SHA512DigestAsString(CalcSHA512(sr)))));
38434: base64:w4ueRtnUZ641dsy4yIISYvhMB9r0c9kc8YO9o5FgkWQRbayHLaZay7tjnt1JmhV3vck2a7Mec7wfK8PbHacrbQ==
38435:   PE Checksum 1: 27767976 PE Checksum 2: 27767976
38436:
38437: // REST API Example
38438: procedure TOnlineRandomizerGetResultsFromWeb(qUrl: String);
38439: var FResultStream: TMemoryStream;
38440:     Flist: TStringList;
38441:     FResultList: TStringList;
38442:     FSettings:TFormatSettings;
38443: begin
38444:   FResultList:= TSTringList.Create;
38445:   //FSettings:= TFormatSettings.Create;
38446:   //FSettings.DecimalSeparator := '.';
38447:   FResultStream:= TMemoryStream.Create;
38448:   FList:= TStringList.Create;
38449:   //httpObj.Get1(qUrl,FResultStream);
38450:   HTTPget(qURL, FResultStream)
38451:   FResultStream.Position:= 0;
38452:   FList.LoadFromStream(FResultStream);
```

```
38453:   FResultStream.Free;
38454:   FResultList.AddStrings(FList);
38455:   writeln('online random: '+flots(StrToFloat(FResultList[3])));
38456:   writeln('online random: '+flots(StrToFloat(FResultList[4])));
38457:   writeln('online random: '+flots(StrToFloat(FResultList[5])));
38458:   FResultList.Free;
38459:   FList.Free;
38460: end;
38461:
38462: //var FDecUrlPattern:String;
38463: procedure TOnlineRandomizerGetRandomDecimals(dPrecission, dCount: Integer);
38464: var qUrl, FDecUrlPattern: String;
38465: begin
38466:   FDecUrlPattern:= 'https://www.random.org/decimal-fractions/'+
38467:                    '?dec=%d&num=%d&col=1&format=plain&rnd=new';
38468:   qUrl:= Format(FDecUrlPattern,[dPrecission, dCount]);
38469:   TOnlineRandomizerGetResultsFromWeb(qUrl);
38470: end;
38471:
38472: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
38473: ------------------------------------------------------------------------------
38474: ////////////////////////////////////////////////////////////////////////////////
38475: ------------------------------------------------------------------------------
38476: ------------------------------------------------------------------------------
38477: https://www.metadefender.com/#!/results/file/933baa9823514320806c9533482a3b89/regular/analysis
38478: https://www.virustotal.
       com/en/file/b80b0bfef22c6b4be3dbc4af984ca897144895f3c1e162f3ad7895d14fb4e667/analysis/1476479319/
38479: Name Virtual address Virtual size Raw size Entropy MD5
38480: .text 4096 21923324 21923328 6.61 0449d5754ddb069630808bfc6bb356ef
38481: .itext 21929984 52108 52224 6.64 0226e6c46fcf9cc23805e75784f48b4b
38482: .data 21983232 407520 407552 5.25 91d5dac4714aa4119ea18fb5c4ed250d
38483: .bss 22392832 390424 0 0.00 d41d8cd98f00b204e9800998ecf8427e
38484: .idata 22786048 59722 59904 5.53 40cc92b8f087c512630bf807d4d20fe1
38485: .edata 22847488 77 512 0.89 0fcb1fd6f1dde231aa401cffaaa8cd63
38486: .tls 22851584 484 0 0.00 d41d8cd98f00b204e9800998ecf8427e
38487: .rdata 22855680 24 512 0.28 916275c8032d420cb22a05c2086b7e93
38488: .reloc 22859776 1378116 1378304 6.76 51ab0f1c39aa5fc52a7bc82557df173e
38489: .rsrc 24240128 4132864 4132864 5.37 c49d04317b7a6aadc424538ff5ede5c0
38490: CodeSize 21975552 InitializedDataSize 5979648
38491: TimeStamp 2019-08-06 08:06:39
38492: EntryPoint 22518720
38493: OriginalFileName maxbox4_6.exe
38494: MIMEType application/octet-stream
38495:
38496: Article concerning maXbox and mapbox at PHP Magazine 1.2017
38497: https://entwickler.de/php-magazin/php-magazin-1-17-301136.html
38498: WineHQ Update:
38499: https://appdb.winehq.org/objectManager.php?sClass=application&iId=17937
38500: http://www.slideshare.net/maxkleiner1/maxbox-starter46-work-with-wine
38501: https://www.academia.edu/30401550/maXbox_Starter_46_Work_with_WineHQ
38502: https://appdb.winehq.org/appimage.php?iId=50376
38503: https://www.scribd.com/document/333983581/maXbox-starter46-work-with-WineHQ
38504: http://www.softwareschule.ch/download/maxbox_starter46.pdf
38505:
38506: Update for Windows 10 Version 1511 for x64-based Systems (KB3152599)
38507: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3200970)
38508: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3201845)
38509: Update for Windows 10 Version 1607 for x64-based Systems (KB3211320)
38510: Intel Corporation - Display - 11/10/2016 12:00:00 AM - 20.19.15.4549
38511: 2017-10 Cumulative Update for Win 10 Version 1703 for x64-based Systems (KB4041676)
38512: 2017-12 Cumulative Update for Win 10 Version 1703 for x64-based Systems (KB4053580)
38513: Feature update to Windows 10, version 1709
38514: Feature update to Windows 10, version 1803 amd64 2019-04
38515: Feature update to Windows 10, version 1903
38516: Feature update to Windows 10, version 1909
38517: November 12, 2019—KB4524570 (OS Builds 18362.476 and 18363.476)
38518: Version: 1903-OS Build 18362.535 and 1909-OS Build 18363.535
38519: Done Adding Additional Store Successfully signed: maXbox4.exe
38520: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maXbox4.exe
38521: Verifying: maXbox4.exe
38522: Successfully verified: maXbox4.exe
38523: Number of files successfully Verified: 1
38524: Number of warnings: 0
38525: Number of errors: 0C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
38526: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
38527: https://maxbox4.wordpress.com/
38528: http://www.softwareschule.ch/teelicht.jpg
38529:
38530: Verifying: maXbox4.exe   4.7.1.80
38531: Signature Index: 0 (Primary Signature)
38532: Hash of file (sha1): 9D181955E579DCFF6920CEEC32749F0E43F2C157
38533: Signing Certificate Chain:
38534:     Issued to: maXboxCertAuth
38535:     Issued by: maXboxCertAuth
38536:     Expires:   Sun Jan 01 00:59:59 2040
38537:     SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
38538:         Issued to: maXbox4signer
38539:         Issued by: maXboxCertAuth
38540:         Expires:   Sun Jan 01 00:59:59 2040
```

```
38541:         SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
38542:             Issued to: maXbox4exe
38543:             Issued by: maXbox4signer
38544:             Expires:   Sun Jan 01 00:59:59 2040
38545:             SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
38546: The signature is timestamped: Tue Dec 03 10:51:06 2019
38547: Timestamp Verified by:
38548:     Issued to: Starfield Root Certificate Authority - G2
38549:     Issued by: Starfield Root Certificate Authority - G2
38550:     Expires:   Fri Jan 01 00:59:59 2038
38551:     SHA1 hash: B51C067CEE2B0C3DF855AB2D92F4FE39D4E70F0E
38552:         Issued to: Starfield Secure Certificate Authority - G2
38553:         Issued by: Starfield Root Certificate Authority - G2
38554:         Expires:   Sat May 03 08:00:00 2031
38555:         SHA1 hash: 7EDC376DCFD45E6DDF082C160DF6AC21835B95D4
38556:             Issued to: Starfield Timestamp Authority - G2
38557:             Issued by: Starfield Secure Certificate Authority - G2
38558:             Expires:   Tue Sep 17 08:00:00 2024
38559:             SHA1 hash: E8551398FF530A9278FD9818E448CB333F67924D
38560:
38561: Successfully verified: maXbox4.exe
38562: Number of files successfully Verified: 1
38563: Number of warnings: 0
38564: Number of errors: 0
38565: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
38566:
38567: ****************************************************************
38568: Release Notes maXbox 4.7.2.82 April 2020 mX47
38569: ****************************************************************
38570: 1254 unit uPSI_MaskEdit.pas    FCL
38571: 1255 unit uPSI_SimpleRSSTypes; BlueHippo
38572: 1256 unit uPSI_SimpleRSS;      BlueHippo
38573: 1257 unit uPSI_psULib.pas      Prometheus
38574: 1258 unit uPSI_psUFinancial;   Prometheus
38575: 1259 uPSI_PsAPI_2.pas          mX4
38576: 1260 uPSI_PersistSettings_2    mX4
38577: Totals of Func Calls: 32339
38578: for 4.7.2.82
38579: https://sourceforge.net/projects/prometheuslibra/
38580:
38581: CDS Func CreateBlobStream(Field: TField; Mode: TBlobStreamMode): TStream;
38582:     Func BookmarkValid(Bookmark: TBookmark):Bool; ');
38583:     Func CompareBookmarks(Bookmark1, Bookmark2: TBookmark): Integer;
38584:
38585: Proc ModifyFontsFor(ctrl: TWinControl);
38586: Proc ModifyFontsFor(ctrl: TWinControl; afontname:Str);
38587: Proc GetAccessTimeOut(var bTimeOut: Bool; var bFeedBack:Boolean;var iTimeOutTime:Int);
38588: Func GetParentProcessName():Str;
38589: writeln('ParentProcessName: '+GetProcessName(GetParentProcessID(getprocessID)));
38590: Func wFillRect(hDC: HDC; const lprc: TRect; hbr: HBRUSH): Integer; stdcall;
38591: Func wFrameRect(hDC: HDC; const lprc: TRect; hbr: HBRUSH):Integer; stdcall;
38592: Proc GetIpAddresses(Results: TStrings); overload;
38593: Proc GetIpAddresses(Results: TStrings; const HostName: Ansistr);
38594: S.RegisterDelphiFunction(@GetIPAddresses, 'GetIPs', cdRegister);
38595: RegisterMethod(@TStringGrid.CellRect, 'CellRect');
38596: RegisterMethod(@TStringGrid.MouseToCell, 'MouseToCell');
38597: Proc SendKeyVar(AppName:Str; key: Variant);
38598: http://www.scalabium.com/faq/dct0069.htm
38599: Proc GetComponentNames(lst: TStrings);
38600: Proc AngleTextOut(ACanvas: TCanvas; Angle, X, Y: Integer; Str:str);
38601: Proc PrintText2Printer(s:Str);
38602: Proc UASwapEndian(var UC: TUArrDW);
38603: Func SliceByteArray(const B: array of Byte; Start, Len: Integer): TBytes;
38604: Proc TailOfFile(aFilename:str; aTailContainer: TStrings; MaxNumLines:integer=5);
38605: Func CloneByteArray(const B: array of Byte): TBytes;
38606: Func PopByteArray(var A: TBytes): Byte;
38607: Proc PushByteArray(const B: Byte; var A: TBytes);
38608: Proc AppendByteArray(var B1:TBytes; const B2: array of Byte);
38609: Func ShiftByteArray(var A: TBytes): Byte;
38610:
38611: ****************************************************************
38612: Release Notes maXbox 4.7.4.64 June 2020 mX47
38613: ****************************************************************
38614: add Tutor 57 - 72
38615: add NoGUI Shell Tutorial 71 and 70 Units
38616: 1307 unit uPSI_statmach,   {StateMachine}
38617: 1308 uPSI_uTPLb_RSA_Primitives,
38618: 1309 unit uPSI_UMatrix,    //for Tensorflow dll
38619: 1310 uPSI_DXUtil,
38620: 1311 uPSI_crlfParser,
38621: 1312 unit uPSI_DCPbase64;
38622: 1313 unit uPSI_FlyFilesUtils;
38623: 1314 uPSI_PJConsoleApp.pas
38624: 1315 uPSI_PJStreamWrapper.pas
38625: 1316 uPSI_LatLonDist,    //DFF
38626: 1317 uPSI_cHash2.pas     //Fundamentals SHA512
38627: 1318 uPSI_ZLib2.pas      //compressor
38628: 1319 unit uPSI_commDriver
38629: 1320 unit uPSI_PXLNetComs.pas   //PXL
```

```
38630: 1321 unit uPSI_PXLTiming.pas    //PXL
38631: 1322 uPSI_Odometer.pas
38632: 1323 unit uPSI_UIntList2;
38633: 1324 uPSI_UIntegerpartition.pas
38634: 1325 unit uPSI_idPHPRunner.pas  //prepare for PHP4D
38635: 1326 unit uPSI_idCGIRunner.pas
38636: 1327 uPSI_DrBobCGI,             //4.7.1.20
38637: 1328 uPSI_OverbyteIcsLogger,
38638: 1329 uPSI_OverbyteIcsNntpCli,  testset
38639: 1330 uPSI_OverbyteIcsCharsetUtils,
38640: 1331 uPSI_OverbyteIcsMimeUtils,
38641: 1332 uPSI_OverbyteIcsUrl(CL: TPSPascalCompiler);
38642: 1333 uPSI_uWebSocket.pas
38643: 1334 uPSI_KhFunction.pas
38644: 1335 uPSI_ALOpenOffice.pas
38645: 1336 unit uPSI_ALLibPhoneNumber
38646: 1337 unit uPSI_ALPhpRunner2;
38647: 1338 unit uPSI_ALWebSpider2;
38648: 1339 unit uPSI_ALFcnHTML2; // RunJavaScript2
38649: 1340 unit uPSI_ALExecute2.pas
38650: 1341 uPSI_ALIsapiHTTP.pas
38651: 1342 uPSI_ALOpenOffice_Routines
38652: 1343 unit uPSI_uUsb;
38653: 1344 uPSI_uWebcam.pas
38654: 1345 uPSI_PersistSettings.pas //fixing
38655: 1346 uPSI_uTPLb_MemoryStreamPool.pas
38656: 1347 uPSI_uTPLb_Signatory.pas
38657: 1348 unit uPSI_uTPLb_Constants.pas  //TurboPower
38658: 1349 uPSI_uTPLb_Random.pas
38659: 1350 unit uPSI_uTPLb_PointerArithmetic;
38660: 1351 unit uPSI_EwbCoreTools.pas
38661: 1352 unit uPSI_EwbUrl.pas
38662: 1353 unit uPSI_SendMail_For_Ewb.pas
38663: 1354 unit uPSI_MaskEdit.pas    FCL
38664: 1355 unit uPSI_SimpleRSSTypes; BlueHippo
38665: 1356 unit uPSI_SimpleRSS;      BlueHippo
38666: 1357 unit uPSI_psULib.pas      Prometheus
38667: 1358 unit uPSI_psUFinancial;   Prometheus
38668: 1359 uPSI_PsAPI_2.pas          mX4
38669: 1360 uPSI_PersistSettings_2    mX4
38670: 1361 uPSI_rfc1213util2.pas     IP
38671: 1362 uPSI_JTools.pas           JCL
38672: 1363 unit uPSI_neuralbit.pas   CAI
38673: 1364 unit uPSI_neuralab.pas    CAI
38674: 1365 unit uPSI_winsvc2.pas     TEK
38675: 1366 unit uPSI_wmiserv2.pas    TEK
38676: 1367 uPSI_neuralcache.pas      CAI
38677: 1368 uPSI_neuralbyteprediction CAI
38678: 1369 unit uPSI_USolarSystem;   glscene.org
38679: 1370 uPSI_USearchAnagrams.pas  DFF
38680: 1371 uPSI_JsonsUtilsEx.pas     Randolph
38681: 1372 unit uPSI_Jsons.pas       Randolph
38682: 1373 unit uPSI_HashUnit;       DFF
38683: 1374 uPSI_U_Invertedtext.pas   DFF
38684: 1275 unit uPSI_Bricks;         Dendron
38685: 1376 unit uPSI_lifeblocks.pas  Dendron
38686:
38687: Bugfix: Pagecontrol,TTabcontrol with change event TTabChangingEvent,TGetSiteInfoEvent,
38688:   Json extension with objects and arrays
38689:   TCheckListBox.Style  Property TListBoxStyle Enumeration
38690:   glsincos11 glsincos0 fix uPSI_UMatrix; Gaussian_Elimination fix  array[1..30] of Extended
38691:   more ico and bmp resouces, fix Textmetric type and strtochars() alias StrtoCharSet()
38692:   shortcut Ctrl+Alt+M to jump to output (memo2) console
38693:
38694: Func MercScaling(Long0,lat0,long1,lat1:extended;x0,y0,x1,y1:integer) :TMercScalingRec');
38695: Func LonglatToPlotPt(Long,lat:Extended; Scalerec:TMercScalingrec):TPoint');
38696: Func PlotPtToLonglat(PlotPt:TPoint; Scalerec:TMercScalingrec):TRealPoint');
38697: Func SphericalEarthDistance(lat1,lon1,lat2,lon2:extended;Units:integer):extended');
38698: Proc PrintUsingShell(psFileName :str);
38699: Func LocalExecute32(FileName:Str;Wait:bool;Visibility:int;lWaitFor:Card=INFINITE):int;
38700:
38701: TidCGIRunner component allows to execute CGI scripts using
38702: Indy TidHTTPServer component.
38703: // Project:     1307 State Machine
38704: // Module:      statmach.pas
38705: // Description: Visual Finite State Machine.
38706: // Version:     2.2a,  Release: 6 preprocessor, compiler, interpreter
38707: Func VarArrayToStr(const vArray: variant):Str;
38708: Func GetWMIObject(const objectName:Str): IDispatch; //create the Wmi instance
38709: Func GetAntiVirusProductInfo: TStringlist;
38710: Proc GetWaveOutDevices(DeviceNames: TStrings);
38711: Proc GetMIDIOutDevices(Devices: TStrings);
38712: Proc SetDecimalSeparator2(Ch: Char);
38713: Totals of Func Calls: 32356
38714: Proc ResizeBitmap(Bitmap:TBitmap;Width,Height:Integer;Background: TColor);
38715: Func ProcessCount(const ExeName:str): Integer;
38716: Func MaxWidthOfStrings(const Strings:Classes.TStrings;const Font:Graphics.TFont):Integer;
38717: Proc ScreenShotMonitor(var Bitmap:TBitmap;const MonitorNum:Integer;const DrawCursor:Bool;const
       Quality:TPixelFormat);
```

```
38718:
38719: https://github.com/maxkleiner/maXbox/blob/master/logisticregression2.ipynb
38720:
38721: ---- mX4 bigbitbox code_cleared_checked  IMPORTS DLL LIB ----
38722: AVICAP32.DLL  01 [+] AVICAP32.DLL
38723: AVICAP32.dll  02 [+] AVICAP32.dll
38724: GLU32.dll     03 [+] GLU32.dll
38725: IMAGEHLP.DLL  04 [+] IMAGEHLP.DLL
38726: KERNEL32.DLL  05 [+] KERNEL32.DLL
38727: MSVCRT.DLL    06 [+] MSVCRT.DLL
38728: MSVFW32.DLL   07 [+] MSVFW32.DLL
38729: OpenGL32.dll  08 [+] OpenGL32.dll
38730: SHFolder.dll  09 [+] SHFolder.dll
38731: SHfolder.dll  10 [+] SHfolder.dll
38732: URLMON.DLL    11 [+] URLMON.DLL
38733: advapi32.dll  12 [+] advapi32.dll
38734: comctl32.dll  13 [+] comctl32.dll
38735: comdlg32.dll  14 [+] comdlg32.dll
38736: gdi32.dll     15 [+] gdi32.dll
38737: imagehlp.dll  16 [+] imagehlp.dll
38738: imm32.dll     17 [+] imm32.dll
38739: iphlpapi.dll  18 [+] iphlpapi.dll
38740: kernel32.dll  19 [+] kernel32.dll
38741: mpr.dll       20 [+] mpr.dll
38742: msacm32.dll   21 [+] msacm32.dll
38743: msimg32.dll   22 [+] msimg32.dll
38744: netapi32.dll  23 [+] netapi32.dll
38745: ole32.dll     24 [+] ole32.dll
38746: oleacc.dll    25 [+] oleacc.dll
38747: oleaut32.dll  26 [+] oleaut32.dll
38748: oledlg.dll    27 [+] oledlg.dll
38749: opengl32.dll  28 [+] opengl32.dll
38750: shell32.dll   29 [+] shell32.dll
38751: shlwapi.dll   30 [+] shlwapi.dll
38752: user32.dll    31 [+] user32.dll
38753: usp10.dll     32 [+] usp10.dll
38754: version.dll   33 [+] version.dll
38755: winhttp.dll   34 [+] winhttp.dll
38756: wininet.dll   35 [+] wininet.dll
38757: winmm.dll     36 [+] winmm.dll
38758: winspool.drv  37 [+] winspool.drv
38759: ws2_32.dll    38 [+] ws2_32.dll
38760: wsock32.dll   39 [+] wsock32.dll
38761: fannfloat.dll 40 [++]fannfloat.dll
38762: dmath.dll     41 [++]dmath.dll
38763: midas.dll     42 [++]midas.dll
38764:
38765: [+] AVICAP32.DLL    [+] AVICAP32.dll  [+] GLU32.dll      [+] IMAGEHLP.DLL
38766: [+] KERNEL32.DLL    [+] MSVCRT.DLL    [+] MSVFW32.DLL    [+] OpenGL32.dll
38767: [+] SHFolder.dll    [+] SHfolder.dll  [+] URLMON.DLL     [+] advapi32.dll
38768: [+] comctl32.dll    [+] comdlg32.dll  [+] gdi32.dll      [+] imagehlp.dll
38769: [+] imm32.dll       [+] iphlpapi.dll  [+] kernel32.dll   [+] mpr.dll
38770: [+] msacm32.dll     [+] msimg32.dll   [+] netapi32.dll   [+] ole32.dll
38771: [+] oleacc.dll      [+] oleaut32.dll  [+] oledlg.dll     [+] opengl32.dll
38772: [+] shell32.dll     [+] shlwapi.dll   [+] user32.dll     [+] usp10.dll
38773: [+] version.dll     [+] winhttp.dll   [+] wininet.dll    [+] winmm.dll [+] midas.dll
38774: [+] winspool.drv    [+] ws2_32.dll    [+] wsock32.dll    [+] dmath.dll [+] fannfloat.dll
38775:
38776: Exports   CreateIncome      ExePath         GetASCII                GetBiosVendor
38777:           GetMemoryInfo     GetOSName       GetProcessorName        GetScriptName2
38778:           GetScriptPath2    IsCOMConnected  IsInternet              RemainingBatteryPercent
38779:           UpTime            getBigPI        getHostIP               isSound
38780:
38781: memo1.CodeFolding.FoldRegions.Add begin','end''try','end repeat, until''case','end {$IFDEF','{$ENDIF}};
38782: memo1.InitCodeFolding;  true, '{','}
38783: writeln(GETDOSOutput('cmd.exe /c wmic cpu get name','C:\'));
38784: >>> Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
38785: V 4.7.4.64
38786: Totals of Func Calls: 32633
38787: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
38788: CRC32: 3EB27A87:  28.2 MB (29,608,248) bytes
38789: V 4.7.5.20
38790: Totals of Func Calls: 33282
38791: SHA1: of 4.7.5.20 D82EAD01C58738887661428F94B207DB1D8FAEB5
38792: CRC32: 203C82F0  29.5 MB (31,012,768 bytes
38793: V 4.7.5.80
38794: Totals of Calls: 33848
38795: SHA1: of 4.7.5.80 3E38A48072D4F828A4BE4A52320F092FE50AE9C3
38796: CRC32: 198E756B:  31344456 bytes
38797: ▯▯▯ mX4 executed: 27/07/2021 19:05:10  Runtime:0:12:3.247  Memload: 39% use
38798: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool sign /f "maxbox4exe
38799: .pfx" /p "password" /tr http://timestamp.comodoca.com /td SHA256 maXbox4.exe
38800: C:\maXbox\EKON_BASTA\\Windows Kits\10\bin\x64>signtool verify /v /pa maXbox4.exe
38801: For SHA256 the Parameter name has to be /tr. The /t is only valid for SHA1.
38802:  ExecuteShell(OpenSSL_Path+'openssl.exe',
38803: 'pkcs12 -in'+OpenSSL_Path+'./certs/maXboxCertAuth3.pfx -out'+OpenSSL_Path+'./certs/maXboxCertAuth3.pfx.pem
       -nodes')
38804: Verifying: maXbox4.exe 4.7.6.10 IV  Compilation Timestamp 2022-01-14 13:20:37
38805: Entry Point 24467336  Contained Sections 10
```

```
38806:
38807: ****************************************************************
38808: Release Notes maXbox 4.7.6.10 VIII August 2022 mX476
38809: ****************************************************************
38810: Add 48 Units + 16 Tutorials
38811:
38812: 1441 unit uPSI_neuralgeneric.pas; CAI
38813: 1442 unit uPSI_neuralthread.pas; CAI
38814: 1443 unit uPSI_uSysTools; TuO
38815: 1444 unit upsi_neuralsets; mX4
38816: 1445 unit uPSI_uWinNT.pas mX4
38817: 1446 unit uPSI_URungeKutta4.pas ICS
38818: 1447 unit uPSI_UrlConIcs.pas ICS
38819: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
38820: 1449 unit uPSI_Numedit2 mX4
38821: 1450 unit uPSI_PsAPI_3.pas mX4
38822: 1451 unit uPSI_SeSHA256.pas
38823: 1452 unit IdHashMessageDigest_max2;
38824: 1453 unit uPSI_BlocksUnit.pas
38825: 1454 unit uPSI_DelticsCommandLine.pas
38826: 1455 unit uPSI_DelticsStrUtils;
38827: 1456 unit uPSI_DelticsBitField;
38828: 1457 unit uPSI_DelticsSysUtils;
38829: 1458 unit uPSI_ALIniFiles2.pas
38830: 1459 unit uPSI_StarCalc2.pas
38831: 1460 unit uPSI_IdHashMessageDigest2.pas
38832: 1461 unit uPSI_U_Splines;
38833: 1462 unit uPSI_U_CoasterB.pas;
38834: 1463 U_SpringMass2.pas
38835: 1464 uPSI_MARSCoreUtils;
38836: 1465 unit uPSI_clJsonParser.pas
38837: 1466 unit uPSI_SynHighlighterPython.pas
38838: 1467 unit uPSI_DudsCommonDelphi;
38839: 1468 unit uPSI_AINNNeuron;
38840: 1469 unit uPSI_PJConsoleApp2;
38841: 1470 unit uPSI_PJPipeFilters2;
38842: 1471 unit uPSI_uHTMLBuilder;
38843: 1472 unit uPSI_PJPipe2;
38844: 1473 uPSI_WinApiDownload,
38845: 1474 uPSI_pxQRcode,     //beta
38846: 1475 unit uPSI_neuralplanbuilder2
38847: 1476 unit uPSI_DelphiZXingQRCode;
38848: 1477 unit uPSI_RestJsonUtils;
38849: 1478 unit UtilsTimeCode;
38850: 1479 unit uPSC_classes2.pas; //TList
38851: 1480 unit uPSC_std2.pas
38852: 1481 unit uPSI_maxIniFiles.pas
38853: 1482 unit uROPSImports.pas
38854: 1483 unit uROPSServerLink.pas
38855: 1484 unit uPSI_KLibUtils;
38856: 1485 unit uPSI_PathFunc2;  //inno setup
38857: 1486 unit KLibVC_Redist.pas;
38858: 1487 unit HTTPApp2.pas;
38859: 1488 unit uPSI_XCollection2;
38860:
38861: Total of Function Calls: 35597
38862: SHA1: 4.7.6.10 AA16F6BCD100F0AD16A39CD9741E78A96DF5F7F1
38863: CRC32: 412DE996: 31.7 MB (33,331,992 bytes)
38864: Amount of Functions: 21002
38865: Amount of Procedures: 12504
38866: Amount of Constructors: 1980
38867: Amount of Destructors: 14
38868: Totals of Calls: 35500
38869: SHA1: of 4.7.6.10 AEF0EE239713A3E142EE3480F86C30B9F0B872FD
38870: CRC32: 2D7A3208:  33300808 bytes
38871: ▯▯▯ mX4 executed: 19/03/2022 09:21:28  Runtime: 0:13:36.211  Memload: 42% use
38872: Amount of Functions: 21220
38873: Amount of Procedures: 12616
38874: Amount of Constructors: 1992
38875: Amount of Destructors: 14
38876: Totals of Calls: 35842
38877: SHA1: of 4.7.6.10 B8590FA3326B735E14D9687076B302461D8D49D9
38878: CRC32: CCBD8C56:  33472792 bytes
38879: ▯▯▯ mX4 executed: 26/08/2022 09:53:21  Runtime: 0:14:24.171  Memload: 42% use
38880: ▯▯▯
38881: Total of Function Calls: 35842
38882: SHA1: 4.7.6.10 B8590FA3326B735E14D9687076B302461D8D49D9
38883: CRC32: CCBD8C56: 31.9 MB (33,472,792 bytes)
38884: recomplied
38885: 25/08/2022  22:41           483,539 fMain.dcu
38886: 25/08/2022  22:42            71,769 ALHttpClient2.dcu
38887: 25/08/2022  15:48            48,080 ALMultiPartParser.dcu
38888: 25/08/2022  22:11            20,366 ALWininetHttpClient2.dcu
38889: 25/08/2022  22:42            13,520 BlocksUnit.dcu
38890: 25/08/2022  23:24            30,875 MathsLib.dcu
38891: 25/08/2022  22:42             6,566 PXLTiming.dcu
38892: 25/08/2022  22:42             7,431 PythonAction.dcu
38893: 25/08/2022  22:42           279,159 PythonEngine.dcu
38894: 25/08/2022  22:42             6,446 SeSHA256.dcu
```

```
38895: 25/08/2022  20:14            45,539 UBigIntsV4.dcu
38896: 25/08/2022  22:42            66,215 uPSI_ALHttpClient2.dcu
38897: 25/08/2022  14:03            39,724 uPSI_ALHttpCommon.dcu
38898: 25/08/2022  23:22             7,652 uPSI_ALWininetHttpClient2.dcu
38899: 25/08/2022  22:42            28,590 uPSI_AzuliaUtils.dcu
38900: 25/08/2022  22:42            20,494 uPSI_BlocksUnit.dcu
38901: 25/08/2022  22:42             8,645 uPSI_PXLTiming.dcu
38902: 25/08/2022  22:42           175,123 uPSI_PythonEngine.dcu
38903: 25/08/2022  20:14            19,746 uPSI_UBigIntsV4.dcu
38904: 25/08/2022  22:42            46,239 VarPyth.dcu
38905: 25/08/2022  14:03            77,150 OverbyteIcsUtils.dcu
38906: ▯▯▯
38907: Recompiled: dir /s /o N *.dcu | find "03/01/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
38908: 03/01/2022  14:28  479,283 fMain.dcu
38909: 03/01/2022  14:28   13,506 BlocksUnit.dcu
38910: 03/01/2022  13:05   30,110 clJsonParser.dcu
38911: 03/01/2022  15:22   30,878 MathsLib.dcu
38912: 03/01/2022  14:28    6,566 PXLTiming.dcu
38913: 03/01/2022  14:28    7,417 PythonAction.dcu
38914: 03/01/2022  14:28  279,035 PythonEngine.dcu
38915: 03/01/2022  14:28    6,446 SeSHA256.dcu
38916: 03/01/2022  14:28   20,494 uPSI_BlocksUnit.dcu
38917: 03/01/2022  13:50   21,242 uPSI_clJsonParser.dcu
38918: 03/01/2022  14:11   59,984 uPSI_OverbyteIcsUtils.dcu
38919: 03/01/2022  14:28    8,645 uPSI_PXLTiming.dcu
38920: 03/01/2022  14:28  175,131 uPSI_PythonEngine.dcu
38921: 03/01/2022  10:50   37,866 uPSI_U_Splines.dcu
38922: 03/01/2022  14:50   30,632 uPSI_uLkJSON.dcu
38923: 03/01/2022  11:45   44,566 uPSI_VectorLists.dcu
38924: 03/01/2022  14:28   46,239 VarPyth.dcu
38925: 03/01/2022  14:11   40,901 OverbyteIcsMimeUtils.dcu
38926: 03/01/2022  14:28   77,151 OverbyteIcsUtils.dcu
38927: 03/01/2022  15:21  108,526 uPSI_UtilsMax4.dcu
38928: 02/01/2022  11:52   11,355 U_Splines.dcu
38929: 02/01/2022  11:27   24,281 uPSI_neuraldatasets.dcu
38930: 02/01/2022  12:15   77,381 uPSI_neuralvolume.dcu
38931: 01/01/2022  22:44   48,480 U_CoasterB.dcu
38932: 01/01/2022  22:44   34,098 uPSI_U_CoasterB.dcu
38933:
38934: mX4.7.6.10 VI
38935: 16/02/2022  21:29  50,625 uPSI_StrUtils.dcu
38936: 16/02/2022  21:36  59,984 uPSI_OverbyteIcsUtils.dcu
38937: 16/02/2022  15:49  16,601 uPSI_PJConsoleApp.dcu
38938: 16/02/2022  15:27   6,527 uPSI_PJPipe.dcu
38939: 16/02/2022  18:25 105,750 uPSI_PsAPI.dcu
38940: 16/02/2022  21:36  40,891 OverbyteIcsMimeUtils.dcu
38941:
38942: 17/02/2022  11:02 479,639 fMain.dcu
38943: 17/02/2022  09:55  30,889 uPSUtils.dcu
38944: 17/02/2022  10:56   3,723 AINNNeuron.dcu
38945: 17/02/2022  11:02  13,524 BlocksUnit.dcu
38946: 17/02/2022  11:27   9,594 DudsCommonDelphi.dcu
38947: 17/02/2022  11:34  30,868 MathsLib.dcu
38948: 17/02/2022  11:02   6,568 PXLTiming.dcu
38949: 17/02/2022  11:02   7,434 PythonAction.dcu
38950: 17/02/2022  11:02 278,950 PythonEngine.dcu
38951: 17/02/2022  11:02   6,446 SeSHA256.dcu
38952: 17/02/2022  11:02   5,744 uPSI_AINNNeuron.dcu
38953: 17/02/2022  11:02  20,494 uPSI_BlocksUnit.dcu
38954: 17/02/2022  11:23   5,442 uPSI_DudsCommonDelphi.dcu
38955: 17/02/2022  10:07  11,793 uPSI_PJEnvVars.dcu
38956: 17/02/2022  11:02   8,642 uPSI_PXLTiming.dcu
38957: 17/02/2022  11:02 175,123 uPSI_PythonEngine.dcu
38958: 17/02/2022  11:02  46,244 VarPyth.dcu
38959: 17/02/2022  11:02  77,152 OverbyteIcsUtils.dcu
38960:
38961: mX4.7.6.10 VII
38962: 17/03/2022  14:08    194,307 IFSI_WinForm1puzzle.dcu
38963: 17/03/2022  20:30     46,854 uPSC_classes.dcu
38964: 17/03/2022  19:48      5,226 uPSC_std.dcu
38965: 17/03/2022  18:49     53,416 uPSR_classes.dcu
38966: 17/03/2022  20:30     19,528 uPSI_SynEditHighlighter.dcu
38967: 17/03/2022  16:03     11,784 uPSI_SynEditMiscProcs.dcu
38968: 17/03/2022  21:15    173,341 uPSI_TeEngine.dcu
38969: 17/03/2022  21:16     30,854 MathsLib.dcu
38970: 17/03/2022  20:13     77,627 uPSI_neuralvolume.dcu
38971: 17/03/2022  13:10     13,188 uPSI_UcomboV2.dcu
38972: 17/03/2022  13:26     14,813 uPSI_UTGraphSearch.dcu
38973: 17/03/2022  13:22     21,041 UTGraphSearch.dcu
38974: 17/03/2022  08:35    <DIR>          n
38975:
38976: 18/03/2022  18:38      5,234 uPSC_std.dcu
38977: 18/03/2022  18:38     30,854 MathsLib.dcu
38978: 18/03/2022  18:33     77,624 uPSI_neuralvolume.dcu
38979:
38980: mX4.7.6.10 VIII
38981: Recompiled Units:
38982: c:\maXbox>dir /s /o N *.dcu | find "01/08/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
38983: 28/07/2022  18:44           211,349 IFSI_SysUtils_max.dcu
```

```
38984: 28/07/2022  15:56            46,996 uPSC_classes.dcu
38985: 28/07/2022  15:23            49,036 uPSC_DB.dcu
38986: 28/07/2022  16:02            53,529 uPSR_classes.dcu
38987: 28/07/2022  15:23            89,402 uPSR_DB.dcu
38988: 28/07/2022  22:37            11,941 uPSI_SynEditMiscProcs.dcu
38989: 28/07/2022  22:37            26,896 uPSI_StrUtil.dcu
38990: 28/07/2022  14:51             9,910 uPSI_IdThreadSafe.dcu
38991: 31/07/2022  22:43             7,253 KLibTypes.dcu
38992: 31/07/2022  21:46            59,910 uPSI_OverbyteIcsUtils.dcu
38993: 31/07/2022  21:38            10,411 uPSI_XCollection.dcu
38994: 31/07/2022  21:38            40,887 OverbyteIcsMimeUtils.dcu
38995: 01/08/2022  12:28           482,562 fMain.dcu
38996: 01/08/2022  13:37            66,516 uPSI_HTTPApp.dcu
38997: 01/08/2022  12:28            13,517 BlocksUnit.dcu
38998: 01/08/2022  14:19            38,401 KLibUtils.dcu
38999: 01/08/2022  14:22            30,856 MathsLib.dcu
39000: 01/08/2022  12:28             6,566 PXLTiming.dcu
39001: 01/08/2022  12:28             7,430 PythonAction.dcu
39002: 01/08/2022  12:28           278,932 PythonEngine.dcu
39003: 01/08/2022  12:28             6,446 SeSHA256.dcu
39004: 01/08/2022  12:28            20,494 uPSI_BlocksUnit.dcu
39005: 01/08/2022  14:21            21,506 uPSI_KLibUtils.dcu
39006: 01/08/2022  12:22             7,927 uPSI_PathFunc.dcu
39007: 01/08/2022  12:28             8,645 uPSI_PXLTiming.dcu
39008: 01/08/2022  12:28           175,131 uPSI_PythonEngine.dcu
39009: 01/08/2022  12:28            46,238 VarPyth.dcu
39010: 01/08/2022  12:28            77,151 OverbyteIcsUtils.dcu
39011: 03/08/2022  18:25           482,909 fMain.dcu
39012: 03/08/2022  18:24            64,415 AzuliaUtils.dcu
39013: 03/08/2022  18:25            13,523 BlocksUnit.dcu
39014: 03/08/2022  14:27             3,112 KlibConstants.dcu
39015: 03/08/2022  15:15            41,680 KLibWindows.dcu
39016: 03/08/2022  18:36            30,869 MathsLib.dcu
39017: 03/08/2022  18:25             6,568 PXLTiming.dcu
39018: 03/08/2022  18:25             7,434 PythonAction.dcu
39019: 03/08/2022  18:25           278,919 PythonEngine.dcu
39020: 03/08/2022  18:25             6,446 SeSHA256.dcu
39021: 03/08/2022  14:21            22,204 uPSI_ALHttpClient.dcu
39022: 03/08/2022  18:24            28,585 uPSI_AzuliaUtils.dcu
39023: 03/08/2022  18:25            20,494 uPSI_BlocksUnit.dcu
39024: 03/08/2022  16:33            17,512 uPSI_KLibWindows.dcu
39025: 03/08/2022  18:15            59,933 uPSI_OverbyteIcsUtils.dcu
39026: 03/08/2022  18:25             8,644 uPSI_PXLTiming.dcu
39027: 03/08/2022  18:25           175,123 uPSI_PythonEngine.dcu
39028: 03/08/2022  16:35           108,026 uPSI_VectorGeometry.dcu
39029: 03/08/2022  18:25            46,235 VarPyth.dcu
39030: 03/08/2022  18:15            40,883 OverbyteIcsMimeUtils.dcu
39031: 03/08/2022  18:25            77,145 OverbyteIcsUtils.dcu
39032:
39033: Target Machine Intel 386 or later processors and compatible processors
39034: Compilation Timestamp 2022-08-03 16:37:05 UTC
39035: Entry Point 24656140
39036: Contained Sections 10
39037: Target Machine Intel 386 or later processors and compatible processors
39038: Compilation Timestamp 2022-08-29 09:10:00 UTC
39039: Entry Point 24725780
39040: Contained Sections 10
39041:
39042: mX4.7.6.10 X
39043: Recompiled Units:
39044: C:\maXbox>dir /s /o N *.dcu | find "23/09/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
39045: 23/09/2022  16:15           483,806 fMain.dcu
39046: 23/09/2022  10:00           194,750 IFSI_WinForm1puzzle.dcu
39047: 23/09/2022  12:43            81,932 uPSI_Chart.dcu
39048: 23/09/2022  11:39            97,061 uPSI_TeCanvas.dcu
39049: 23/09/2022  11:39           177,404 uPSI_TeEngine.dcu
39050: 23/09/2022  11:20            73,888 uPSI_TeeProcs.dcu
39051: 23/09/2022  16:16            71,782 ALHttpClient2.dcu
39052: 23/09/2022  16:16            13,522 BlocksUnit.dcu
39053: 23/09/2022  17:45            30,862 MathsLib.dcu
39054: 23/09/2022  16:16             6,568 PXLTiming.dcu
39055: 23/09/2022  16:16             7,434 PythonAction.dcu
39056: 23/09/2022  16:16           278,867 PythonEngine.dcu
39057: 23/09/2022  16:06            10,517 RestUtils.dcu
39058: 23/09/2022  16:16             6,446 SeSHA256.dcu
39059: 23/09/2022  16:16            66,632 uPSI_ALHttpClient2.dcu
39060: 23/09/2022  17:45            31,846 uPSI_AzuliaUtils.dcu
39061: 23/09/2022  16:16            20,494 uPSI_BlocksUnit.dcu
39062: 23/09/2022  16:16             8,642 uPSI_PXLTiming.dcu
39063: 23/09/2022  16:16           175,123 uPSI_PythonEngine.dcu
39064: 23/09/2022  16:15            11,240 uPSI_RestUtils.dcu
39065: 23/09/2022  16:16            46,235 VarPyth.dcu
39066: 23/09/2022  15:11            63,422 uPSI_Series.dcu
39067:
39068: Total of Function Calls: 35958
39069: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
39070: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
39071: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
39072: Entry Point 24783132 - Contained Sections 10
```

```
39073:
39074: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maXbox4.exe
39075: Verifying: maXbox4.exe
39076: Signature Index: 0 (Primary Signature)
39077: Hash of file (sha1): 5E556D92930A5ED88CBDAE8E5A174EE8D3161F56
39078:
39079: Signing Certificate Chain:
39080:     Issued to: maXboxCertAuth
39081:     Issued by: maXboxCertAuth
39082:     Expires:   Sun Jan 01 01:59:59 2040
39083:     SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
39084:
39085:         Issued to: maXbox4signer
39086:         Issued by: maXboxCertAuth
39087:         Expires:   Sun Jan 01 01:59:59 2040
39088:         SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
39089:
39090:             Issued to: maXbox4exe
39091:             Issued by: maXbox4signer
39092:             Expires:   Sun Jan 01 01:59:59 2040
39093:             SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
39094:
39095: The signature is timestamped: Mon Oct 17 11:48:48 2022
39096: Timestamp Verified by:
39097:     Issued to: USERTrust RSA Certification Authority
39098:     Issued by: USERTrust RSA Certification Authority
39099:     Expires:   Tue Jan 19 01:59:59 2038
39100:     SHA1 hash: 2B8F1B57330DBBA2D07A6C51F70EE90DDAB9AD8E
39101:
39102:         Issued to: Sectigo RSA Time Stamping CA
39103:         Issued by: USERTrust RSA Certification Authority
39104:         Expires:   Tue Jan 19 01:59:59 2038
39105:         SHA1 hash: 02D65B95E28370C1570095FA88F923DD937FAD8F
39106:
39107:             Issued to: Sectigo RSA Time Stamping Signer #3
39108:             Issued by: Sectigo RSA Time Stamping CA
39109:             Expires:   Thu Aug 11 01:59:59 2033
39110:             SHA1 hash: AB34013AAC4097319F081AF0B318E183F80F7881
39111:
39112: Successfully verified: maXbox4.exe
39113:
39114: Number of files successfully Verified: 1
39115: Number of warnings: 0
39116: Number of errors: 0
39117:
39118: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39119: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39120:
39121: MD5 8990ff3e02c7d38fe8f2f70763f57324
39122: SHA-1 c0459a5dd0ccc12ce9f3cdec82e3e0c920d38d9c
39123: SHA-256 09eb95c4cc3b7fe60772405b170af4705085da89d190213228c65313c8733499
39124: Vhash 0370a6666d5c0d555c051031e0900e5003116012z33034z17013109054843115z10
39125: Authentihash 85f84acb00bf5644194683264bf589f1128cfc29635f7c5b30a36207c526bc0f
39126: Imphash 599a525d65b80ffcce0c92b9562d6bf7
39127: SSDEEP
       393216:HrHoG3YY+kLLN+aINuFwZ8s8PmxLBTKZRPBOi/9YQz0xmwmLlq6D80PRmj3UcWxu:bouck1+dhuBN9b01ED032mVSXGfag
39128: TLSH T19B777C62B2C0D632D0570578CC0BDAD85A6A3D206DB4549779F6BF8E2E311A17F3A2C7
39129: File type Win32 EXE Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
39130: TrID Windows ActiveX control (60.1%)
39131: TrID InstallShield setup (22.2%)
39132: TrID Win64 Executable (generic) (5.4%)
39133: TrID DOS Borland compiled Executable (generic) (5.1%)
39134: TrID Win32 Executable (generic) (2.3%)
39135: File size 32.13 MB (33690392 bytes)
39136:
39137: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maXbox4.exe
39138: Verifying: maXbox4.exe
39139: Signature Index: 0 (Primary Signature)
39140: Hash of file (sha1): 4D224CD33BAC1E61E8BFF4350832BE2C427FFFBD
39141: Signing Certificate Chain:
39142:     Issued to: maXboxCertAuth
39143:     Issued by: maXboxCertAuth
39144:     Expires:   Sun Jan 01 01:59:59 2040
39145:     SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
39146:         Issued to: maXbox4signer
39147:         Issued by: maXboxCertAuth
39148:         Expires:   Sun Jan 01 01:59:59 2040
39149:         SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
39150:             Issued to: maXbox4exe
39151:             Issued by: maXbox4signer
39152:             Expires:   Sun Jan 01 01:59:59 2040
39153:             SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
39154: The signature is timestamped: Sat Oct 15 18:18:09 2022
39155: Timestamp Verified by:
39156:     Issued to: USERTrust RSA Certification Authority
39157:     Issued by: USERTrust RSA Certification Authority
39158:     Expires:   Tue Jan 19 01:59:59 2038
39159:     SHA1 hash: 2B8F1B57330DBBA2D07A6C51F70EE90DDAB9AD8E
39160:         Issued to: Sectigo RSA Time Stamping CA
```

```
39161:         Issued by: USERTrust RSA Certification Authority
39162:         Expires:   Tue Jan 19 01:59:59 2038
39163:         SHA1 hash: 02D65B95E28370C1570095FA88F923DD937FAD8F
39164:             Issued to: Sectigo RSA Time Stamping Signer #3
39165:             Issued by: Sectigo RSA Time Stamping CA
39166:             Expires:   Thu Aug 11 01:59:59 2033
39167:             SHA1 hash: AB34013AAC4097319F081AF0B318E183F80F7881
39168: Successfully verified: maXbox4.exe
39169: Number of files successfully Verified: 1
39170: Number of warnings: 0
39171: Number of errors: 0
39172: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39173:
39174: Borland Conf 1999 Video: https://www.youtube.com/watch?v=m1Gn4echQIE
39175: http://www.softwareschule.ch/examples/classifier_compare2confusion2.py.htm
39176: Over the last few years, it has become more and more common to deploy server side solutions (and in fact
       any type of application) to lightweight containers rather than physical machines or virtual machines, as
       this allows more flexibility(also in terms of testing),a better way to rebuild the same execution
       environment, and more scalability.
39177: https://www.udemy.com/course/learn-coding-from-the-scratch/learn/
39178: - Valuable information-Clear explanations Engaging delivery-Helpful practice activities
39179: - Accurate course description - Knowledgeable instructor
39180:
39181: This Version is dedicated to Katherine Goble Johnson, NASA mathematician
39182: --- The girl who asked questions ---
39183:
39184: https://github.com/joaopauloschuler/neural-api
39185: https://sourceforge.net/projects/cai/
39186: http://www.formatio-reticularis.de/code.html
39187: http://cyberunits.sourceforge.net/
39188: https://torry.net/files/tools/developers/scripters/maxbox4.zip
39189: https://softwareschule.code.blog/
39190: https://maxbox4.wordpress.com/
39191: https://my6.code.blog/2020/01/17/maxbox4machine/
39192: https://www.filecroco.com/download-maxbox/
39193: https://towardsdatascience.com/how-to-implement-machine-learning-for-predictive-maintenance-4633cdbe4860
39194: https://www.udemy.com/course/learn-coding-from-the-scratch/learn/
39195: https://data.world/maxbox
39196: https://data.world/maxbox/maxbox
39197: https://maxbox4.files.wordpress.com/2020/08/twinwizard-copy.jpeg?w=688&h=&zoom=2
39198: https://maxbox4.files.wordpress.com/2020/07/ice4_interlaken_max.jpg?w=768&h=&zoom=2
39199: https://github.com/maxkleiner/maXbox4/blob/master/objectdetector3.ipynb
39200: https://colab.research.google.com/github/maxkleiner/maXbox4/blob/master/objectdetector3.ipynb
39201: https://my6.code.blog/
39202: https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
39203: https://kiosk.entwickler.de/windows-developer-magazin/windows-developer-magazin-10-20/erklaerbare-modelle-
       schaffen-vertrauen/
39204: https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
39205: https://repl.it/@MaxKleiner/machinelearning4#main.py
39206: https://maxbox4.wordpress.com/2020/05/15/ekon-24/
39207: https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.
       ipynb
39208: https://github.com/maxkleiner/maXbox4/blob/master/sentimenttree2.ipynb
39209: https://towardsdatascience.com/6-amateur-mistakes-ive-made-working-with-train-test-splits-916fabb421bb
39210: https://repl.it/@maxbox/machinelearning4#main.py
39211: https://maxbox.sourceforge.io/
39212: https://linuxschweizag.wordpress.com/
39213: https://maxkleiner1.medium.com/
39214: https://ideone.com/hQ3zAd
39215: https://appdb.winehq.org/objectManager.php?sClass=version&iId=39678
39216: https://bitbucket.org/maxbox4/maxbox4/downloads/
39217: https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
39218: https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
39219: https://maxbox4.blogspot.com/
39220: https://wiki.freepascal.org/Python4Delphi
39221: https://www.clevercomponents.com/articles/article052/
39222: https://entwickler-konferenz.de/delphi-innovations-fundamentals/vcl4python/
39223:
39224: Async/Await: Bake the parallel library features into the language/compilers, much like C#, Delphi and
       other languages have done over recent years (ala async, await). That makes it possible/easier to write
       highly scalable servers, over the top of asynchronous I/O. Yes, it's technically possible now, but it's so
       damned difficult to get right/prove it
       to use the Promise pattern, something along these lines - https://github.com/Real-Serious-Games/C-Sharp-
       Promise to handle async tasks.
39225: Benfits: Write scalable task oriented server code without messing with low level threading, write
       responsive non blocking client code. This is something that is possible to do only with the extensive
       support of a compiler and there's absolutely no way to write an async/await clone in Delphi. But… there's
       a simple trick which allows you to write the code in almost the same way. It uses OmniThreadLibrary's
       Async construct and the magic of anonymous methods. Caveats : Microsoft have a patent on Async/Await
39226:
39227: Max Kleiner's professional environment is in the areas of OOP, UML and coding - among other things as a
       trainer, developer and consultant. The focus is on training, IT security and frameworks that work in a
       service-oriented manner. As a lecturer at a university of applied sciences and on a work contract,
       microcontrollers and machine learning have also been added. His published book "Patterns in C #" is a
       constant companion in the digital jungle.
39228: Association Rules uncover the relationship between two or more attributes. It is mainly in the form of- If
       antecedent than consequent.
39229: Picture Exhibition 2021
39230: https://my6.code.blog/2021/11/12/duesseldorf-ekon-25/
```

```
https://github.com/maxkleiner/neural-api
https://my6.code.blog/2021/09/08/improver-4/
https://my6.code.blog/2020/11/09/grauzone/
https://github.com/maxkleiner/DelphiVCL4Python
https://www.scribd.com/document/553749096/Blaise-101-UK-6P-Max-Kleiner-V-Nr91-Tutor-Synthetic-Data-SynDat
https://opensea.io/maxbox4
https://blogs.embarcadero.com/fun-maxbox-is-an-all-in-one-script-engine-application-powered-by-delphi/
https://www.academia.edu/37015210/Machine_Learning_maXbox_starter60_1
https://github.com/maxkleiner/agsi-data
https://www.hybrid-analysis.com/sample/c6e8c0828c61b0a05cb26ebdafe6985983eb8f6b4d92fb99ad2fa62d2ba50bed

Data Scientist also requires strong technical skills in:
1. Math (e.g. linear algebra, calculus and probability).
2. Statistics (e.g. hypothesis testing and summary statistics).
3. Machine learning tools and techniques(e.g.k-nearest neighbours,random forests,ensemble methods,svm).
4. Software engineering skills (e.g.distributed computing,algorithms and data structures).
5. Data mining.
6. Data cleaning and merging.
7. Data visualization (e.g. plottsystem for R ggplot2, py and JavaScript library for manipulating documents based on data d3.js) and reporting techniques.
8. Unstructured data techniques.
9. R and/or SAS languages.
10. SQL databases and database querying languages.
11. Python (most common), C/C++ Java, Perl.
12. Big data platforms like Hadoop, Hive & Pig.
13. Cloud tools like Amazon S3.
14. Pascal Python in maXbox4

Data Analyst
1. Work with IT teams, management and/or data scientists to determine organizational goals.
2. Mine data from primary and secondary sources.
3. Clean and prune data to discard irrelevant information.
4. Analyze and interpret results using standard statistical tools and techniques.
5. Pinpoint trends, correlations and patterns in complicated data sets.
6. Identify new opportunities for process improvement.
7. Provide concise data reports and clear data visualizations for management.
8. Design, create and maintain relational databases and data systems.
9. Triage code problems and data-related issues.
10. check always the semantic consistency (does data make valid sense?)

 When you run a Python script, the interpreter converts a Python program into something that that the computer can understand. Executing a Python program can be done in two ways: calling the Python interpreter with a shebang line, and using the interactive Python shell or to invoke the script file from a script shell editor like crontab or maXbox. I show 4 solutions.
 Suppose you have a python script file httpheader1.py:
 "G:\maXbox\works2022\maxbox4\examples\httpheader1.py"

                ____  ______.
   _____ _____  \   \/  /\_ |__   _______  ___
  /     \\__  \  \     /  | __ \ /  _ \  \/  /
 |  Y Y  \/ __ \_/     \  | \_\ (  <_> >    <
 |__|_|  (____  /___/\  \ |___  /\____/__/\_ \
       \/     \/       \_/     \/           \/


Sort of a microservice you find at:
https://www.academia.edu/31112544/Work_with_microservice_maXbox_starter48.pdf

                        .,,uod8B8bou,,.
              ..,uod8BBBBBBBBBBBBBBBRPFT?l!i:.
         ,=m8BBBBBBBBBBBBBBBRPFT?!||||||||||||||
         !...:!TVBBBRPFT||||||||||!!^^""'   ||||
         !.....:!?|||||!!^^""'             ||||
         !.........||||                    ||||
         !.........||||  ##                ||||
         !.........||||                    ||||
         !.........||||                    ||||
         !.........||||                    ||||
         !.........||||                    ||||
         `.........||||                  ,||||
          .;.......||||               _.-!!||||||
   .,uodWBBBBb.....||||       _.-!!||||||||||:'
!YBBBBBBBBBBBBBb..!||||:..-!!||||||||iof68BBBBBb....
!..YBBBBBBBBBBBBBb!!||||||||iof68BBBBBBRPFT?!::   `.
!....YBBBBBBBBBBBBBbaaitf68BBBBBBRPFT?!::::::::     `.
!......YBBBBBBBBBBBBBBBBBBRPFT?!::::::;::!^"`;:::       `.
!........YBBBBBBBBBBRPFT?!::::::::::^''...::::::;         iBBbo.
`..........YBRPFT?!::::::::::::::::::::::;iof68bo.      WBBBBbo.
  `.........::::::::::::::::::::::::;iof688888888888b.     `YBBBP^'
    `.......::::::::::::::::::;iof688888888888888888888b.     `
      `.....::::::::::;iof688888888888888888888888888888b.
        `....:::;iof688888888888888888888888888888888899fT!
          `..::!8888888888888888888888888888888899fT||^"'
            `' !!988888888888888888888888888899fT||^"'
                `!!88888888888888888899fT||^"'
                  `!988888888899fT||^"'
                    `!9899fT||^"'
                      `!^"'
```

```
 Each service should be independently developed and deployed. No coordination should be needed with other
service teams if no breaking API changes have been made. Each service is effectively it
it's own codebase and lifecycle.

                     ----
                   /~@@~\,
 _______ . _\_\___/\ __ /\___|_|_ . _______
/ ____  |=|       \  <_+>  /       |=|  ____ \
~|    |\|=|======\\_______//======|=|/|    |~
 |_   |    \       |       |       /    |    |
  \==-|     \      |  mX4  |      /     |----|~~)
  |   |      |     |       |     |      |____/~/
  |   |       \____\____/____/      /    / /
  |   |         {----------}       /____/ /
  |___|        /~~~~~~~~~~~~\     |_/~|_|/
   \_/        [/~~~~~||~~~~~\]     /__|\
   | |         |    ||||    |     (/|[[\)
   [_]        |     |  |     |
              |_____|  |_____|
              (_____)  (_____)
              |     |  |     |
              |     |  |     |
              |/~~~\|  |/~~~\|
              /|___|\  /|___|\
             <_______><_______>

 A microservice architecture shifts around complexity. Instead of a single complex system, you have a
bunch of simple services with complex interactions.

  Simplify API development for users, teams, and enterprises with the Swagger open source and professional
toolset. Find out how Swagger can help you design, document your APIs at scale.

 >>>

                _od#HMM6&*MMMH::-_
          _dHMMMR??MMM? ""| `"'-?Hb_
       .~HMMMMMMMHMMM#M?        `*HMb.
     ./?HMMMMMMMMMM"*"""           &MHb.
    /'|MMMMMMMMMM'              -   `*MHM\
   /  |MMM'MMHHM''                   .MMMHb
  |   9HMMP   .Hq,                   TMM'MMH
 /     |MM\,H-""&&6\__               `MMMMMMb
|       `""HH#,        \             - MMMMMMM|
|            `HoodHMM###.               `9MMMMMH
|              .MMMMMMMM##\               `*"?HM
|          ..  ,HMMMMMMMMMMMo\.               |M
|              |MMMMM'MMMMMMM'MNHo            |M
|               ?MMMMMMM'MMMMMMMM*            |H
|.               `#MMMMMMMM'MMMM'            .M|
 \                 `MMMMMMMMMMM*             |P
 `\                 MMMMMMMMT"'             ,H
  `\               `MMM'MMH?               ./
    \.             |MMMH#"                ,/
     `\.           |MMP'                ./'
       `~\         `HM:.-    .         ,/'
          "-\_       '_\ .        _.-"
              "-\-#odMM\_,oo==-"


 This is what we want to use in maXbox:

 const res = await fetch("https://libretranslate.com/translate", {
 method: "POST",
 body: JSON.stringify({
  q: "Hello!",
  source: "en",
  target: "es"
 }),
 headers: { "Content-Type": "application/json" }
 });
 console.log(await res.json());
```