```
 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:32.4 MB(34,057,496) V4.7.6.20 Dec.2022 EKON26/BASTA/JAX/IBZ/SWS/PASCON
 9: *****************Now the Funclist*********************************************
10: Funclist Func : 21460_; Source Compiled: 2'712'783 lines
11: *****************Now the Proclist*********************************************
12: Proclist Proc Size is: 12722_
13: *****************Now Constructors*********************************************
14: Constructlist Constructor Size is: 2009_
15: def head: max: maXbox10: 15/12/2022 21:35:42
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: 36205! Constructor, Func and Procedure
22: AExtraxt of EXE Functions of maXbox4.exe, locs of file = 39966
23: ASize of EXE: 34,057,496 bytes  CRC32: 4A7231F8
24: SHA1: of maXbox4.exe (4.7.6.20) BF042A1DC1C479C706100BB83F8DEE351554F74B
25: SHA-256 102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3
26:
27: https://www.hybrid-analysis.com/sample/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3
28: https://www.virustotal.
    com/gui/file/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3/detection
29: https://www.hybrid-analysis.com/sample/66e02bfc560c5082c2bfcb5e4b8e2a7b19fa1f60386474097625d55c7cc6b472
30: https://www.virustotal.
    com/gui/file/66e02bfc560c5082c2bfcb5e4b8e2a7b19fa1f60386474097625d55c7cc6b472/detection
31: https://www.virustotal.
    com/gui/file/2acfa55e4bb3ae08a46cf2a691c8f62a6404d84db3ae6fb4bb3f7042149bfb73/detection
32: https://www.virustotal.
    com/gui/file/212f8fda437e877c7f89f771ce40e57814cfb4380ee010c090df78d8dbbc4d9f/detection
33: https://www.virustotal.
    com/gui/file/4f4219830b728537ef6c4ddec92c2fbdfd8f6658a2f03c3981fdfcc833ae8c1e/detection
34: https://www.virustotal.com/gui/file/c6e8c0861b0a05cb26ebdafe6985983eb8f6b4d92fb99ad2fa62d2ba50bed/detection
35: https://www.virustotal.com/gui/file/5d231d7b91a8e1d79ee2b1f19279a0dd2685c7c99974b0da9df749347fc58/detection
36: https://www.virustotal.
    com/gui/file/09eb95c4cc3b7fe60772405b170af4705085da89d190213228c65313c8733499/details
37: https://www.virustotal.
    com/gui/file/e35b4b6cde46d1c0c8a0c30bc125d76d6738e9d0541523214c12d0aa1ba91ca9/details
38: https://www.virustotal.
    com/gui/file/1a2ed2b64f9b7d68e1a16158d06742eef1cde94864a50ce29f08bfadaf402a44/detection
39: https://www.virustotal.com/gui/file/ec6aeb784bdf2dec91834378ef46be698cc0d0eddad9e6488c7fcafac7728/detection
40: https://www.virustotal.
    com/gui/file/9b4ce8468d70b11a0b111009ccf9b4a9e1daf6f9ec53680bf301256e5f69518a/detection
41: https://www.virustotal.
    com/gui/file/db0c72990863b4a08341410b51d76a00632519c0f1df9cc4d63a40c28f2c9778/detection
42: https://www.virustotal.com/gui/file/52a07df33308c2ded182c50194a4c731f9921461ba1cf26625005ec0db672/detection
43: https://www.virustotal.
    com/gui/file/e99be9470f26f69169cc81dfbd8984c7a27dd3fb4b16941cb4028e5dbba66957/detection
44: ////////////////////////////////////////////////////////////////////////////
45:
46: Func Metric of Script: 256_findfunctions476_of_EXE_80_1.txt
47: Func *************Now the Funclist*****************
48: Func  GetResStringChecked(Ident:Str; const Args: array of const):str
49: Func ( Index : Longint) : Integer
50: Func (Command: Word; Data: Longint; var CallHelp:Bool):Bool
51: Func _CheckAutoResult( ResultCode : HResult) : HResult
52: Func _T(Name: tbtString): Variant;
53: Func ABNFToText(const AText :Str) :Str
54: Func Abs(e : Extended) : Extended;
55: Func AbsInt(e : Extended) : Integer;
56: Func AbsInt( const B : integer) : integer;
57: Func AbsFloat( const B : double) : extended;
58: Func Ackermann( const A, B : Integer) : Integer
59: Func AcquireLayoutLock :Bool
60: Func ActionByName( const AName :Str) : TWebActionItem
61: Func ACTIVEBUFFER : PCHAR
62: Func Add:TAggregate
63: Func Add:TCollectionItem
64: Func Add:TColumn
65: Func Add:TComboExItem
66: Func Add:TCookie
67: Func Add:TCoolBand
68: Func Add:TFavoriteLinkItem
69: Func Add:TFileTypeItem
70: Func Add:THeaderSection
71: Func Add:THTMLTableColumn
72: Func Add:TIdEMailAddressItem
73: Func Add:TIdMessagePart
74: Func Add:TIdUserAccount
75: Func Add:TListColumn
76: Func Add:TListItem
77: Func Add:TStatusPanel
78: Func Add:TTaskDialogBaseButtonItem
```

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

```
167: Func AnsiReplaceText( const AText, AFromText, AToText :Str) :Str
168: Func AnsiResemblesText( const AText, AOther :Str) :Bool
169: Func AnsiReverseString( const AText : Ansistr) : Ansistr
170: Func AnsiRightStr( const AText : Ansistr; const ACount : Int) : Ansistr
171: Func AnsiSameCaption(const Text1:Str; const Text2:Str):Bool)
172: Func AnsiSameStr( S1, S2 :Str) :Bool
173: Func AnsiSameStr(const S1:Str; const S2:Str):Bool)
174: Func AnsiSameText( const S1, S2 :Str) :Bool
175: Func AnsiSameText( S1, S2 :Str) :Bool
176: Func AnsiSameText(const S1:Str; const S2:Str):Bool)
177: Func AnsiStartsStr( const ASubText, AText :Str) :Bool
178: Func AnsiStartsText( const ASubText, AText :Str) :Bool
179: Func AnsiStrComp( S1, S2 : PChar) : Int
180: Func AnsiStrComp(S1: PChar; S2: PChar): Int)
181: Func AnsiStrIComp( S1, S2 : PChar) : Int
182: Func AnsiStrIComp(S1: PChar; S2: PChar): Int)
183: Func AnsiStrLastChar( P : PChar) : PChar
184: Func AnsiStrLastChar(P: PChar): PChar)
185: Func AnsiStrLComp( S1, S2 : PChar; MaxLen :Card) : Int
186: Func AnsiStrLIComp( S1, S2 : PChar; MaxLen :Card) : Int
187: Func AnsiStrLower( Str : PChar) : PChar
188: Func AnsiStrPos( Str, SubStr : PChar) : PChar
189: Func AnsiStrPos(Str: PChar; SubStr: PChar): PChar)
190: Func AnsiStrScan(Str: PChar; Chr: Char): PChar)
191: Func AnsiStrUpper( Str : PChar) : PChar
192: Func AnsiToUtf8( const S :Str) : UTF8String
193: Func AnsiToUtf8Ex( const S :Str; const cp : Int) : UTF8String
194: Func AnsiUpperCase( S :Str) :Str
195: Func AnsiUppercase(s :Str) :Str;
196: Func AnsiUpperCaseFileName( S :Str) :Str
197: Func ApplyUpdates(const Delta:OleVariant;MaxErrors:Int;out ErrorCount:Int): OleVariant
198: Func ApplyUpdates(const Delta:OleVariant;MaxErrors: Int;out ErrorCount:Int): OleVariant;
199: Func ApplyUpdates( MaxErrors : Int) : Int
200: Func ApplyUpdates1(const Delta:OleVar;MaxErrs:Int;out ErrCount:Int;var OwnerData:OleVar):OleVariant;
201: Func ArcCos(const X : Extended) : Extended
202: Func ArcCosh(const X : Extended) : Extended
203: Func ArcCot(const X : Extended) : Extended
204: Func ArcCotH(const X : Extended) : Extended
205: Func ArcCsc(const X : Extended) : Extended
206: Func ArcCscH(const X : Extended) : Extended
207: Func ArcSec(const X : Extended) : Extended
208: Func ArcSecH(const X : Extended) : Extended
209: Func ArcSin(const X : Extended) : Extended
210: Func ArcSinh(const X : Extended) : Extended
211: Func ArcTan(const X : Extended) : Extended
212: Func ArcTan2(const Y, X : Extended) : Extended
213: Func ArithmeticMean(const X : TDynDoubleArray) : Float
214: Func ArrayLength: Int;
215: Func AsHex( const AValue : T4x4LongWordRecord):Str
216: Func AsHex( const AValue : T5x4LongWordRecord):Str
217: Func ASNDecLen( var Start : Int; const Buffer :Str) : Int
218: Func ASNDecOIDItem( var Start : Int; const Buffer :Str) : Int
219: Func ASNEncInt( Value : Int) :Str
220: Func ASNEncLen( Len : Int) :Str
221: Func ASNEncOIDItem( Value : Int) :Str
222: Func ASNEncUInt( Value : Int) :Str
223: Func ASNItem(var Start: Int; const Buffer:str; var ValueType:Int):str
224: Func ASNObject( const Data :Str; ASNType : Int) :Str
225: Func Assigned(I: Longint):Bool;
226: Func AspectRatio(aWidth, aHeight: Int):Str;
227: Func AsWideString( Field : TField) : WideString
228: Func AtLeast( ACount : Int) :Bool
229: Func AttemptToUseSharedMemoryManager :Bool
230: Func Authenticate :Bool
231: Func AuthenticateUser( const AUsername, APassword :Str) :Bool
232: Func Authentication :Str
233: Func BatchMove( ASource : TBDEDataSet; AMode : TBatchMode) : Longint
234: Func BcdCompare( const bcd1, bcd2 : TBcd) : Int
235: Func BcdFromBytes( const AValue : TBytes) : TBcd
236: Func BcdPrecision( const Bcd : TBcd) : Word
237: Func BcdScale( const Bcd : TBcd) : Word
238: Func BcdToBytes( const Value : TBcd) : TBytes
239: Func BCDToCurr( const BCD : TBcd; var Curr : Currency) :Bool
240: Func BcdToDouble( const Bcd : TBcd) : Double
241: Func BcdToInt( const Bcd : TBcd; Truncate :Bool) : Int
242: Func BcdToStr( const Bcd : TBcd) :Str;
243: Func BcdToStrF(const Bcd:TBcd;Format:TFloatFormat;const Precision,Digits:Int):str
244: Func beep2(dwFreq, dwDuration: Int):Bool;
245: Func BeginPeriod( const Period :Card) :Bool
246: Func BeginTrans : Int
247: Func BeginTransaction : TDBXTransaction;
248: Func BeginTransaction1( Isolation : TDBXIsolation) : TDBXTransaction;
249: Func BigMulu(aone, atwo:Str):Str;
250: Func BigNumber(aone, atwo:Str):Str;
251: Func BigExp(aone,atwo:Str):Str;
252: Func BigMul(aone,atwo:Str):Str;
253: Func BigAdd(aone,atwo:Str):Str;
254: Func BigSub(aone,atwo:Str):Str;
255: Func BigFactorial(aone:Str):Str;
```

```
256: Func BinaryToDouble( ABinary :Str; DefValue : Double): Double
257: Func BinomialCoeff( N, R :Card) : Float
258: Func BinominalCoefficient(n, k: Int):Str;
259: Func BinStrToInt( const ABinary :Str) : Int
260: Func BinToByte(Binary:Str): Byte;
261: Func BinToHex2(Binary:Str):Str;
262: Func BinToInt(Binary:Str): Int;
263: Func BinToChar(St:Str): Char;
264: Func BinToStr(ans:Str):Str;
265: Func BitBlt(hdcDest:HDC;nXDest,nYDest,nWidth,nHeigh:Int;hdcSrc:HDC;nXSrc,nYSrc:Int;dwRop:DWORD):Bool;
266: Func BitmapsAreIdentical( ABitmap1, ABitmap2 : TBitmap) :Bool
267: Func BitsHighest( X: Byte) : Int;
268: Func BitsHighest1(X: ShortInt) : Int;
269: Func BitsHighest2(X: SmallInt) : Int;
270: Func BitsHighest3(X: Word) : Int;
271: Func BitsHighest4(X: Int) : Int;
272: Func BitsHighest5(X:Card) : Int;
273: Func BitsHighest6(X: Int64) : Int;
274: Func BitsLowest( X: Byte) : Int;
275: Func BitsLowest1(X: Shortint) : Int;
276: Func BitsLowest2(X: Smallint) : Int;
277: Func BitsLowest3(X: Word) : Int;
278: Func BitsLowest4(X:Card) : Int;
279: Func BitsLowest5(X: Int) : Int;
280: Func BitsLowest6(X: Int64) : Int;
281: Func BitsNeeded( const X : Byte) : Int;
282: Func BitsNeeded1(const X : Word) : Int;
283: Func BitsNeeded2(const X : Int) : Int;
284: Func BitsNeeded3(const X : Int64) : Int;
285: Func BlueComponent(const Color32 : TColor32) : Int
286: Func BooleanToInt( const Pb :Bool) : Int
287: Func BoolToStr(B:Bool; UseBoolStrs:Bool):Str)
288: Func BoolToStr1(value :Bool) :Str;
289: Func booltoint( aBool :Bool) : LongInt
290: Func inttobool( aInt : LongInt) :Bool
291: Func Bounds( ALeft, ATop, AWidth, AHeight : Int) : TRect
292: Func Bounds(ALeft, ATop, AWidth, AHeight: Int): TRect)
293: Func BreakApart(BaseString,BreakString:str;StringList: TStrings): TStrings
294: Func BrightColor( const Color : TColor; const Pct : Single) : TColor
295: Func BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
296: Func BufferRequest( Length : Int) : TStream
297: Func BuildFileList(const Path:Str;const Attr:Int;const List:TStrings):Bool
298: Func Buttons: PTaskDialogButton
299: Func BytesPerScanline(PixelsPerScanline,BitsPerPixel,Alignment:Longint):Longint
300: Func BytesToCardinal( const AValue : TIdBytes; const AIndex : Int) :Card
301: Func BytesToChar( const AValue : TIdBytes; const AIndex : Int) : Char
302: Func BytesToInt64( const AValue : TIdBytes; const AIndex : Int) : Int64
303: Func BytesToInt( const AValue : TIdBytes; const AIndex : Int) : Int
304: Func BytesToIPv6( const AValue : TIdBytes; const AIndex : Int) : TIdIPv6Address
305: Func BytesToShort( const AValue : TIdBytes; const AIndex : Int) : Short
306: Func BytesToString(ABytes:TIdBytes; AStartIndex:Int; AMaxCount:Int):Str;
307: Func BytesToStr(const Value: TBytes):Str;
308: Func BytesToWord( const AValue : TIdBytes; const AIndex : Int) : Word
309: Func ByteToBin(Int: Byte):Str;
310: Func ByteToCharIndex( S :Str; Index : Int) : Int
311: Func ByteToCharIndex(const S:Str; Index: Int): Int)
312: Func ByteToCharLen( S :Str; MaxLen : Int) : Int
313: Func ByteToCharLen(const S:Str; MaxLen: Int): Int)
314: Func ByteToHex( const AByte : Byte) :Str
315: Func ByteToOctal( const AByte : Byte) :Str
316: Func ByteType( S :Str; Index : Int) : TMbcsByteType
317: Func ByteType(const S:Str; Index: Int): TMbcsByteType)
318: Func CalcTitleRect( Col : TColumn; ARow : Int; var MasterCol:TColumn):TRect
319: Func CalculateDFAFingerprint( oStates : TList) : Int
320: Func CallTerminateProcs:Bool)
321: Func CANFOCUS:BOOLEAN
322: Func CanLoad( const Ext :Str) :Bool
323: Func CanParse( AWebRequest : TWebRequest) :Bool
324: Func CanSave( const Ext :Str) :Bool
325: Func CanStart( cChar : char) :Bool
326: Func CaptureScreen : TBitmap;
327: Func CaptureScreen1( Rec : TRect) : TBitmap;
328: Func CardinalToFourChar( ACardinal : LongWord) :Str
329: Func CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Bool):Bool
330: Func CastSoapToVariant1(SoapInfo:PTypeInfo;const SoapData:WideString):Variant;
331: Func Ceil( const X : Extended) : Int
332: Func Ceil16( X : Int) : Int
333: Func Ceil4( X : Int) : Int
334: Func Ceil8( X : Int) : Int
335: Func Ceiling( const X : Extended) : Int
336: Func CellRect( ACol, ARow : Longint) : TRect
337: Func CelsiusToFahrenheit( const AValue : Double) : Double
338: Func CenterPoint( const Rect : TRect) : TPoint
339: Func CenterPoint(const Rect: TRect): TPoint)
340: Func ChangeFileExt( FileName, Extension :Str) :Str
341: Func ChangeFileExt(const FileName:Str; const Extension:Str):Str)
342: Func CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) :Bool
343: Func CharInSet3( C : Char; const CharSet : TSysCharSet) :Bool
344: Function CharInSet4(C: Char; const CharSet: CharSet):Bool;
```

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

```
434: Func CompatibleConversionType(const AType:TConvType; const AFamily : TConvFamily):Boolean
435: Func CompatibleConversionTypes( const AFrom, ATo : TConvType) :Bool
436: Func ComponentTypeToString( const ComponentType : DWORD) :Str
437: Func ComposeDateTime(Date,Time : TDateTime) : TDateTime;;
438: Func ComponentToStringProc(Component: TComponent):Str;
439: Func StringToComponentProc(Value:Str): TComponent;
440: Func CompToCurrency( Value : Comp) : Currency
441: Func CompToDouble( Value : Comp) : Double
442: Func ComputeFileCRC32(const FileName :Str) : Int;
443: Func ComputeSHA256(astr:Str; amode: char):Str)  //mode F:File, S:str
444: Func ComputeSHA512(astr:Str; amode: char):Str)  //mode F:File, S:str
445: Func ComPortSelect: Int;  // Search for the first available port
446: Func Concat(s:Str):Str
447: Func ConnectAndGetAll :Str
448: Func Connected :Bool
449: Func constrain(x, a, b: Int): Int;
450: Func ConstraintCallBack(Req:DsInfoReq;var ADataSources:DataSources): DBIResult
451: Func ConstraintsDisabled :Bool
452: Func CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
453: Func ContainsState( oState : TniRegularExpressionState) :Bool
454: Func ContainsStr( const AText, ASubText :Str) :Bool
455: Func ContainsText( const AText, ASubText :Str) :Bool
456: Func ContainsTransition(oTransition:TniRegularExpressionTransition):boolean
457: Func Content:Str
458: Func ContentFromStream( Stream : TStream) :Str
459: Func ContentFromString( const S :Str) :Str
460: Func CONTROLSDISABLED :Bool
461: Func Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
462: Func Convert1(const AValue:Double;const AFrom1,AFrom2,ATo1,ATo2:TConvType):Double;
463: Func ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
464: Func ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Int) : Int
465: Func ConvertTo( const AValue : Double; const ATo : TConvType) : Double
466: Func ConvertWriteStream( Stream : TStream; Buffer: PChar; BufSize : Int) : Int
467: Func ConvFamilyToDescription( const AFamily : TConvFamily) :Str
468: Func ConvTypeToDescription( const AType : TConvType) :Str
469: Func ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
470: Func ConvTypeToFamily( const AType : TConvType) : TConvFamily;
471: Func ConvAdd(const AVal:Dbl;const AType1:TConvType;const AVal2:Dbl;const AType2,
     AResultType:TConvType):Double
472: Func ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
     AType2:TConvType):TValueRelationship
473: Func ConvDec( const AValue:Double;const AType,AAmountType:TConvType):Double;
474: Func ConvDec1(const AValue:Dbl;const AType:TConvType;const AAmount:Dble;const
     AAmountType:TConvType):Double;
475: Func ConvDiff(const AVal1:Dbl;const AType1:TConvType;const AVal2:Dble;const AType2,
     AResuType:TConvType):Double
476: Func ConvInc( const AValue:Double; const AType,AAmountType:TConvType):Double;
477: Func ConvInc1(const AValue:Dbl;const AType:TConvType;const AAmount:Double;const
     AAmountType:TConvType):Double;
478: Func ConvSameValue(const AValue1:Dbl;const AType1:TConvType;const AValue2:Dbl;const AType2:TConvType):Bool
479: Func ConvToStr( const AValue : Double; const AType : TConvType) :Str
480: Func ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmount : Double; const
     AAmountType : TConvType) :Bool
481: Func ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType;const AAmount:Double;const
     AAmountType: TConvType) :Bool
482: Func Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
483: Func CopyFile( Source, Dest :Str; CanOverwrite :Bool) : Bool
484: Func CopyFileEx( Source, Dest :Str; Flags : FILEOP_FLAGS) : Bool
485: Func CopyFileTo( const Source, Destination :Str) :Bool
486: Func CopyFrom(Source:TStream;Count:Int64):LongInt
487: Func CopyPalette( Palette : HPALETTE) : HPALETTE
488: Func CopyTo( Length : Int) :Str
489: Func CopyTo(stm:IStream;cb:Largeint;out cbRead:Largeint;out cbWritten:Largeint):HResult
490: Func CopyToEOF :Str
491: Func CopyToEOL :Str
492: Func Cos(e : Extended) : Extended;
493: Func Cosecant( const X : Extended) : Extended
494: Func Cot( const X : Extended) : Extended
495: Func Cotan( const X : Extended) : Extended
496: Func CotH( const X : Extended) : Extended
497: Func Count : Int
498: Func CountBitsCleared( X:Byte) : Int;
499: Func CountBitsCleared1(X:Shortint) : Int;
500: Func CountBitsCleared2(X:Smallint) : Int;
501: Func CountBitsCleared3(X:Word) : Int;
502: Func CountBitsCleared4(X:Int) : Int;
503: Func CountBitsCleared5(X:Card) : Int;
504: Func CountBitsCleared6(X:Int64) : Int;
505: Func CountBitsSet( X:Byte) : Int;
506: Func CountBitsSet1(X:Word) : Int;
507: Func CountBitsSet2(X:Smallint) : Int;
508: Func CountBitsSet3(X:ShortInt) : Int;
509: Func CountBitsSet4(X:Int) : Int;
510: Func CountBitsSet5(X:Card) : Int;
511: Func CountBitsSet6(X:Int64) : Int;
512: Func countDirfiles(const apath:Str): Int;
513: Func CountGenerations(Ancestor,Descendent: TClass): Int
514: Func Coversine( X : Float) : Float
515: Func CRC32(const fileName:Str): LongWord;
```

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

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

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

```
781: Proc ExecuteThread2(afunc:TThreadFunction2;thrOK:bool);AddTypeS('TThreadFunction2','procedure
782: Func CreateThread2(ThreadFunc: TThreadFunction2) : THandle
783: Func ExeFileIsRunning(ExeFile:Str):Bool;
784: Func ExePath:Str;
785: Func ScriptPath:Str;
786: Func ScriptName:Str;
787: Func ExePathName:Str;
788: Func Exists( AItem : Pointer) :Bool
789: Func ExitWindows( ExitCode :Card) :Bool
790: Func Exit2( ExitCode :Card) :Bool
791: Func Exp(x: Extended): Extended;
792: Func ExpandEnvironmentVar( var Value :Str) :Bool
793: Func ExpandFileName( FileName :Str) :Str
794: Func ExpandFileName(const FileName:Str):Str)
795: Func ExpandUNCFileName( FileName :Str) :Str
796: Func ExpandUNCFileName(const FileName:Str):Str)
797: Func ExpJ( const X : Float) : Float;
798: Func Exsecans( X : Float) : Float
799: Func Extract( const AByteCount : Int) :Str
800: Func Extract( Item : TClass) : TClass
801: Func Extract( Item : TComponent) : TComponent
802: Func Extract( Item : TObject) : TObject
803: Func ExtractFileDir( FileName :Str) :Str
804: Func ExtractFileDir(const FileName:Str):Str)
805: Func ExtractFileDrive( FileName :Str) :Str
806: Func ExtractFileDrive(const FileName:Str):Str)
807: Func ExtractFileExt( FileName :Str) :Str
808: Func ExtractFileExt(const FileName:Str):Str)
809: Func ExtractFileExtNoDot( const FileName :Str) :Str
810: Func ExtractFileExtNoDotUpper( const FileName :Str) :Str
811: Func ExtractFileName( FileName :Str) :Str
812: Func ExtractFileName(const filename:Str):str;
813: Func ExtractFilePath( FileName :Str) :Str
814: Func ExtractFilePath(const filename:Str):str;
815: Func ExtractRelativePath( BaseName, DestName :Str) :Str
816: Func ExtractRelativePath(const BaseName:Str; const DestName:Str):Str)
817: Func ExtractShortPathName( FileName :Str) :Str
818: Func ExtractShortPathName(const FileName:Str):Str)
819: Func ExtractStrings(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strings:TStrings):Int
820: Func ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings):Int)
821: Func Fact(numb: Int): Extended;
822: Func FactInt(numb: Int): int64;
823: Func Factorial( const N : Int) : Extended
824: Func FahrenheitToCelsius( const AValue : Double) : Double
825: Func FalseBoolStrs: array of string
826: Func Fetch(var AInput:str;const ADelim:str;const ADelete:Bool;const ACaseSensitive:Bool):str
827: Func FetchCaseInsensitive(var AInput:str;const ADelim:str;const ADelete:Bool):str
828: Func Fibo(numb: Int): Extended;
829: Func FiboInt(numb: Int): Int64;
830: Func Fibonacci( const N : Int) : Int
831: Func FIELDBYNAME(const FIELDNAME :Str) : TFIELD
832: Func FIELDBYNAME(const FIELDNAME : WIDESTRING) : TFIELD
833: Func FIELDBYNAME(const NAME :Str) : TFIELD
834: Func FIELDBYNAME(const NAME :Str) : TFIELDDEF
835: Func FIELDBYNUMBER( FIELDNO : Int) : TFIELD
836: Func FileAge( FileName :Str) : Int
837: Func FileAge(const FileName:Str): Int)
838: Func FileCompareText( const A, B :Str) : Int
839: Func FileContains(const FileName:str;Text:str;CaseSensitive:Bool;ExceptionOnError:Bool):Bool
840: Func FileCreate(FileName:Str): Int;
841: //FileCreate2(FileName:str;Rights:Int):Int;
842: Func FileCreate(const FileName:Str): Int)
843: Func FileCreateTemp( var Prefix :Str) : THandle
844: Func FileDateToDateTime( FileDate : Int) : TDateTime
845: Func FileDateToDateTime(FileDate: Int): TDateTime;
846: Func FileExists( const FileName :Str) :Bool
847: Func FileExists( FileName :Str) :Bool
848: Func fileExists(const FileName:Str):Bool;
849: Func FileGetAttr( FileName :Str) : Int
850: Func FileGetAttr(const FileName:Str): Int)
851: Func FileGetDate( Handle : Int) : Int
852: Func FileGetDate(handle: Int): Int
853: Func FileGetDisplayName( const FileName :Str) :Str
854: Func FileGetSize( const FileName :Str) : Int
855: Func FileGetTempName( const Prefix :Str) :Str
856: Func FileGetTypeName( const FileName :Str) :Str
857: Func FileIsReadOnly( FileName :Str) :Bool
858: Func FileLoad( ResType:TResType;const Name:str;MaskColor:TColor):Boolean
859: Func FileOpen( FileName :Str; Mode : LongWord) : Int
860: Func FileOpen(const FileName:Str; mode:Int): Int)
861: Func FileRead(handle: Int; Buffer: PChar; count: LongWord): Int
862: Func FileSearch( Name, DirList :Str) :Str
863: Func FileSearch(const Name, dirList:Str):Str)
864: Func FileSeek( Handle : Int; Offset : Int64; Origin : Int) : Int64;
865: Func FileSeek( Handle, Offset, Origin : Int) : Int;
866: Func FileSeek(handle, offset, origin: Int): Int
867: Func FileSetAttr( FileName :Str; Attr : Int) : Int
868: Func FileSetAttr(const FileName:Str; Attr: Int): Int)
869: Func FileSetDate(FileName :Str; Age : Int) : Int;
```

```
870: Func FileSetDate(handle: Int; age: Int): Int
871: Func FileSetDate2(FileHandle : Int; Age : Int) : Int;
872: Func FileSetDateH( Handle : Int; Age : Int) : Int;
873: Func FileSetReadOnly( FileName :Str; ReadOnly :Bool) :Bool
874: Func FileSize( const FileName :Str) : int64
875: Func FileSizeByName( const AFilename :Str) : Longint
876: Func FileWrite(Handle: Int; const Buffer: pChar; Count: LongWord): Int)
877: Func FilterSpecArray : TComdlgFilterSpecArray
878: Func FIND(ACAPTION :Str) : TMENUITEM
879: Func Find(AItem : Pointer; out AData : Pointer) :Bool
880: Func FIND(const ANAME :Str) : TNAMEDITEM
881: Func Find(const DisplayName :Str) : TAggregate
882: Func Find(const Item : TBookmarkStr; var Index : Int) :Bool
883: Func FIND(const NAME :Str) : TFIELD
884: Func FIND(const NAME :Str) : TFIELDDEF
885: Func FIND(const NAME :Str) : TINDEXDEF
886: Func Find(const S : WideString; var Index : Int) :Bool
887: Func Find(S:str;var Index:Int):Boolean
888: Func FindAuthClass( AuthName :Str) : TIdAuthenticationClass
889: Func FindBand( AControl : TControl) : TCoolBand
890: Func FindBoundary( AContentType :Str) :Str
891: Func FindButton( AModalResult : TModalResult) : TTaskDialogBaseButtonItem
892: Func FindCaption(StartIndex:Int;Value:str;Partial,Inclusive,Wrap:Boolean):TListItem
893: Func FindCdLineSwitch( Switch :Str; IgnoreCase :Bool) :Bool;
894: Func FindCloseW(FindFile: Int): LongBool; stdcall;
895: Func FindCmdLineSwitch(Switch:str;Chars:TSysCharSet;IgnoreCase:Boolean):Bool;
896: Func FindCmmdLineSwitch( Switch :Str) :Bool;
897: Func FindComponent(AName:Str): TComponent;
898: Func FindComponent(vlabel:Str): TComponent;
899: Func FindComponent2(vlabel:Str): TComponent;
900: Func FindControl(Handle: HWnd): TWinControl;
901: Func FindData( StartIndex : Int; Value : Pointer; Inclusive, Wrap :Bool):TListItem
902: Func FindDatabase( const DatabaseName :Str) : TDatabase
903: Func FindDragTarget(const Pos: TPoint; AllowDisabled:Bool): TControl;
904: Func FINDFIELD( const FIELDNAME :Str) : TFIELD
905: Func FINDFIELD( const FIELDNAME : WideString) : TFIELD
906: Func FindFirst2(const Path:Str; Attr: Int; var F: TSearchRec):Int)
907: Func FindNext2(var F: TSearchRec): Int)
908: Proc FindClose2(var F: TSearchRec)
909: Func FINDFIRST :Bool
910: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
911:   sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms,sfRecent,sfSendTo,
     sfStartMenu, stStartUp, sfTemplates);
912:  FFolder: array [TJvSpecialFolder] of Int =
913:     (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
914:      CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
915:      CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
916:      CSIDL_STARTUP, CSIDL_TEMPLATES);
917: Func FindFilesDlg(StartIn:str;SpecialFolder:TJvSpecialFolder;UseFolder:Bool):Bool);
918: Func Findfirst(const filepath:Str; attr: Int): Int;
919: Func FindFirst2(const Path:Str; Attr: Int; var F: TSearchRec): Int)
920: Func FindFirstNotOf( AFind, AText :Str) : Int
921: Func FindFirstOf( AFind, AText :Str) : Int
922: Func FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
923: Func FINDINDEXFORFIELDS( const FIELDS :Str) : TINDEXDEF
924: Func FindInstanceOf( AClass : TClass; AExact :Bool; AStartAt : Int) : Int
925: Func FINDITEM( VALUE : Int; KIND : TFINDITEMKIND) : TMENUITEM
926: Func FindItemId( Id : Int) : TCollectionItem
927: Func FindKey( const KeyValues : array of const) :Bool
928: Func FINDLAST:BOOLEAN
929: Func FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
930: Func FindModuleClass( AClass : TComponentClass) : TComponent
931: Func FindModuleName( const AClass :Str) : TComponent
932: Func FINDNEXT :Bool
933: Func FindNext: Int;
934: Func FindNext2(var F: TSearchRec): Int)
935: Func FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible :Bool):TTabSheet
936: Func FindNextToSelect : TTreeNode
937: Func FINDPARAM( const VALUE :Str) : TPARAM
938: Func FindParam( const Value : WideString) : TParameter
939: Func FINDPRIOR :Bool
940: Func FindResource(ModuleHandle:HMODULE;ResourceName,ResourceType:PChar):TResourceHandle
941: Func FindSession( const SessionName :Str) : TSession
942: Func FindStringResource(Ident: Int):Str)
943: Func FindText(const SearchStr:str;StartPos,Length:Int;Options:TSearchTypes):Int
944: Func FindUnusedFileName( const FileName, FileExt, Suffix : Ansistr) : Ansistr
945: Func FindVCLWindow(const Pos: TPoint): TWinControl;
946: Func FindWindow(C1, C2: PChar): Longint;
947: Func FindInPaths(const fileName,paths:Str):Str;
948: Func Finger :Str
949: Func First: TClass
950: Func First: TComponent
951: Func First: TObject
952: Func FirstDelimiter( const delimiters :Str; const Str :Str) : Int;
953: Func FirstDelimiter1(const delimiters:WideString; const Str : WideString) : Int;
954: Func FirstInstance( const ATitle :Str) :Bool
955: Func FloatPoint( const X, Y : Float) : TFloatPoint;
956: Func FloatPoint1( const P : TPoint) : TFloatPoint;
957: Func FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) :Bool
```

```
 958: Func FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
 959: Func FloatRect1( const Rect : TRect) : TFloatRect;
 960: Func FloatsEqual( const X, Y : Float) :Bool
 961: Func FloatToBin(const D: Double):Str;      //doubletohex -> hextobin! in buffer
 962: Func FloatToCurr( Value : Extended) : Currency
 963: Func FloatToDateTime( Value : Extended) : TDateTime
 964: Func FloatToStr( Value : Extended) :Str;
 965: Func FloatToStr(e : Extended) :Str;
 966: Func FloatToStrF(Value:Extended;Format:TFloatFormat;Precision,Digits:Int):str;
 967: Func FloatToStrF(Value:Extended;Format:TFloatFormat;Precision:Int;Digits:Int):Str)
 968: Func FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Int; FormatSettings :
      TFormatSettings) :Str;
 969: Func FloatToStrFS(Value:Extended;Format:TFloatFormat;Precision,
      Digits:Int;FormatSettings:TFormatSettings):str;
 970: Func FloatToText(BufferArg:PChar;const Value:Extended;ValueType:TFloatValue;Format:TFloatFormat;Precision,
      Digits: Int):Int)
 971: Func Floor( const X : Extended) : Int
 972: Func FloorInt( Value : Int; StepSize : Int) : Int
 973: Func FloorJ( const X : Extended) : Int
 974: Func Flush( const Count :Card) :Bool
 975: Func Flush(var t: Text): Int
 976: Func FmtLoadStr(Ident: Int; const Args: array of const):Str)
 977: Func FOCUSED:BOOLEAN
 978: Func ForceBackslash( const PathName :Str) :Str
 979: Func ForceDirectories( const Dir :Str) :Bool
 980: Func ForceDirectories( Dir :Str) :Bool
 981: Func ForceDirectories( Name :Str) :Bool
 982: Func ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
 983: Func ForceInRange( A, Min, Max : Int) : Int
 984: Func ForceInRangeR( const A, Min, Max : Double) : Double
 985: Func ForEach( AProc : TBucketProc; AInfo : Pointer) :Bool;
 986: Func ForEach1( AEvent : TBucketEvent) :Bool;
 987: Func ForegroundTask:Bool
 988: Func Format(const Format:Str; const Args: array of const):Str;
 989: Func FormatBcd( const Format :Str; Bcd : TBcd) :Str
 990: Func FormatBigInt(s:Str):Str;
 991: Func FormatByteSize(const bytes: int64):Str;
 992: Func FormatBuf(var Buffer:PChar;BufLen:Card;const Format:str;FmtLen:Card;const Args:array of const):Card
 993: Func FormatCount(iCount:Int;const sSingular:str;const sPlural:str):str
 994: Func FormatCurr( Format :Str; Value : Currency) :Str;
 995: Func FormatCurr(const Format:Str; Value: Currency):Str)
 996: Func FormatDateTime( Format :Str; DateTime : TDateTime) :Str;
 997: Func FormatDateTime(const fmt:Str; D: TDateTime):Str;
 998: Func FormatDateTime2(const Formatting:str;DateTime:TDateTime;const FormatSettings: TFormatSettings):str;
 999: Func FormatFloat( Format :Str; Value : Extended) :Str;
1000: Func FormatFloat(const Format:Str; Value: Extended):Str)
1001: Func FormatFloat( Format :Str; Value : Extended) :Str;
1002: Func FormatFloat2(Format:str;Value:Extended;FormatSettings:TFormatSettings):str;
1003: Func FormatF('%.2f - %.12f ',[notes[it],(ln(notes[it+1]/notes[it])*1200)/ln(2)]);  //with println - writeln
1004: Func FormatCurr( Format :Str; Value : Currency) :Str;
1005: Func FormatCurr2(Format:str;Value:Currency;FormatSettings:TFormatSettings):str;
1006: Func Format2(const Frmat:str;const Args:array of const;const FSettings:TFormatSettings):str
1007: Func FormatInt(i: Int):Str;
1008: Func FormatInt64(i: int64):Str;
1009: Func FormatMaskText( const EditMask :Str; const Value :Str) :Str
1010: Func FormatValue( AValue :Card) :Str
1011: Func FormatVersionString( const HiV, LoV : Word) :Str;
1012: Func FormatVersionString1( const Major, Minor, Build, Revision : Word) :Str;
1013: Func Frac(X: Extended): Extended);
1014: Func FreeResource( ResData : HGLOBAL) : LongBool
1015: Func FromCommon( const AValue : Double) : Double
1016: Func FromCommon(const AValue: Double): Double;
1017: Func FTPGMTDateTimeToMLS( const ATimeStamp:TDateTime; const AIncludeMSecs:Boolean):str
1018: Func FTPLocalDateTimeToMLS(const ATimeStamp:TDateTime;const AIncludeMSecs:Boolean):str
1019: Func FTPMLSToGMTDateTime( const ATimeStamp :Str) : TDateTime
1020: Func FTPMLSToLocalDateTime( const ATimeStamp :Str) : TDateTime
1021: Func FuncIn(AValue: Variant; ASet: Variant):Bool;
1022: //Func Funclist Size is: 6444 of mX3.9.8.9
1023: Func FutureValue(const Rate:Extended;NPeriods:Int;const Payment,
      PresentValue:Extended;PaymentTime:TPaymentTime):Extended
1024: Func FullTimeToStr(SUMTime: TDateTime):Str;;
1025: Func Gauss( const x, Spread : Double) : Double
1026: Func Gauss(const x,Spread: Double): Double;
1027: Func GCD(x, y : LongInt) : LongInt;
1028: Func GCDJ( X, Y :Card) :Card
1029: Func GDAL: LongWord
1030: Func GdiFlush : BOOL
1031: Func GdiSetBatchLimit( Limit : DWORD) : DWORD
1032: Func GdiGetBatchLimit : DWORD
1033: Func GenerateHeader : TIdHeaderList
1034: Func GeometricMean( const X : TDynFloatArray) : Float
1035: Func Get( AURL :Str) :Str;
1036: Func Get2( AURL :Str) :Str;
1037: Func Get8087CW : Word
1038: Func GetActiveOleObject(const ClassName:Str): IDispatch;
1039: Func GetAliasDriverName( const AliasName :Str) :Str
1040: Func GetAPMBatteryFlag : TAPMBatteryFlag
1041: Func GetAPMBatteryFullLifeTime : DWORD
1042: Func GetAPMBatteryLifePercent : Int
```

```
1043: Func GetAPMBatteryLifeTime : DWORD
1044: Func GetAPMLineStatus : TAPMLineStatus
1045: Func GetAppdataFolder :Str
1046: Func GetAppDispatcher : TComponent
1047: Func GetArrayLength: Int;
1048: Func GetASCII:Str;
1049: Func GetASCIILine:Str;
1050: Func GetAsHandle( Format : Word) : THandle
1051: Func GetAssociatedProgram(const Extension:str;var Filename,Description:str):bool;
1052: Func GetBackupFileName( const FileName :Str) :Str
1053: Func GetBaseAddress(PID:DWORD):DWORD;  //Process API
1054: Func GetBBitmap( Value : TBitmap) : TBitmap
1055: Func GetBIOSCopyright :Str
1056: Func GetBIOSDate : TDateTime
1057: Func GetBIOSExtendedInfo :Str
1058: Func GetBIOSName :Str
1059: Func getBitmap(apath:Str): TBitmap;
1060: Func GetBitmap( Index : Int; Image : TBitmap) :Bool  //object
1061: Func getBitMapObject(const bitmappath:Str): TBitmap;
1062: Func GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1063: Func GetCapsLockKeyState :Bool
1064: Func GetCaptureControl: TControl;
1065: Func GetCDAudioTrackList(var TrackList:TJclCdTrackInfoArray;Drive:Char):TJclCdTrackInfo;
1066: Func GetCDAudioTrackList1(TrackList:TStrings;IncludeTrackType:Boolean;Drive:Char):str;
1067: Func GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) :Str
1068: Func GetChangedText( const Text:str; SelStart, SelLength : Int; Key : Char):str
1069: Func GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1070: Func GetClockValue : Int64
1071: Func getCmdLine: PChar;
1072: Func getCmdShow: Int;
1073: Func GetCPUSpeed: Double;
1074: Func GetColField( DataCol : Int) : TField
1075: Func GetColorBlue( const Color : TColor) : Byte
1076: Func GetColorFlag( const Color : TColor) : Byte
1077: Func GetColorGreen( const Color : TColor) : Byte
1078: Func GetColorRed( const Color : TColor) : Byte
1079: Func GetComCtlVersion : Int
1080: Func GetComPorts: TStringlist;
1081: Func GetCommonAppdataFolder :Str
1082: Func GetCommonDesktopdirectoryFolder:Str
1083: Func GetCommonFavoritesFolder :Str
1084: Func GetCommonFilesFolder :Str
1085: Func GetCommonProgramsFolder :Str
1086: Func GetCommonStartmenuFolder :Str
1087: Func GetCommonStartupFolder :Str
1088: Func GetComponent( Owner, Parent : TComponent) : TComponent
1089: Func GetConnectionRegistryFile( DesignMode :Bool):Str
1090: Func GetCookiesFolder :Str
1091: Func GetCPUSpeed( var CpuSpeed : TFreqInfo) :Bool
1092: Func GetCurrent:TFavoriteLinkItem
1093: Func GetCurrent:TListItem
1094: Func GetCurrent:TTaskDialogBaseButtonItem
1095: Func GetCurrent:TToolButton
1096: Func GetCurrent:TTreeNode
1097: Func GetCurrent:WideString
1098: Func GetCurrentDir :Str
1099: Func GetCurrentDir:Str)
1100: Func GetCurrentFolder :Str
1101: Func GETCURRENTRECORD( BUFFER : PCHAR) :Bool
1102: Func GetCurrentProcessId : TIdPID
1103: Func GetCurrentThreadHandle : THandle
1104: Func GetCurrentThreadID: LongWord; stdcall;
1105: Func GetCustomHeader( const Name :Str) :Str
1106: Func GetDataItem( Value : Pointer) : Longint
1107: Func GetDataLinkFiles(FileNames:TWideStrings; Directory string): Int;
1108: Func GetDataLinkFiles1( FileNames : TStrings; Directory:Str):Int;
1109: Func GETDATASIZE : Int
1110: Func GetDC(hdwnd: HWND): HDC;
1111: Func GetDefaultFileExt( const MIMEType :Str) :Str
1112: Func GetDefaults :Bool
1113: Func GetDefaultSchemaName : WideString
1114: Func GetDefaultStreamLoader : IStreamLoader
1115: Func GetDesktopDirectoryFolder :Str
1116: Func GetDesktopFolder :Str
1117: Func GetDFAState( oStates : TList) : TniRegularExpressionState
1118: Func GetDirectorySize( const Path :Str) : Int64
1119: Func GetDisplayWidth : Int
1120: Func GetDLLVersion(const DLLName:str;var pdwMajor, pdwMinor:Int):Boolean
1121: Func GetDomainName :Str
1122: Func GetDriverRegistryFile( DesignMode :Bool) :Str
1123: Func GetDriveType(rootpath: pchar):Card;
1124: Func GetDriveTypeStr( const Drive : Char) :Str
1125: Func GetEnumerator : TFavoriteLinkItemsEnumerator
1126: Func GetEnumerator : TListItemsEnumerator
1127: Func GetEnumerator : TTaskDialogButtonsEnumerator
1128: Func GetEnumerator : TToolBarEnumerator
1129: Func GetEnumerator : TTreeNodesEnumerator
1130: Func GetEnumerator : TWideStringsEnumerator
1131: Func GetEnvVar( const VarName :Str) :Str
```

```
Func GetEnvironmentVar( const AVariableName :Str) :Str
Func GetEnvironmentVariable( const VarName :Str) :Str
Func GetEnvironmentVar(const Name string; var Value:str;Expand:Boolean):Bool
Func GetEnvironmentVars( const Vars : TStrings; Expand :Bool):Boolean
Func getEnvironmentString:Str;
Func GetExceptionHandler : TObject
Func GetFavoritesFolder :Str
Func GetFieldByName( const Name :Str) :Str
Func GetFieldInfo(const Origin: Widestring;var FieldInfo TFieldInfo):Boolean
Func GetFieldValue( ACol : Int) :Str
Func GetFileAgeCoherence( const FileName :Str) :Bool
Func GetFileCreation( const FileName :Str) : TFileTime
Func GetFileCreationTime( const Filename :Str) : TDateTime
Func GetFileInformation( const FileName :Str) : TSearchRec
Func GetFileLastAccess( const FileName :Str) : TFileTime
Func GetFileLastWrite( const FileName :Str) : TFileTime
Func GetFileList(FileList: TStringlist; apath:Str): TStringlist;
Func GetFileList1(apath:Str): TStringlist;
Func GetFileMIMEType( const AFileName :Str) :Str
Func GetFileSize( const FileName :Str) : Int64
Func GetFileVersion( AFileName :Str) :Card
Func GetFileVersion( const AFilename :Str) :Card
Func GetFileSize2(Handle: Int; x: Int): Int; stdcall;
Func GetFileDate(aFile:str; aWithTime:Boolean):str;
Func GetFileCount(adirmask:Str): Int;  //files count in directory!
Func GetFilterData( Root : PExprNode) : TExprData
Func getFirstChild : LongInt
Func getFirstChild : TTreeNode
Func GetFirstDelimitedToken( const cDelim:char;const cStr:Str):str
Func GetFirstNode : TTreeNode
Func GetFontsFolder :Str
Func GetFormulaValue( const Formula :Str) : Extended
Func GetFreePageFileMemory : Int
Func GetFreePhysicalMemory : Int
Func GetFreeSystemResources( const ResourceType : TFreeSysResKind): Int;
Func GetFreeSystemResources1 : TFreeSystemResources;
Func GetFreeVirtualMemory : Int
Func GetFromClipboard :Bool
Func GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet):Str
Func GetGBitmap( Value : TBitmap) : TBitmap
Func GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
Func GetGroupState( Level : Int) : TGroupPosInds
Func GetHandle : HWND
Func GETHELPCONTEXT( VALUE : Int; BYCOMMAND :Bool) : THELPCONTEXT
Func GetHexArray(ahexdig: THexArray): THexArray;
Func GetHighLightColor( const Color : TColor; Luminance : Int):TColor
Func GetHINSTANCE: longword;
Func GetHistoryFolder :Str
Func GetHitTestInfoAt( X, Y : Int) : THitTests
Func getHMODULE: longword;
Func GetHostByName(const AComputerName:Str):str;
Func GetHostName :Str
Func getHostIP:Str;
Func GetHotSpot : TPoint
Func GetHueBitmap( Value : TBitmap) : TBitmap
Func GetImageBitmap : HBITMAP
Func GETIMAGELIST : TCUSTOMIMAGELIST
Func GetIncome( const aNetto : Currency): Currency
Func GetIncome( const aNetto : Extended): Extended
Func GetIncome( const aNetto : Extended): Extended
Func GetIncome(const aNetto : Extended) : Extended
Func GetIncome(const aNetto: Currency): Currency
Func GetIncome2( const aNetto : Currency) : Currency
Func GetIncome2( const aNetto : Currency): Currency
Func getIndex_Attrs( tag :Str; var idx : Int; var Attrs :Str) :Str
Func GETINDEXFORFIELDS( const FIELDS :Str; CASEINSENSITIVE :Bool) : TINDEXDEF
Func GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet): TIndexDef
Func GetInstRes(Instance:THandle;ResType:TResType;const
Name:str;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
Func
GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
Func GetIntelCacheDescription( const D : Byte):Str
Func GetInteractiveUserName :Str
Func GetInternetCacheFolder :Str
Func GetInternetFormattedFileTimeStamp( const AFilename :Str) :Str
Func GetIPAddress( const HostName :Str) :Str
Func GetIP( const HostName :Str) :Str
Func GetIPHostByName(const AComputerName:Str):Str;
Func GetIsAdmin:Bool;
Func GetItem( X, Y : Int) : LongInt
Func GetItemAt( X, Y : Int) : TListItem
Func GetItemHeight(Font: TFont): Int;
Func GetItemPath( Index : Int) :Str
Func GetKeyFieldNames( List : TStrings) : Int;
Func GetKeyFieldNames1( List : TWideStrings) : Int;
Func GetKeyState( const VirtualKey :Card) :Bool
Func GetLastChild : LongInt
Func GetLastChild : TTreeNode
Func GetLastDelimitedToken(const cDelim:char;const cStr:str):Str
```

```
1219: Func GetLastError: Int
1220: Func GetLAT_CONV_FACTOR: double;  //for WGS84 power(1 - 1 / 298.257223563, 2);
1221: Func GetLinesCount(sFileName :Str): Int;
1222: Func GetLoader( Ext :Str) : TBitmapLoader
1223: Func GetLoadFilter :Str
1224: Func GetLocalComputerName :Str
1225: Func GetLocaleChar( Locale, LocaleType : Int; Default : Char) : Char
1226: Func GetLocaleStr( Locale, LocaleType : Int; Default :Str) :Str
1227: Func GetLocalUserName :Str
1228: Func GetLoginUsername : WideString
1229: Func getLongDayNames:Str)
1230: Func GetLongHint(const hint:Str):Str
1231: Func getLongMonthNames:Str)
1232: Func GetMacAddresses( const Machine :Str; const Addresses : TStrings) : Int
1233: Func GetMainAppWndFromPid( PID : DWORD) : HWND
1234: Func GetMapX(C_form,apath:Str; const Data:Str):Bool; //c_form: [html/json/xml]
1235: //if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne then
1236: Proc GetGEOMap(C_form,apath:Str; const Data:Str);
1237: Func GetMapXGeoReverse(C_form:Str; const lat,long:Str):Str;
1238: //if GetMapXGeoReverse('XML',topPath,'47.0397826','7.62914761277888 then
1239: Func GetGeoCode(C_form,apath:Str; const data:Str; sfile:bool):Str;
1240: Func GetMaskBitmap : HBITMAP
1241: Func GetMaxAppAddress : Int
1242: Func GetMciErrorMessage( const MciErrNo : MCIERROR) :Str
1243: Func GetMemoryLoad : Byte
1244: Func GetMIMEDefaultFileExt( const MIMEType :Str) : TIdFileName
1245: Func GetMIMETypeFromFile( const AFile :Str) :Str
1246: Func GetMIMETypeFromFile( const AFile : TIdFileName) :Str
1247: Func GetMinAppAddress : Int
1248: Func GetModule : TComponent
1249: Func GetModuleHandle( ModuleName : PChar) : HMODULE
1250: Func GetModuleName( Module : HMODULE) :Str
1251: Func GetModulePath( const Module : HMODULE) :Str
1252: Func GetModuleFileName(Module: Int; Filename: PChar;Size: Int): Int; stdcall;
1253: Func GetMorseID(InChar : Char): Word;;
1254: Func GetMorseString2(InChar : Char):Str;;
1255: Func GetMorseLine(dots:Bool):Str;;  //whole table!  {1 or dots}
1256: Func GetMorseTable(dots:Bool):Str;;  //whole table!
1257: Func GetMorseSign(InChar : Char):Str;;
1258: Func GetCommandLine: PChar; stdcall;
1259: Func GetMonochromeBitmap( Value : TBitmap) : TBitmap
1260: Func GetMultiN(aval: Int):Str;
1261: Func GetName :Str
1262: Func GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1263: Func GetNethoodFolder :Str
1264: Func GetNext : TTreeNode
1265: Func GetNextChild( Value : LongInt) : LongInt
1266: Func GetNextChild( Value : TTreeNode) : TTreeNode
1267: Func GetNextDelimitedToken( const cDelim : char; var cStr :Str) :Str
1268: Func GetNextItem(StartItem:TListItem;Direction:TSearchDirection;States:TItemStates):TListItem
1269: Func GetNextPacket : Int
1270: Func getNextSibling : TTreeNode
1271: Func GetNextVisible : TTreeNode
1272: Func GetNode( ItemId : HTreeItem) : TTreeNode
1273: Func GetNodeAt( X, Y : Int) : TTreeNode
1274: Func GetNodeDisplayWidth( Node : TOutlineNode) : Int
1275: Func GetNumberOfProcessors: longint;
1276: Func GetNumLockKeyState :Bool
1277: Func GetObjectProperty( Instance : TPersistent; const PropName :Str) : TObject
1278: Func GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1279: Func GetOptionalParam( const ParamName :Str) : OleVariant
1280: Func GetOSName:Str;
1281: Func GetOSVersion:Str;
1282: Func GetOSNumber:Str;
1283: Func GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
1284: Func GetPackageModuleHandle( PackageName : PChar) : HMODULE
1285: Func GetPageSize:Card;
1286: Func GetParameterFileName :Str
1287: Func GetParams( var OwnerData : OleVariant) : OleVariant
1288: Func GETPARENTCOMPONENT : TCOMPONENT
1289: Func GetParentForm(control: TControl): TForm
1290: Func GETPARENTMENU : TMENU
1291: Func GetPassword:Bool
1292: Func GetPassword:Str
1293: Func GetPersonalFolder :Str
1294: Func GetPidFromProcessName( const ProcessName :Str) : DWORD
1295: Func getPI: extended;        //of const PI math
1296: Func GetPixel(dc:HDC; PixelCoords.X: TPoint, PixelCoords.Y: TPoint);
1297: Func GetPosition : TPoint
1298: Func GetPrev : TTreeNode
1299: Func GetPrevChild( Value : LongInt) : LongInt
1300: Func GetPrevChild( Value : TTreeNode) : TTreeNode
1301: Func getPrevSibling : TTreeNode
1302: Func GetPrevVisible : TTreeNode
1303: Func GetPrinthoodFolder :Str
1304: Func GetPrivilegeDisplayName( const PrivilegeName:Str):Str
1305: Func getProcessList: TStrings;
1306: Func GetProcessId : TIdPID
1307: Func GetProcessNameFromPid( PID : DWORD) :Str
```

```
1308: Func GetProcessNameFromWnd( Wnd : HWND) :Str
1309: Func GetProcessMemoryInfo(Process:THandle;ppsmemCounters:TProcessMemoryCounters;cb:DWORD):BOOL
1310: Func getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1311: Func getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1312: Func GetProgramFilesFolder :Str
1313: Func GetProgramsFolder :Str
1314: Func GetProxy :Str
1315: Func GetQuoteChar : WideString
1316: Func GetQrCode4(Width,Height:Word; Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
1317: Func GetQrCodetoFile(Width,Height:Word;Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
1318: Func GetRate : Double
1319: Func getPerfTime:Str;
1320: Func getRuntime:Str;
1321: Func GetRBitmap( Value : TBitmap) : TBitmap
1322: Func GetReadableName( const AName :Str) :Str
1323: Func GetRecentDocs : TStringList
1324: Func GetRecentFolder :Str
1325: Func GetRecords( Count : Int; out RecsOut : Int; Options : Int) : OleVariant;
1326: Func GetRecords1(Count:Int;out RecsOut:Int;Options:Int;const CommandText:WideString;var Params,
      OwnerData:OleVariant):OleVariant;
1327: Func GetRecordset(const CommandText: WideString;ConnectionString:WideString):_Recordset
1328: Func GetRegisteredCompany :Str
1329: Func GetRegisteredOwner :Str
1330: Func GetResource(ResType:TResType;const Name:str;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor:Bool
1331: Func GetResourceName( ObjStream : TStream; var AName :Str) :Bool
1332: Func GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1333: Func GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1334: Func GetRValue(rgb: DWORD): Byte
1335: Func GetGValue(rgb: DWORD): Byte
1336: Func GetBValue(rgb: DWORD): Byte
1337: Func GetCValue(cmyk: COLORREF): Byte
1338: Func GetMValue(cmyk: COLORREF): Byte
1339: Func GetYValue(cmyk: COLORREF): Byte
1340: Func GetKValue(cmyk: COLORREF): Byte
1341: Func CMYK( c, m, y, k : Byte) : COLORREF
1342: Proc GetScreenShot(var ABitmap : TBitmap);
1343: Func GetOSName:Str;
1344: Func GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR):FARPROC
1345: Func GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1346: Func GetSafeCallExceptionMsg :Str
1347: Func GetSaturationBitmap( Value : TBitmap) : TBitmap
1348: Func GetSaveFilter :Str
1349: Func GetSaver( Ext :Str) : TBitmapLoader
1350: Func GetScrollLockKeyState :Bool
1351: Func GetSearchString :Str
1352: Func GetSelections( AList : TList) : TTreeNode
1353: Func GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:Int):Int
1354: Func GetSendToFolder :Str
1355: Func GetServer : IAppServer
1356: Func GetServerList : OleVariant
1357: Func GetShadowColor( const Color : TColor; Luminance:Int): TColor
1358: Func GetShellProcessHandle : THandle
1359: Func GetShellProcessName :Str
1360: Func GetShellVersion :Card
1361: Func getShortDayNames:Str)
1362: Func GetShortHint(const hint:Str):Str
1363: Func getShortMonthNames:Str)
1364: Func GetSizeOfFile( const FileName :Str) : Int64;
1365: Func GetSizeOfFile1( Handle : THandle) : Int64;
1366: Func GetStdHandle(nStdHandle: Int): Int; stdcall;
1367: Func GetStartmenuFolder :Str
1368: Func GetStartupFolder :Str
1369: Func GetStringProperty( Instance: TPersistent; const PropName:str):WideString
1370: Func GetSuccessor( cChar : char): TniRegularExpressionState
1371: Func GetSwapFileSize : Int
1372: Func GetSwapFileUsage : Int
1373: Func GetSystemLocale : TIdCharSet
1374: Func GetSystemMetrics( nIndex : Int) : Int
1375: Func GetSystemPathSH(Folder: Int): TFilename ;
1376: Func GetTableNameFromQuery( const SQL : Widestring) : Widestring
1377: Func GetTableNameFromSQL( const SQL : WideString) : WideString
1378: Func GetTableNameFromSQLEx(const SQL:WideString; IdOption:IDENTIFIEROption):WideString
1379: Func GetTasksList( const List : TStrings) :Bool
1380: Func getTeamViewerID:Str;
1381: Func GetTemplatesFolder :Str
1382: Func GetText : PwideChar
1383: Func GetText:PChar
1384: Func GetTextBuf( Buffer : PChar; BufSize : Int) : Int
1385: Func GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:Int):Int
1386: Func GetTextItem( const Value :Str) : Longint
1387: Func GETTEXTLEN:Int
1388: Func GetThreadLocale: Longint; stdcall
1389: Func GetCurrentThreadID: LongWord; stdcall;
1390: Func GetTickCount :Card
1391: Func GetTickDiff( const AOldTickCount, ANewTickCount :Card) :Card
1392: Func GetTicketNr : longint
1393: Func GetTime :Card
1394: Func GetTime : TDateTime
1395: Func GetTimeout : Int
```

```
1396: Func GetTimeStr:Str
1397: Func GetTimeString:Str
1398: Func GetTimeZoneInformation(var lpTimeZoneInformation: TTimeZoneInformation):DWORD
1399:   Ex: if GetTimeZoneInformation(tzi) >0 then writeln(itoa(tzi.bias));
1400: Func GetTodayFiles(startdir, amask:Str): TStringlist;
1401: Func getTokenCounts : Int
1402: Func GetTotalPageFileMemory : Int
1403: Func GetTotalPhysicalMemory : Int
1404: Func GetTotalVirtualMemory : Int
1405: Func GetUniqueFileName( const APath, APrefix, AExt :Str) :Str
1406: Func GetUseNowForDate :Bool
1407: Func GetUserDomainName( const CurUser :Str) :Str
1408: Func GetUserName :Str
1409: Func GetUserName:Str;
1410: Func GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD):BOOL;
1411: Func GetUserObjectName( hUserObject : THandle) :Str
1412: Func GetValueBitmap( Value : TBitmap) : TBitmap
1413: Func GetValueMSec :Card
1414: Func GetValueStr :Str
1415: Func GetVersion: int;
1416: Func GetVersionString(FileName:Str):Str;
1417: Func getVideoDrivers:Str;
1418: Func GetVisibleNode( Index : LongInt) : TOutlineNode
1419: Func GetVolumeFileSystem( const Drive :Str) :Str
1420: Func GetVolumeName( const Drive :Str) :Str
1421: Func GetVolumeSerialNumber( const Drive :Str) :Str
1422: Func GetWebAppServices : IWebAppServices
1423: Func GetWebRequestHandler : IWebRequestHandler
1424: Func GetWindowCaption( Wnd : HWND) :Str
1425: Func GetWindowDC(hdwnd: HWND): HDC;
1426: Func GetWindowIcon( Wnd : HWND; LargeIcon :Bool) : HICON
1427: Func GetWindowRect(hwnd: HWND; arect: TRect):Bool
1428: Func GetWindowsComputerID :Str
1429: Func GetWindowsDirectory(lpBuffer: PChar; uSize: longword):longword;
1430: Func GetWindowsFolder :Str
1431: Func GetWindowsServicePackVersion : Int
1432: Func GetWindowsServicePackVersionString :Str
1433: Func GetWindowsSystemFolder :Str
1434: Func GetWindowsTempFolder :Str
1435: Func GetWindowsUserID :Str
1436: Func GetWindowsVersion : TWindowsVersion
1437: Func GetWindowsVersionString :Str
1438: Func GmtOffsetStrToDateTime( S :Str) : TDateTime
1439: Func GMTToLocalDateTime( S :Str) : TDateTime
1440: Func GotoKey :Bool
1441: Func GradToCycle( const Grads : Extended) : Extended
1442: Func GradToDeg( const Grads : Extended) : Extended
1443: Func GradToDeg( const Value : Extended) : Extended;
1444: Func GradToDeg1( const Value : Double) : Double;
1445: Func GradToDeg2( const Value : Single) : Single;
1446: Func GradToRad( const Grads : Extended) : Extended
1447: Func GradToRad( const Value : Extended) : Extended;
1448: Func GradToRad1( const Value : Double) : Double;
1449: Func GradToRad2( const Value : Single) : Single;
1450: Func Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1451: Func GreenComponent( const Color32 : TColor32) : Int
1452: Func GUIDToString(const GUID: TGUID):Str)
1453: Func HandleAllocated :Bool
1454: Func HandleAllocated:Bool;
1455: Func HandleRequest :Bool
1456: Func HandleRequest( Request : TWebRequest; Response : TWebResponse:Boolean
1457: Func HarmonicMean( const X : TDynFloatArray) : Float
1458: Func HasAsParent( Value : TTreeNode) :Bool
1459: Func HASCHILDDEFS :Bool
1460: Func HasCurValues :Bool
1461: Func HasExtendCharacter( const s : UTF8String) :Bool
1462: Func HasFormat( Format : Word) :Bool
1463: Func HashValue( AStream : TStream) : T5x4LongWordRecord;
1464: Func HashValue(AStream : TStream) : T4x4LongWordRecord
1465: Func HashValue(AStream: TStream): LongWord
1466: Func HashValue(AStream: TStream): Word
1467: Func HashValue1(AStream TStream; const ABeginPos,AEndPos:Int64):T5x4LongWordRecord;
1468: Func HashValue1(AStream : TStream): T4x4LongWordRecord
1469: Func HashValue128(const ASrc:Str): T4x4LongWordRecord;
1470: Func HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1471: Func HashValue16( const ASrc :Str) : Word;
1472: Func HashValue16stream( AStream : TStream) : Word;
1473: Func HashValue32( const ASrc :Str) : LongWord;
1474: Func HashValue32Stream( AStream : TStream) : LongWord;
1475: Func HasMergeConflicts :Bool
1476: Func hasMoreTokens :Bool
1477: Func HASPARENT :Bool
1478: Func HasParent:Bool
1479: Func HasTransaction( Transaction : TDBXTransaction) :Bool
1480: Func HasUTF8BOM( S : TStream) :Bool;
1481: Func HasUTF8BOM1( S : Ansistr) :Bool;
1482: Func Haversine( X : Float) : Float
1483: Func Head( s :Str; const subs :Str; var tail :Str):str
1484: Func HELPCOMMAND(COMMAND:Int;DATA:LONGINT):BOOLEAN
```

```
1485: Func HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1486: Func HELPJUMP(JUMPID:str):BOOLEAN
1487: Func HeronianMean( const a, b : Float) : Float
1488: Func HexStrToStr(Value:Str):Str;
1489: Func HexToBin(Text,Buffer:PChar; BufSize:Int):Int;
1490: Func HexToBin2(HexNum:Str):Str;
1491: Func HexToDouble( const Hex :Str) : Double
1492: Func HexToInt(hexnum:Str): LongInt;
1493: Func HexToStr(Value:Str):Str;
1494: Func HexifyBlock( var Buffer, BufferSize : Int) :Str
1495: Func Hi(vdat: word): byte;
1496: Func HiByte(W: Word): Byte)
1497: Func High: Int64;
1498: Func HighlightCell(DataCol,DataRow:Int;const Value:str;AState:TGridDrawState):Bool
1499: Func HINSTANCE: longword;
1500: Func HiWord(l: DWORD): Word)
1501: Func HMODULE: longword;
1502: Func HourOf( const AValue : TDateTime) : Word
1503: Func HourOfTheDay( const AValue : TDateTime) : Word
1504: Func HourOfTheMonth( const AValue : TDateTime) : Word
1505: Func HourOfTheWeek( const AValue : TDateTime) : Word
1506: Func HourOfTheYear( const AValue : TDateTime) : Word
1507: Func HoursBetween( const ANow, AThen : TDateTime) : Int64
1508: Func HourSpan( const ANow, AThen : TDateTime) : Double
1509: Func HSLToRGB1( const H, S, L : Single) : TColor32;
1510: Func HTMLDecode( const AStr :Str) :Str
1511: Func HTMLEncode( const AStr :Str) :Str
1512: Func HTMLEscape( const Str :Str) :Str
1513: Func HtmlTable(DataSet:TDataSet;DataSetHandler:TDSTableProducer;MaxRows:Int):str
1514: Func HTTPDecode( const AStr :Str) :Str
1515: Func HTTPEncode( const AStr :Str) :Str
1516: Func Hypot( const X, Y : Extended) : Extended
1517: Func IBMax( n1, n2 : Int) : Int
1518: Func IBMin( n1, n2 : Int) : Int
1519: Func IBRandomString( iLength : Int) :Str
1520: Func IBRandomInt( iLow, iHigh : Int) : Int
1521: Func IBStripString( st :Str; CharsToStrip :Str) :Str
1522: Func IBFormatIdentifier( Dialect : Int; Value :Str) :Str
1523: Func IBFormatIdentifierValue( Dialect : Int; Value :Str) :Str
1524: Func IBExtractIdentifier( Dialect : Int; Value :Str) :Str
1525: Func IBQuoteIdentifier( Dialect : Int; Value :Str) :Str
1526: Func IBAddIBParamSQLForDetail(Params:TParams;SQL:str;Native:Boolean;Dialect:Int):str
1527: Proc IBDecomposeDatabaseName(DatabaseName:Str;var ServerName,Protocol,DatabasePath:str)
1528: Func RandomString( iLength : Int) :Str;
1529: Func RandomInt( iLow, iHigh : Int) : Int;
1530: Func StripString( st :Str; CharsToStrip :Str) :Str;
1531: Func Strip(const SubString:Str; MainString:Str):Str;
1532: Func StripTags(const S:Str):Str;   //<''> of HTML
1533: Func SizeToString(size : Int64; const unitStr :Str) :Str;
1534: Func NumbertoString(No: Word):Str;
1535: Func FormatIdentifier( Dialect : Int; Value :Str) :Str;
1536: Func FormatIdentifierValue( Dialect : Int; Value :Str) :Str;
1537: Func ExtractIdentifier( Dialect : Int; Value :Str) :Str;
1538: Func QuoteIdentifier( Dialect : Int; Value :Str) :Str;
1539: Func AddIBParamSQLForDetail(Params:TParams;SQL:str;Native:Boolean;Dialect:Int):str;
1540: Proc DecomposeDatabaseName(DatabaseName:str;var ServerName,Protocol,DatabasePath:str;
1541: Func NextSQLToken(var p: PChar; var Token:Str; CurSection: TSQLToken): TSQLToken;
1542: Func IconToBitmap( Ico : HICON) : TBitmap
1543: Func IconToBitmap2( Ico : HICON; Size : Int; TransparentColor : TColor) : TBitmap
1544: Func IconToBitmap3( Ico : HICON; Size : Int; TransparentColor : TColor) : TBitmap
1545: Func IdentToCharset(const Ident:Str; var Charset: Longint):Bool)
1546: Func IdentToColor(const Ident:Str; var Color: Longint):Bool)
1547: Func IdentToCursor(const Ident:Str; var cursor: Longint):Bool;
1548: Func IdGetDefaultCharSet : TIdCharSet
1549: Func IDispatchInvoke(Self:IDispatch;ProperSet:Bool;const Name:Str;Par:array of variant):variant
1550: Func IdPorts2 : TStringList
1551: Func IdToMib( const Id :Str) :Str
1552: Func IdSHA1Hash(apath:Str):Str;
1553: Func IdHashSHA1(apath:Str):Str;
1554: Func IfStr(const bCondition:boolean;const sTrue:str;const sFalse:str):Str
1555: Func IfThen( AValue :Bool; const ATrue :Str; AFalse :Str) :Str;
1556: Func IfThenInt( AValue :Bool; const ATrue : Int; AFalse : Int): Int;;
1557: Func IfThenDouble( AValue :Bool; const ATrue : double; AFalse : double): double;;
1558: Func IfThenBool( AValue :Bool; const ATrue :Bool; AFalse :Bool):Bool;;
1559: Func iif1( ATest :Bool; const ATrue : Int; const AFalse : Int) : Int;
1560: Func iif2( ATest :Bool; const ATrue :Str; const AFalse :Str) :Str;
1561: Func iif3( ATest:Boolean; const ATrue:Bool;const AFalse:Bool):Bool;
1562: Func ImportTest(S1:Str;s2:longint; s3:Byte; s4:word; var s5:str):Str;
1563: Func IncDay( const AValue : TDateTime; const ANumberOfDays : Int) : TDateTime
1564: Func IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1565: Func IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1566: Func IncLimit1(var B:Shortint;const Limit:Shortint;const Incr:Shortint):Shortint;
1567: Func IncLimit2(var B:Smallint;const Limit:Smallint;const Incr:Smallint):Smallint;
1568: Func IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1569: Func IncLimit4( var B : Int; const Limit : Int; const Incr : Int) : Int;
1570: Func IncLimit5(var B:Card;const Limit:Card;const Incr:Card):Card;
1571: Func IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1572: Func IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1573: Func IncLimitClamp1(var B: Shortint;const Limit:Shortint;const Incr: Shortint): Shortint;
```

```
1574: Func IncLimitClamp2(var B:Smallint;const Limit:Smallint;const Incr: Smallint): Smallint;
1575: Func IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1576: Func IncLimitClamp4( var B : Int; const Limit : Int; const Incr : Int) : Int;
1577: Func IncLimitClamp5(var B:Card;const Limit:Card; const Incr:Card):Card;
1578: Func IncLimitClamp6(var B: Int64;const Limit:Int64; const Incr:Int64):Int64;
1579: Func IncludeTrailingBackslash( S :Str) :Str
1580: Func IncludeTrailingBackslash(const S:Str):Str)
1581: Func IncludeTrailingPathDelimiter( const APath :Str) :Str
1582: Func IncludeTrailingPathDelimiter( S :Str) :Str
1583: Func IncludeTrailingPathDelimiter(const S:Str):Str)
1584: Func IncludeTrailingSlash( const APath :Str) :Str
1585: Func IncMilliSecond(const AValue:TDateTime;const ANumberOfMilliSeconds:Int64):TDateTime
1586: Func IncMinute(const AValue:TDateTime;const ANumberOfMinutes:Int64):TDateTime
1587: Func IncMonth( DateTime : TDateTime; NumberOfMonths : Int) : TDateTime
1588: Func IncMonth(const DateTime: TDateTime; NumberOfMonths: Int): TDateTime)
1589: Func IncSecond(const AValue:TDateTime;const ANumberOfSeconds: nt64) : TDateTime
1590: Func IncWeek(const AValue : TDateTime; const ANumberOfWeeks : Int) : TDateTime
1591: Func IncYear(const AValue : TDateTime; const ANumberOfYears : Int) : TDateTime
1592: Func IndexOf(AClass : TClass) : Int
1593: Func IndexOf(AComponent : TComponent) : Int
1594: Func IndexOf(AObject : TObject) : Int
1595: Func INDEXOF(const ANAME :Str) : Int
1596: Func IndexOf(const DisplayName :Str) : Int
1597: Func IndexOf(const Item : TBookmarkStr) : Int
1598: Func IndexOf(const S : WideString) : Int
1599: Func IndexOf(const View : TJclFileMappingView) : Int
1600: Func INDEXOF(FIELD : TFIELD) : Int
1601: Func IndexOf(ID : LCID) : Int
1602: Func INDEXOF(ITEM : TMENUITEM) : Int
1603: Func IndexOf(Value : TListItem) : Int
1604: Func IndexOf(Value : TTreeNode) : Int
1605: Func IndexOf(const S:Str): Int;
1606: Func IndexOfName( const Name : WideString) : Int
1607: Func IndexOfName(Name:Str): Int;
1608: Func IndexOfObject( AObject : TObject) : Int
1609: Func IndexofObject(AObject:tObject):Int
1610: Func IndexOfTabAt( X, Y : Int) : Int
1611: Func IndexStr( const AText :Str; const AValues : array of string) : Int
1612: Func IndexText( const AText :Str; const AValues : array of string): Int
1613: Func IndexOfInt( AList : TStringList; Value : Variant) : Int
1614: Func IndexOfFloat( AList : TStringList; Value : Variant) : Int
1615: Func IndexOfDate( AList : TStringList; Value : Variant) : Int
1616: Func IndexOfString( AList : TStringList; Value : Variant) : Int
1617: Func IndyCompareStr( const A1 :Str; const A2 :Str) : Int
1618: Func IndyGetHostName :Str
1619: Func IndyInterlockedDecrement( var I : Int) : Int
1620: Func IndyInterlockedExchange( var A : Int; B : Int) : Int
1621: Func IndyInterlockedExchangeAdd( var A : Int; B : Int) : Int
1622: Func IndyInterlockedIncrement( var I : Int) : Int
1623: Func IndyLowerCase( const A1 :Str) :Str
1624: Func IndyStrToBool( const AString :Str) :Bool
1625: Func IndyUpperCase( const A1 :Str) :Str
1626: Func InitCommonControl( CC : Int) :Bool
1627: Func InitTempPath :Str
1628: Func InMainThread :Bool
1629: Func inOpArray(W : WideChar; sets:array of WideChar):Bool
1630: Func Input: Text)
1631: Func InputBox( const ACaption, APrompt, ADefault :Str) :Str
1632: Func InputBox(const ACaption:str;const APrompt:str;const ADefault:str):str)
1633: Func InputLn(const AMask:str;AEcho:Boolean;ATabWidth:Int;AMaxLineLength:Int):str
1634: Func InputQuery( const ACaption, APrompt :Str; var Value :Str) :Bool
1635: Func InputQuery(const ACaption:str; const APrompt:str; var Value:Str):Bool)
1636: Func InquireSignal( RtlSigNum : Int) : TSignalState
1637: Func InRangeR( const A, Min, Max : Double) :Bool
1638: Func Insert(Index:Int):TCollectionItem
1639: Func Insert(Index:Int):TComboExItem
1640: Func Insert(Index:Int):THeaderSection
1641: Func Insert(Index:Int):TListItem
1642: Func Insert(Index:Int):TStatusPanel
1643: Func Insert(Index:Int):TWorkArea
1644: Func Insert(Index:LongInt; const Text :Str) : LongInt
1645: Func Insert(Sibling : TTreeNode; const S :Str) : TTreeNode
1646: Func INSERTNEWLINEAFTER( AITEM : TMENUITEM) : Int
1647: Func INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : Int
1648: Func InsertNode( Node, Sibling : TTreeNode; const S :Str; Ptr : Pointer) : TTreeNode
1649: Func InsertObject( Index : LongInt; const Text :Str; const Data : Pointer) : LongInt
1650: Func InsertObject( Sibling : TTreeNode; const S :Str; Ptr : Pointer) : TTreeNode
1651: Func Instance : Longint
1652: Func InstanceSize: Longint
1653: Func Int(e : Extended) : Extended;
1654: Func Int64ToStr(i: Int64):Str;
1655: Func IntToBcd( const AValue : Int) : TBcd
1656: Func Intensity( const Color32 : TColor32) : Int;
1657: Func Intensity( const R, G, B : Single) : Single;
1658: Func InterestPayment(const Rate:Ext;Period,NPeriods:Int;const PresentValue,FutureValue:Ext;PaymentTime
      :TPaymentTime):Ext
1659: Func InterestRate(NPeriods:Int;const Payment,PresVal,FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1660: Func InternalDecodeDate(DateTime:TDateTime;var Year,Month,Day,DOW:Word):Boolean
1661: Func InternalRateOfReturn(const Guess:Extended;const CashFlows:array of Double):Extended
```

```
1662: Func InternalUpdateRecord( Tree : TUpdateTree) :Bool
1663: Func IntersectRect( out Rect : TRect; const R1, R2 : TRect) :Bool
1664: Func IntersectRect(out Rect: TRect; const R1, R2: TRect):Bool)
1665: Func IntMibToStr( const Value :Str) :Str
1666: Func IntPower( const Base : Extended; const Exponent : Int) : Extended
1667: Func IntToBin( Value :Card) :Str
1668: Func IntToHex( Value : Int; Digits : Int) :Str;
1669: Func IntToHex(a: Int; b: Int):Str;
1670: Func IntToHex64( Value : Int64; Digits : Int) :Str;
1671: Func IntToHex64(Value: Int64; Digits: Int):Str)
1672: Func IntTo3Str( Value : Longint; separator:Str) :Str
1673: Func inttobool( aInt : LongInt) :Bool
1674: Func IntToStr(i: Int64):Str;
1675: Func IntToStr64(Value: Int64):Str)
1676: Func IOResult: Int
1677: Func IPv6AddressToStr(const AValue: TIdIPv6Address):Str
1678: Func IPAddrToHostName(const IP:Str):Str;
1679: Func IsAccel(VK: Word; const Str:Str):Bool
1680: Func IsAddressInNetwork( Address :Str) :Bool
1681: Func IsAdministrator :Bool
1682: Func IsAlias( const Name :Str) :Bool
1683: Func IsApplicationRunning( const AClassName, ApplName:str):Bool
1684: Func IsASCII( const AByte : Byte) :Bool;
1685: Func IsASCIILDH( const AByte : Byte) :Bool;
1686: Func IsAssembly(const FileName:Str):Bool;
1687: Func IsBcdNegative( const Bcd : TBcd) :Bool
1688: Func IsBinary(const AChar : Char) :Bool
1689: Func IsConsole:Bool)
1690: Func IsDelimiter( Delimiters, S :Str; Index : Int) :Bool
1691: Func IsDelimiter(const Delimiters:Str;const S:Str;Index: Int):Boolean)
1692: Func IsDelphiDesignMode :Bool
1693: Func IsDelphiRunning :Bool
1694: Func IsDFAState :Bool
1695: Func IsDirectory( const FileName :Str) :Bool
1696: Func IsDomain( const S :Str) :Bool
1697: Func IsDragObject(Sender: TObject):Bool;
1698: Func IsEditing :Bool
1699: Func ISEMPTY :Bool
1700: Func IsEqual( Value : TParameters) :Bool
1701: Func ISEQUAL( VALUE : TPARAMS) :Bool
1702: Func IsEqualGUID(const guid1, guid2: TGUID):Bool)
1703: Func IsFirstNode :Bool
1704: Func IsFloatZero( const X : Float) :Bool
1705: Func IsFormatRegistered( Extension, AppID :Str) :Bool
1706: Func IsFormOpen(const FormName:Str):Bool;
1707: Func IsFQDN( const S :Str) :Bool
1708: Func IsGrayScale :Bool
1709: Func IsHex( AChar : Char) :Bool;
1710: Func IsHexString(const AString:Str):Bool;
1711: Func IsHostname( const S :Str) :Bool
1712: Func IsInfinite( const AValue : Double) :Bool
1713: Func IsInLeapYear( const AValue : TDateTime) :Bool
1714: Func IsInternet:Bool;
1715: Func IsLeadChar( ACh : Char) :Bool
1716: Func IsLeapYear( Year : Word) :Bool
1717: Func IsLeapYear(Year: Word):Bool)
1718: Func IsLibrary:Bool)
1719: Func ISLINE :Bool
1720: Func IsLinkedTo( DataSet : TDataSet) :Bool
1721: Func ISLINKEDTO( DATASOURCE : TDATASOURCE) :Bool
1722: Func IsLiteralChar( const EditMask :Str; Offset : Int) :Bool
1723: Func IsMatch( const Pattern, Text :Str) :Bool  //Grep like RegEx
1724: Func IsMainAppWindow( Wnd : HWND) :Bool
1725: Func IsMediaPresentInDrive( Drive : Char) :Bool
1726: Func IsMemoryManagerSet:Bool)
1727: Func IsMultiTableQuery( const SQL : WideString) :Bool
1728: Func IsMultiThread:Bool)
1729: Func IsNumeric( AChar : Char) :Bool;
1730: Func IsNumeric2( const AString :Str) :Bool;
1731: Func IsNTFS:Bool;
1732: Func IsOctal( AChar : Char) :Bool;
1733: Func IsOctalString(const AString:Str) :Bool;
1734: Func IsPathDelimiter( S :Str; Index : Int) :Bool
1735: Func IsPathDelimiter(const S:Str; Index: Int):Bool)
1736: Func IsPM( const AValue : TDateTime) :Bool
1737: Func IsPositiveFloatArray( const X : TDynFloatArray) :Bool
1738: Func IsPortAvailable( ComNum :Card) :Bool;
1739: Func IsCOMPortReal( ComNum :Card) :Bool;
1740: Func IsCOM( ComNum :Card) :Bool;
1741: Func IsCOMPort:Bool;
1742: Func IsPrimeFactor( const F, N :Card) :Bool
1743: Func IsPrimeRM( N :Card) :Bool  //rabin miller
1744: Func IsPrimeTD( N :Card) :Bool  //trial division
1745: Func IsPrivilegeEnabled( const Privilege :Str) :Bool
1746: Func ISqrt( const I : Smallint) : Smallint
1747: Func IsReadOnly(const Filename:Str):Bool;
1748: Func IsRectEmpty( const Rect : TRect) :Bool
1749: Func IsRectEmpty(const Rect: TRect):Bool)
1750: Func IsRelativePrime( const X, Y :Card) :Bool
```

```
1751: Func ISRIGHTTOLEFT :Bool
1752: Func IsRightToLeft:Bool
1753: Func IsSameDay( const AValue, ABasis : TDateTime) :Bool
1754: Func ISSEQUENCED :Bool
1755: Func IsSystemModule( const Module : HMODULE) :Bool
1756: Func IsSystemResourcesMeterPresent :Bool
1757: Func IsTCPPortOpen(dwPort : Word; ipAddressStr:Str):Bool;
1758: Func IsToday( const AValue : TDateTime) :Bool
1759: Func IsToday(const AValue: TDateTime):Bool;
1760: Func IsTopDomain( const AStr :Str) :Bool
1761: Func IsUTF8LeadByte( Lead : Char) :Bool
1762: Func IsUTF8String( const s : UTF8String) :Bool
1763: Func IsUTF8TrailByte( Lead : Char) :Bool
1764: Func ISVALIDCHAR( INPUTCHAR : CHAR) :Bool
1765: Func IsValidDate( const AYear, AMonth, ADay : Word) :Bool
1766: Func IsValidDateDay( const AYear, ADayOfYear : Word) :Bool
1767: Func IsValidDateMonthWeek(const AYear,AMonth,AWeekOfMonth,ADayOfWeek:Word): Bool
1768: Func IsValidDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond:Word): Bool
1769: Func IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) :Bool
1770: Func IsValidIdent( Ident :Str) :Bool
1771: Func IsValidIdent1(const Ident:Str; AllowDots:Bool):Bool)
1772: Func IsValidIP( const S :Str) :Bool
1773: Func IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) :Bool
1774: Func IsValidPNG(stream: TStream):Bool;
1775: Func IsValidJPEG(stream: TStream):Bool;
1776: Func IsValidISBN( const ISBN : Ansistr) :Bool
1777: Func IsVariantManagerSet:Bool; //deprecated;
1778: Func IsVirtualPcGuest :Bool;
1779: Func IsVmWareGuest :Bool;
1780: Func IsVCLControl(Handle: HWnd):Bool;
1781: Func IsWhiteString( const AStr :Str) :Bool
1782: Func IsWindowResponding( Wnd : HWND; Timeout : Int) :Bool
1783: Func IsWoW64:Bool;
1784: Func IsWin64:Bool;
1785: Func IsWow64String(var s:Str):Bool;
1786: Func IsWin64String(var s:Str):Bool;
1787: Func IsWindowsVista:Bool;
1788: Func isPowerof2(num: int64):Bool;
1789: Func powerOf2(exponent: Int): int64;
1790: Func IsZero(const A: Extended; Epsilon: Extended):Bool //overload;
1791: Func IsZero1(const A: Double; Epsilon: Double):Bool //overload;
1792: Func IsZero2(const A: Single; Epsilon: Single):Bool //overload;
1793: Func ItemAtPos( Pos : TPoint; IgnoreTabHeight :Bool) : Int
1794: Func ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):Int
1795: Func ItemRect( Index : Int) : TRect
1796: Func ITEMRECT(INDEX:Int):TRECT
1797: Func ItemWidth( Index : Int) : Int
1798: Func JavahashCode(val:Str): Int;
1799: Func JosephusG(n,k: Int; var graphout:Str): Int;
1800: Func JulianDateToDateTime( const AValue : Double) : TDateTime
1801: Func JustName(PathName :Str) :Str;  //in path and ext
1802: Func JvMessageBox( const Text, Caption :Str; Flags : DWORD) : Int;
1803: Func JvMessageBox1( const Text :Str; Flags : DWORD) : Int;
1804: Func KeepAlive :Bool
1805: Func KeysToShiftState(Keys: Word): TShiftState;
1806: Func KeyDataToShiftState(KeyData: Longint): TShiftState;
1807: Func KeyboardStateToShiftState2(const KeyboardState: TKeyboardState):TShiftState;
1808: Func KeyboardStateToShiftState: TShiftState; overload;
1809: Func Languages : TLanguages
1810: Func Last: TClass
1811: Func Last: TComponent
1812: Func Last: TObject
1813: Func LastDelimiter( Delimiters, S :Str) : Int
1814: Func LastDelimiter(const Delimiters:Str; const S:Str): Int)
1815: Func LastPos( const ASubstr :Str; const AStr :Str) : Int
1816: Func Latitude2WGS84(lat: double): double;
1817: Func LCM(m,n:longint):longint;
1818: Func LCMJ( const X, Y :Card) :Card
1819: Func Ldexp( const X : Extended; const P : Int) : Extended
1820: Func LeftStr( const AText : Ansistr; const ACount : Int) : Ansistr;
1821: Func LeftStr( const AText : WideString; const ACount : Int) : WideString;
1822: Func Length: Int;
1823: Proc LetFileList(FileList: TStringlist; apath:Str);
1824: Func lengthmp3(mp3path:Str):Int;
1825: Func LineInRect( const P1, P2 : TPoint; const Rect : TRect) :Bool;
1826: Func LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) :Bool;
1827: Func LinesCount(sfilename:str):extended;
1828: Func TextFileLineCount(const FileName:Str): Int;
1829: Func LineSegmentIntersection(const L1P1:TFloatPoint;L1P2:TFloatPoint;const L2P1:TFloatPoint;
      L2P2:TFloatPoint;var P:TFloatPoint):Boolean
1830: Func LineStart(Buffer, BufPos: PChar): PChar
1831: Func LineStart(Buffer, BufPos: PChar): PChar)
1832: Func ListSeparator: char;
1833: Func Ln(x: Extended): Extended;
1834: Func LnXP1( const X : Extended) : Extended
1835: Func Lo(vdat: word): byte;
1836: Func LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1837: Func LoadedModulesList(const List:TStrings;ProcessID:DWORD;HandlesOnly:Boolean):Bool
1838: Func LoadFileAsString( const FileName :Str) :Str
```

```
1839: Func LoadFromFile( const FileName :Str) : TBitmapLoader
1840: Func LoadFile(const FileName: TFileName):Str;
1841: Func LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1842: Func LoadPackage(const Name:Str): HMODULE
1843: Func LoadResource( ModuleHandle : HMODULE; ResHandle: TResourceHandle):HGLOBAL
1844: Func LoadStr( Ident : Int) :Str
1845: Func LoadString(Instance: Longint; IDent: Int; Buffer: PChar; Size: Int):Int;stdcall;
1846: Func LoadWideStr( Ident : Int) : WideString
1847: Func LOCATE(const KEYFIELDS:str;const KEYVALUES:VARIANT;OPTIONS:TLOCATEOPTIONS):BOOLEAN
1848: Func LockRegion(libOffset:Longint;cb:Largeint;dwLockType:Longint): HResult
1849: Func LockServer( fLock : LongBool) : HResult
1850: Func LockVolume( const Volume :Str; var Handle : THandle) :Bool
1851: Func Log( const X : Extended) : Extended
1852: Func Log10( const X : Extended) : Extended
1853: Func Log2( const X : Extended) : Extended
1854: Func LogBase10(X: Float): Float;
1855: Func LogBase2(X: Float): Float;
1856: Func LogBaseN(Base, X: Float): Float;
1857: Func LogN( const Base, X : Extended) : Extended
1858: Func LogOffOS :Bool
1859: Func LoginDialog( const ADatabaseName :Str; var AUserName, APassword :Str):Bool
1860: Func LoginDialogEx(const ADbName:str;var AUserName,APassword:str;NameReadOnly:Bool):Bool;
1861: Func LongDateFormat:Str;
1862: Func LongTimeFormat:Str;
1863: Func LongWordToFourChar( ACardinal : LongWord) :Str
1864: Func LOOKUP(const KEYFIELDS:Str;const KEYVALUES:VARIANT;const RESULTFIELDS:str):VARIANT
1865: Func LookupName( const name :Str) : TInAddr
1866: Func LookupService( const service :Str) : Int
1867: Func Low: Int64;
1868: Func LowerCase( S :Str) :Str
1869: Func Lowercase(s : AnyString) : AnyString;
1870: Func LRot( const Value : Byte; const Count : TBitRange) : Byte;
1871: Func LRot1( const Value : Word; const Count : TBitRange) : Word;
1872: Func LRot2( const Value : Int; const Count : TBitRange) : Int;
1873: Func MainInstance: longword
1874: Func MainThreadID: longword
1875: Func Map(x, in_min, in_max, out_min, out_max: Int): Int;  //arduino
1876: Func mapMax(ax, in_min, in_max, out_min, out_max: Int): Int;
1877: Func MakeCanonicalIPv4Address( const AAddr :Str) :Str
1878: Func MakeCanonicalIPv6Address( const AAddr :Str) :Str
1879: Func MakeDIB( out Bitmap : PBitmapInfo) : Int
1880: Func MakeDWordIntoIPv4Address( const ADWord :Card) :Str
1881: Func MakeFile(const FileName:Str): Int);
1882: Func MakeLong(A, B: Word): Longint)
1883: Func MakeTempFilename( const APath :Str) :Str
1884: Func MakeValidFileName( const Str :Str) :Str
1885: Func MakeValueMap( Enumeration :Str; ToCds :Bool) :Str
1886: Func MakeWord(A, B: Byte): Word)
1887: Func MakeYear4Digit( Year, Pivot : Int) : Int
1888: Func MapDateTime(const DateFormatType:str;DateFormat:str;Value:str;ToCds:Bool):str
1889: Func MapValues( Mapping :Str; Value :Str) :Str
1890: Func MaskDoFormatText(const EditMask:str;const Value:str;Blank:Char):str
1891: Func MaskGetCharType( const EditMask :Str; MaskOffset : Int) : TMaskCharType
1892: Func MaskGetCurrentDirectives( const EditMask :Str; MaskOffset:Int):TMaskDirectives
1893: Func MaskGetFldSeparator( const EditMask :Str) : Int
1894: Func MaskGetMaskBlank( const EditMask :Str) : Char
1895: Func MaskGetMaskSave( const EditMask :Str) :Bool
1896: Func MaskIntlLiteralToChar( IChar : Char) : Char
1897: Func MaskOffsetToOffset( const EditMask :Str; MaskOffset : Int) : Int
1898: Func MaskOffsetToWideOffset( const EditMask :Str; MaskOffset : Int) : Int
1899: Func MaskString( Mask, Value :Str) :Str
1900: Func Match( const sString :Str) : TniRegularExpressionMatchResul
1901: Func Match1( const sString:Str;iStart:Int):TniRegularExpressionMatchResult
1902: Func Matches( const Filename :Str) :Bool
1903: Func MatchesMask( const Filename, Mask :Str) :Bool
1904: Func MatchStr( const AText :Str; const AValues : array of string):Bool
1905: Func MatchText( const AText :Str; const AValues : array of string):Bool
1906: Func Max( AValueOne, AValueTwo : Int) : Int
1907: Func Max(const x,y: Int): Int;
1908: Func Max1(const B1, B2 : Shortint) : Shortint;
1909: Func Max2(const B1, B2 : Smallint) : Smallint;
1910: Func Max3(const B1, B2 : Word) : Word;
1911: Func Max3(const x,y,z: Int): Int;
1912: Func Max4(const B1, B2 : Int) : Int;
1913: Func Max5(const B1, B2 :Card) :Card;
1914: Func Max6(const B1, B2 : Int64) : Int64;
1915: Func Max64(const AValueOne, AValueTwo : Int64) : Int64
1916: Func MaxFloat(const X, Y : Float) : Float
1917: Func MaxFloatArray(const B : TDynFloatArray) : Float
1918: Func MaxFloatArrayIndex( const B : TDynFloatArray) : Int
1919: Func MaxIntValue(const Data: array of Int):Int)
1920: Func MaxJ( const B1, B2 : Byte) : Byte;
1921: Func MaxPath:Str;
1922: Func MaxValue(const Data: array of Double): Double)
1923: Func MaxCalc( const Formula :Str) : Extended  //math expression parser
1924: Proc MaxCalcF( const Formula :Str);   //out to console memo2
1925: Func MaxCalcS( const Formula :Str):Str;
1926: Func MD5(const fileName:Str):Str;
1927: Func Mean( const Data : array of Double) : Extended
```

```
1928: Func Median( const X : TDynFloatArray) : Float
1929: Func Memory : Pointer
1930: Func MemoryPos( const ASubStr :Str; MemBuff : PChar; MemorySize: Int): Int
1931: Func MessageBox(hndl:Card; text, caption:Str; utype:Card): Int;
1932: Func MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):Int
1933: Func MessageDlg(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint):Int;
1934: Func MessageDlg1(const
      Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn):Int;
1935: Func MessageDlgPos(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y:Int):Int;
1936: Func MessageDlgPos1(const Msg:str; DlgType:TMsgDlgType; Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y:
      Int;DefaultButton:TMsgDlgBtn):Int;
1937: Func MessageDlgPosHelp(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y:
      Int;const HelpFileName:str):Int;
1938: Func MessageDlgPosHelp1(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y :
      Int;const HelpFileName:str; DefaultButton:TMsgDlgBtn):Int;
1939: Func MibToId( Mib :Str) :Str
1940: Func MidStr( const AText : Ansistr; const AStart, ACount : Int) : Ansistr;
1941: Func MidStr( const AText : WideString; const AStart, ACount : Int) : WideString;
1942: Func microsecondsToCentimeters(mseconds:longint):longint;//340m/s speed of sound
1943: Func Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1944: Func MIDIOut( DeviceID :Card) : IJclMIDIOut
1945: Proc GetMidiOutputs( const List : TStrings)
1946: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne
1947: Proc GetGEOMap(C_form,apath:str; const Data:Str);//c_form:[html/json/xml]
1948: Func MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1949: Func MIDINoteToStr( Note : TMIDINote) :Str
1950: Func WinMidiOut( DeviceID :Card) : IJclWinMidiOut
1951: Proc GetMidiOutputs( const List : TStrings)
1952: Proc MidiOutCheck( Code : MMResult)
1953: Proc MidiInCheck( Code : MMResult)
1954: Func MilliSecondOf( const AValue : TDateTime) : Word
1955: Func MilliSecondOfTheDay( const AValue : TDateTime): LongWord
1956: Func MilliSecondOfTheHour( const AValue : TDateTime): LongWord
1957: Func MilliSecondOfTheMinute( const AValue : TDateTime): LongWord
1958: Func MilliSecondOfTheMonth( const AValue : TDateTime): LongWord
1959: Func MilliSecondOfTheSecond( const AValue : TDateTime): Word
1960: Func MilliSecondOfTheWeek( const AValue : TDateTime): LongWord
1961: Func MilliSecondOfTheYear( const AValue : TDateTime): Int64
1962: Func MilliSecondsBetween( const ANow, AThen : TDateTime): Int64
1963: Func MilliSecondSpan( const ANow, AThen : TDateTime): Double
1964: Func milliToDateTime( MilliSecond : LongInt) : TDateTime;
1965: Func Micros( const Timer : THPTimer; const TimerRunning:Boolean):Int64
1966: Func millis: int64;
1967: Func Min( AValueOne, AValueTwo : Int) : Int
1968: Func Min1(const B1,B2: Shortint) : Shortint;
1969: Func Min2(const B1,B2: Smallint) : Smallint;
1970: Func Min3(const B1,B2: Word) : Word;
1971: Func Min4(const B1,B2: Int) : Int;
1972: Func Min5(const B1,B2:Card) :Card;
1973: Func Min6(const B1,B2: Int64) : Int64;
1974: Func Min64( const AValueOne, AValueTwo : Int64) : Int64
1975: Func MinClientRect : TRect;
1976: Func MinClientRect1( IncludeScroller :Bool) : TRect;
1977: Func MinClientRect2( TabCount : Int; IncludeScroller :Bool) : TRect;
1978: Func MinFloat( const X, Y : Float) : Float
1979: Func MinFloatArray( const B : TDynFloatArray) : Float
1980: Func MinFloatArrayIndex( const B : TDynFloatArray) : Int
1981: Func MinimizeName(const Filename:str;Canvas : TCanvas; MaxLen:Int):Str
1982: Func MinimizeName(const Filename:TFileName;Canvas:TCanvas; MaxLen:Int):FileName
1983: Func MinimizeName(const Filename:Str; Canvas: TCanvas;MaxLen: Int):TFileName
1984: Func MinIntValue( const Data : array of Int) : Int
1985: Func MinIntValue(const Data: array of Int):Int)
1986: Func MinJ( const B1, B2 : Byte) : Byte;
1987: Func MinuteOf( const AValue : TDateTime) : Word
1988: Func MinuteOfTheDay( const AValue : TDateTime) : Word
1989: Func MinuteOfTheHour( const AValue : TDateTime) : Word
1990: Func MinuteOfTheMonth( const AValue : TDateTime) : Word
1991: Func MinuteOfTheWeek( const AValue : TDateTime) : Word
1992: Func MinuteOfTheYear( const AValue : TDateTime) : LongWord
1993: Func MinutesBetween( const ANow, AThen : TDateTime) : Int64
1994: Func MinuteSpan( const ANow, AThen : TDateTime) : Double
1995: Func MinValue( const Data : array of Double) : Double
1996: Func MinValue(const Data: array of Double): Double)
1997: Func MixerLeftRightToArray( Left, Right :Card) : TDynCardinalArray
1998: Func MMCheck( const MciError : MCIERROR; const Msg :Str) : MCIERROR
1999: Func ModFloat( const X, Y : Float) : Float
2000: Func ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
2001: Func Modify( const Key :Str; Value : Int) :Bool
2002: Func ModuleCacheID :Card
2003: Func ModuleFromAddr( const Addr : Pointer) : HMODULE
2004: Func MonitorFromPoint(const Point:TPoint;MonitorDefault:TMonitorDefaultTo): TMonitor
2005: Func MonitorFromRect(const Rect:TRect;MonitorDefault:TMonitorDefaultTo):TMonitor
2006: Func MonitorFromWindow(const Handle:THandle; MonitorDefault:TMonitorDefaultTo): TMonitor
2007: Func MonthOf( const AValue : TDateTime) : Word
2008: Func MonthOfTheYear( const AValue : TDateTime) : Word
2009: Func MonthsBetween( const ANow, AThen : TDateTime) : Int
2010: Func MonthSpan( const ANow, AThen : TDateTime) : Double
2011: Func MonthStr( DateTime : TDateTime) :Str
2012: Func MouseCoord( X, Y : Int) : TGridCoord
```

```
2013: Func MOVEBY( DISTANCE : Int) : Int
2014: Func MoveFile( Source, Dest :Str; Flags : FILEOP_FLAGS) :Bool
2015: Func MoveNext :Bool
2016: Func MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
2017: Func MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
2018: Func Name :Str
2019: Func NetPresentValue(const Rate:Extended;const CashFlows:array of Double;PaymentTime:TPaymentTime):Extended
2020: Func NetworkVolume(DriveChar: Char):Str
2021: Func NEWBOTTOMLINE : Int
2022: Func NewCompareNode(Field: TField;Operator:TCANOperator; const Value:Variant) : PExprNode
2023: Func NEWITEM(const ACAPTION:str;ASHORTCUT:TSHORTCUT;ACHECKED,AENABLED:BOOL;AONCLICK:
      TNOTIFYEVENT;HCTX:WORD;const ANAME:str):TMENUITEM
2024: Func NEWLINE : TMENUITEM
2025: Func NEWMENU( OWNER : TCOMPONENT;const ANAME:str;ITEMS: array of TMenuItem): TMAINMENU
2026: Func NewNode(Kind: TExprNodeKind;Operator:TCANOperator;const Data:Variant;Left,Right:PExprNode):PExprNode
2027: Func NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:str; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN; const
      ITEMS : array of TCMENUITEM) : TPOPUPMENU
2028: Func NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
2029: Func NEWSUBMENU(const ACAPT:str;HCTX:WORD;const ANAME:str;ITEMS:array of TMenuItem;AENABLED:BOOL):
      TMENUITEM
2030: Func NEWTOPLINE : Int
2031: Func Next : TIdAuthWhatsNext
2032: Func NextCharIndex( S :Str; Index : Int) : Int
2033: Func NextRecordSet : TCustomSQLDataSet
2034: Func NextRecordset( var RecordsAffected : Int) : _Recordset
2035: Func NextSQLToken1(var p: WideChar;out Token:WideString;CurSection:TSQLToken):TSQLToken;
2036: Func NextToken : Char
2037: Func nextToken : WideString
2038: Func NextToken:Char
2039: Func Norm( const Data : array of Double) : Extended
2040: Func NormalizeAngle( const Angle : Extended) : Extended
2041: Func NormalizeBcd(const InBcd TBcd;var OutBcd:TBcd;const Prec,Scale Word):Bool
2042: Func NormalizeRect( const Rect : TRect) : TRect
2043: Func NormalizeRect(const Rect: TRect): TRect;
2044: Func Now :TDateTime
2045: Func Now2:tDateTime
2046: Func NumProcessThreads:Int
2047: Func NumThreadCount : Int
2048: Func NthDayOfWeek( const AValue : TDateTime) : Word
2049: Func NtProductType : TNtProductType
2050: Func NtProductTypeString :Str
2051: Func Null: Variant;
2052: Func NullPoint : TPoint
2053: Func NullRect : TRect
2054: Func Null2Blank(aString:str):str;
2055: Func NumberOfPeriods(const Rate:Extended;Payment:Extended;const PresentValue,
      FutureValue:Extended;PaymentTime:TPaymentTime):Extended
2056: Func NumIP:Int
2057: Func Odd(x: Longint):Bool;
2058: Func OffsetFromUTC : TDateTime
2059: Func OffsetPoint( const P, Offset : TPoint) : TPoint
2060: Func OffsetRect( var Rect : TRect; DX : Int; DY : Int) :Bool
2061: Func OffsetRect(var Rect: TRect; DX:Int; DY:Int):Bool)
2062: Func OffsetToMaskOffset( const EditMask :Str; Offset : Int) : Int
2063: Func OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment):Bool
2064: Func OldBCDToCurr( const BCD : TBcd; var Curr : Currency) :Bool
2065: Func OldCurrToBCD(const Curr:Currency;var BCD:TBcd;Precision:Int;Decimals:Int):Bool
2066: Func OpenBit:Int
2067: Func OpenDatabase : TDatabase
2068: Func OpenDatabase( const DatabaseName:str):TDatabase
2069: Proc OpenDir(adir:Str);
2070: Func OpenGLColorToWinColor( const Red, Green, Blue:Float):TColor
2071: Func OpenMap(const Data:Str):Bool;
2072: Func OpenMapX(const Data:Str):Bool;
2073: Func OpenObject( Value : PChar) :Bool;
2074: Func OpenObject1( Value :Str) :Bool;
2075: Proc Openweb('http://snippets.delphidabbler.com/#')
2076: Proc OpenURL( const AText : TKkString);
2077: Proc OpenWeb( const AText : TKkString);
2078: Proc OpenBrowser( const AText : TKkString);
2079: Func OpenSession( const SessionName :Str) : TSession
2080: Func OpenVolume( const Drive : Char) : THandle
2081: Func OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte):Card
2082: Func OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte): LongWord
2083: Func OrdToBinary( const Value: Byte) :Str;
2084: Func OrdToBinary1(const Value: Shortint) :Str;
2085: Func OrdToBinary2(const Value: Smallint) :Str;
2086: Func OrdToBinary3(const Value: Word) :Str;
2087: Func OrdToBinary4(const Value: Int) :Str;
2088: Func OrdToBinary5(const Value:Card) :Str;
2089: Func OrdToBinary6(const Value: Int64) :Str;
2090: Func OSCheck( RetVal :Bool) :Bool
2091: Func OSFileTypeToString( const OSFileType:DWORD;const OSFileSubType:DWORD):str
2092: Func OSIdentToString( const OSIdent : DWORD) :Str
2093: Func Output: Text)
2094: Func Overlay( ImageIndex : Int; Overlay : TOverlay):Boolean
2095: Func Owner : TCustomListView
2096: Func Owner : TPersistent
2097: Func PadInputLiterals(const EditMask:str;const Value:str;Blank:Char):str
```

```
2098: Func PadL( pStr :Str; pLth : Int) :Str
2099: Func Padl(s : AnyString;I : longInt) : AnyString;
2100: Func PadLCh( pStr :Str; pLth : Int; pChr : char) :Str
2101: Func PadR( pStr :Str; pLth : Int) :Str
2102: Func Padr(s : AnyString;I : longInt) : AnyString;
2103: Func PadRCh( pStr :Str; pLth : Int; pChr : char) :Str
2104: Func PadString(const AString:Str;const ALen:Int; const AChar:Char):str
2105: Func Padz(s : AnyString;I : longInt) : AnyString;
2106: Func PaethPredictor( a, b, c : Byte) : Byte
2107: Func PARAMBYNAME( const VALUE :Str) : TPARAM
2108: Func ParamByName( const Value : WideString): TParameter
2109: Func ParamCount: Int
2110: Func ParamsEncode( const ASrc :Str):Str
2111: Func ParamStr(Index: Int):Str)
2112: Func ParseDate( const DateStr :Str): TDateTime
2113: Func PARSESQL( SQL :Str; DOCREATE :Bool) :Str
2114: Func ParseSQL( SQL : WideString; DoCreate :Bool): WideString
2115: Func PathAddExtension( const Path, Extension :Str):str
2116: Func PathAddSeparator( const Path :Str) :Str
2117: Func PathAppend( const Path, Append :Str) :Str
2118: Func PathBuildRoot( const Drive : Byte) :Str
2119: Func PathCanonicalize( const Path :Str) :Str
2120: Func PathCommonPrefix( const Path1, Path2 :Str) : Int
2121: Func PathCompactPath(const DC:HDC;const Path:str;const Width:Int;CmpFmt:TCompactPath):str;
2122: Func PathCompactPath1(const Canv:TCanvas;const Path:str;const Width:Int;CmpFmt:TCompactPath):str;
2123: Func PathEncode( const ASrc :Str) :Str
2124: Func PathExtractFileDirFixed( const S : Ansistr): Ansistr
2125: Func PathExtractFileNameNoExt( const Path :Str):Str
2126: Func PathExtractPathDepth( const Path :Str; Depth: Int):str
2127: Func PathGetDepth( const Path :Str): Int
2128: Func PathGetLongName( const Path :Str):Str
2129: Func PathGetLongName2( Path :Str):Str
2130: Func PathGetShortName( const Path :Str):Str
2131: Func PathIsAbsolute( const Path :Str): Bool
2132: Func PathIsChild( const Path, Base : Ansistr): Bool
2133: Func PathIsDiskDevice( const Path :Str): Bool
2134: Func PathIsUNC( const Path :Str) : Bool
2135: Func PathRemoveExtension( const Path:Str):Str
2136: Func PathRemoveSeparator( const Path:Str):Str
2137: Func Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2138: Func Peek: Pointer
2139: Func Peek: TObject
2140: Func PERFORM(MSG:Card;WPARAM,LPARAM:LONGINT):LONGINT
2141: Func PeriodPayment(const Rate:Extended;Period,NPeriods:Int; const PresentValue,FutureValue:
      Extended;PaymentTime:TPaymentTime):Extended
2142: Func Permutation(npr, k: Int): extended;
2143: Func PermutationInt(npr, k: Int): Int64;
2144: Func PermutationJ( N, R :Card) : Float
2145: Func Pi : Extended;
2146: Func PiE : Extended;
2147: Func PixelsToDialogsX( const Pixels : Word) : Word
2148: Func PixelsToDialogsY( const Pixels : Word) : Word
2149: Func PlaySound(s: pchar; flag,syncflag: Int):Bool;
2150: Func Point( X, Y : Int) : TPoint
2151: Func Point(X, Y: Int): TPoint)
2152: Func PointAssign( const X, Y : Int) : TPoint
2153: Func PointDist( const P1, P2 : TPoint) : Double;
2154: Func PointDist(const P1,P2: TFloatPoint): Double;
2155: Func PointDist1( const P1, P2 : TFloatPoint) : Double;
2156: Func PointDist2(const P1,P2: TPoint): Double;
2157: Func PointEqual( const P1, P2 : TPoint) :Bool
2158: Func PointIsNull( const P : TPoint) :Bool
2159: Func PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint): Double
2160: Func Poly( const X : Extended; const Coefficients : array of Double): Extended
2161: Func PortTCPIsOpen(dwPort : Word; ipAddressStr:Str):bool;
2162: Func IsTCPPortOpen(dwPort : Word; ipAddressStr:Str):bool;
2163: Func Pop : Pointer
2164: Func Pop : TObject
2165: Func PopnStdDev( const Data : array of Double) : Extended
2166: Func PopnVariance( const Data : array of Double) : Extended
2167: Func PopulationVariance( const X : TDynFloatArray) : Float
2168: Func Pos(SubStr, S: AnyString): Longint;
2169: Func PosEqual( const Rect : TRect) :Bool
2170: Func PosEx( const SubStr, S :Str; Offset : Int) : Int
2171: Func PosInSmallIntArray(const ASearchInt:SmallInt;AArray:array of SmallInt):Int
2172: Func PosInStrArray(const SearchStr:str;Contents:array of str;const CaseSensitive:Bool):Int
2173: Func Post1( AURL:str; const ASource : TStrings) :Str;
2174: Func Post2( AURL:str; const ASource : TStream) :Str;
2175: Func Post3( AURL:str; const ASource : TIdMultiPartFormDataStream) :Str;
2176: Func PostData( const UserData : WideString; const CheckSum : DWORD) :Bool
2177: Func PostData( const UserData : WideString; const CheckSum : Int):Bool
2178: Func PostMessage(hWnd:HWND;Msg:longword;wParam: ongint;lParam:longint):Bool;
2179: Func Power( const Base, Exponent : Extended) : Extended
2180: Func PowerBig(aval, n:Int):Str;
2181: Func PowerIntJ( const X : Float; N : Int) : Float;
2182: Func PowerJ( const Base, Exponent : Float) : Float;
2183: Func PowerOffOS :Bool
2184: Func PreformatDateString( Ps :Str) :Str
```

```
2185: Func PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2186: Func PrimeFactors( N :Card) : TDynCardinalArray
2187: Func Printer : TPrinter
2188: Func ProcessPath2(const ABasePath:str;const APath:str;const APathDelim:str):str
2189: Func ProcessResponse : TIdHTTPWhatsNext
2190: Func ProduceContent :Str
2191: Func ProduceContentFromStream( Stream : TStream) :Str
2192: Func ProduceContentFromString( const S :Str) :Str
2193: Func ProgIDToClassID(const ProgID:Str): TGUID;
2194: Func PromptDataLinkFile(ParentHandle:THandle;InitialFile:WideString):WideString
2195: Func PromptDataSource(ParentHandle:THandle;InitialString:WideString):WideString
2196: Func PromptForFileName(var AFileName:str;const AFilter:str;const ADefaultExt:str;const ATitle:str;const
      AInitialDir:str;SaveDialog:Bool):Bool
2197: Func PromptForFileName(var AFileName:str;const AFilter:str;const ADefaultExt:str;const ATitle:str;const
      AInitialDir:Str;SaveDialog:Bool):Bool)
2198: Func PSScriptNeedFile(Sender:TObject;const OrginFileName:Str;var FileName,Output:Str):Bool
2199: Func PtInRect( const Rect : TRect; const P : TPoint) :Bool
2200: Func PtInRect(const Rect: TRect; const P: TPoint):Bool)
2201: Func Push( AItem : Pointer) : Pointer
2202: Func Push( AObject : TObject) : TObject
2203: Func Put1( AURL :Str; const ASource : TStream) :Str;
2204: Func Pythagoras( const X, Y : Extended) : Extended
2205: Func queryDLLInterface( var queryList : TStringList) : TStringList
2206: Func queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2207: Func QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2208: Func queryPerformanceCounter2(mse: int64): int64;
2209: //Func QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2210: //Func QueryPerformanceFrequency(mse: int64):Bool;
2211: Func QueryPerformanceCounter(var lcount: Int64):Bool; stdcall;
2212: Func QueryPerformanceFrequency(var lfreq: int64):Bool; stdcall;
2213: Proc QueryPerformanceCounter1(var aC: Int64);
2214: Func QueryPerformanceFrequency1(var freq: int64):Bool;
2215: Func Quote( const ACommand :Str) : SmallInt
2216: Func QuotedStr( S :Str) :Str
2217: Func RadToCycle( const Radians : Extended) : Extended
2218: Func RadToDeg( const Radians : Extended) : Extended
2219: Func RadToDeg( const Value : Extended) : Extended;
2220: Func RadToDeg1(const Value : Double) : Double;
2221: Func RadToDeg2(const Value : Single) : Single;
2222: Func RadToGrad(const Radians : Extended) : Extended
2223: Func RadToGrad(const Value : Extended) : Extended;
2224: Func RadToGrad1( const Value : Double) : Double;
2225: Func RadToGrad2( const Value : Single) : Single;
2226: Func RandG( Mean, StdDev : Extended) : Extended
2227: Func Random(const ARange: Int): Int;
2228: Func random2(a: Int): double
2229: Func RandomE: Extended;
2230: Func RandomF: Extended;
2231: Func RandomFrom( const AValues : array of string):Str;
2232: Func RandomRange( const AFrom, ATo : Int) : Int
2233: Func randSeed: longint
2234: Proc setRandseed; //no params!
2235: Func RawToDataColumn( ACol : Int) : Int
2236: Func Read : Char
2237: Func Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2238: Func Read(Buffer:str;Count:LongInt):LongInt
2239: Func ReadBinaryStream( const Section, Name :Str; Value : TStream) : Int
2240: Func ReadBool( const Section, Ident :Str; Default :Bool) :Bool
2241: Func ReadCardinal( const AConvert :Bool) :Card
2242: Func ReadChar : Char
2243: Func ReadClient( var Buffer, Count : Int) : Int
2244: Func ReadDate( const Section, Name :Str; Default : TDateTime) : TDateTime
2245: Func ReadDateTime( const Section, Name :Str; Default : TDateTime) : TDateTime
2246: Func ReadFloat( const Section, Name :Str; Default : Double) : Double
2247: Func ReadFromStack(const ARaiseExceptfDiscon:Bool;ATimeout:Int;const ARaiseExceptTimeout:Bool):Int
2248: Func ReadInt( const AConvert :Bool) : Int
2249: Func ReadInt( const Section, Ident :Str; Default : Longint) : Longint
2250: Func ReadLn :Str
2251: Func ReadLn(ATerminator:Str;const ATimeout:Int;AMaxLineLength:Int):str
2252: Func Readln(question:Str):Str;
2253: Func readm:Str;  //read last line in memo2 - console!
2254: Func ReadLnWait( AFailCount : Int) :Str
2255: Func ReadReg(Base: HKEY; KeyName, ValueName:Str):Str;
2256: Func ReadRegistry(Base: HKEY; KeyName, ValueName:Str):Str;
2257: Func ReadSmallInt( const AConvert :Bool) : SmallInt
2258: Func ReadString( const ABytes : Int) :Str
2259: Func ReadString( const Section, Ident, Default :Str) :Str
2260: Func ReadString( Count : Int) :Str
2261: Func ReadTime( const Section, Name :Str; Default : TDateTime) : TDateTime
2262: Func ReadTimeStampCounter : Int64
2263: Func RebootOS :Bool
2264: Func Receive( ATimeOut : Int) : TReplyStatus
2265: Func ReceiveBuf( var Buf, Count : Int) : Int
2266: Func ReceiveLength : Int
2267: Func ReceiveText :Str
2268: Func ReceiveSerialData(var Data: TByteArray; DataSize:Card):Card
2269: Func ReceiveSerialText:Str
2270: Func RecodeDate(const AValue:TDateTime;const AYear,AMonth,ADay:Word):TDateTime
```

```
2271: Func RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,AMilliSec:Word):TDateTime
2272: Func RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2273: Func RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2274: Func RecodeMilliSecond(const AValue:TDateTime;const AMilliSecond:Word):TDateTime
2275: Func RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2276: Func RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2277: Func RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2278: Func RecodeTime(const AValue:TDateTime;const AHour,AMinute,ASecd,AMilliSec:Word):TDateTime
2279: Func RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2280: Func Reconcile( const Results : OleVariant) :Bool
2281: Func Rect( Left, Top, Right, Bottom : Int) : TRect
2282: Func Rect(ALeft: Int; ATop: Int; ARight: Int; ABottom: Int): TRect)
2283: Func Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2284: Func RectAssign( const Left, Top, Right, Bottom : Int) : TRect
2285: Func RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2286: Func RectBounds( const Left, Top, Width, Height : Int) : TRect
2287: Func RectCenter( const R : TRect) : TPoint
2288: Func RectEqual( const R1, R2 : TRect) :Bool
2289: Func RectHeight( const R : TRect) : Int
2290: Func RectIncludesPoint( const R : TRect; const Pt : TPoint) :Bool
2291: Func RectIncludesRect( const R1, R2 : TRect) :Bool
2292: Func RectIntersection( const R1, R2 : TRect) : TRect
2293: Func RectIntersectRect( const R1, R2 : TRect) :Bool
2294: Func RectIsEmpty( const R : TRect) :Bool
2295: Func RectIsNull( const R : TRect) :Bool
2296: Func RectIsSquare( const R : TRect) :Bool
2297: Func RectIsValid( const R : TRect) :Bool
2298: Func RectsAreValid( R : array of TRect) :Bool
2299: Func RectUnion( const R1, R2 : TRect) : TRect
2300: Func RectWidth( const R : TRect) : Int
2301: Func RedComponent( const Color32 : TColor32) : Int
2302: Func Refresh :Bool
2303: Func RefStringListCopy(aRefArray:TStringlist):TStringList;
2304: Func RegisterConversionFamily( const ADescription :Str) : TConvFamily
2305: Func RegisterConversionType(AConvTypeInfo: TConvTypeInfo; out AType : TConvType) :Bool;
2306: Func RegisterConversionType(const AFam:TConvFamil;const ADescr:str;const AFact:Double):TConvType
2307: Func RegistryRead(keyHandle: Longint; keyPath, myField:Str):Str;
2308: Func ReleaseDC(hdwnd: HWND; hdc: HDC): Int;
2309: Func ReleaseHandle : HBITMAP
2310: Func ReleaseHandle : HENHMETAFILE
2311: Func ReleaseHandle : HICON
2312: Func ReleasePalette : HPALETTE
2313: Func RemainderFloat( const X, Y : Float) : Float
2314: Func Remove(AClass : TClass) : Int
2315: Func Remove(AComponent : TComponent) : Int
2316: Func Remove(AItem : Int) : Int
2317: Func Remove(AItem : Pointer) : Pointer
2318: Func Remove(AItem : TObject) : TObject
2319: Func Remove(AObject : TObject) : Int
2320: Func RemoveBackslash( const PathName :Str) :Str
2321: Func RemoveDF( aString :Str) :Str  //removes thousand separator
2322: Func RemoveDir( Dir :Str) :Bool
2323: Func RemoveDir(const Dir:Str):Bool)
2324: Func RemoveDirectory(PathName: PChar): WordBool; stdcall;
2325: Func RemoveFileExt( const FileName :Str) :Str
2326: Func RemoveHeaderEntry( AHeader, AEntry :Str) :Str
2327: Func RenameFile( OldName, NewName :Str) :Bool
2328: Func RenameFile(const OldName:Str; const NewName:Str):Bool)
2329: Func ReplaceStr( const AText, AFromText, AToText :Str) :Str
2330: Func ReplaceText( const AText, AFromText, AToText :Str):Str
2331: Func Replicate(c : char;I : longInt) :Str;
2332: Func Request : TWebRequest
2333: Func ResemblesText( const AText, AOther :Str):Bool
2334: Func Reset :Bool
2335: Func Reset2(mypath:Str): TStringlist //string;
2336: Func ResInstLoad(Instance:THandle;ResType:TResType;const Name:str;MaskColor:TColor):Bool
2337: Func ResourceLoad(ResType:TResType;const Name string;MaskColor:TColor):oolean
2338: Func Response : TWebResponse
2339: Func ResumeSupported :Bool
2340: Func RETHINKHOTKEYS :Bool
2341: Func RETHINKLINES :Bool
2342: Func Retrieve( const MsgNum : Int; AMsg : TIdMessage) :Bool
2343: Func RetrieveCurrentDir :Str
2344: Func RetrieveDeltas( const cdsArray : array of TClientDataset): Variant
2345: Func RetrieveHeader( const MsgNum : Int; AMsg : TIdMessage):Bool
2346: Func RetrieveMailBoxSize : Int
2347: Func RetrieveMsgSize( const MsgNum : Int) : Int
2348: Func RetrieveProviders( const cdsArray : array of TClientDataset): Variant
2349: Func RetrieveRaw( const MsgNum : Int; const Dest: TStrings):bool
2350: Func ReturnMIMEType( var MediaType, EncType :Str):Bool
2351: Func ReverseBits( Value:Byte) : Byte;
2352: Func ReverseBits1(Value:Shortint) : Shortint;
2353: Func ReverseBits2(Value:Smallint) : Smallint;
2354: Func ReverseBits3(Value:Word) : Word;
2355: Func ReverseBits4(Value:Card) :Card;
2356: Func ReverseBits4(Value:Int) : Int;
2357: Func ReverseBits5(Value:Int64) : Int64;
2358: Func ReverseBytes(Value:Word) : Word;
2359: Func ReverseBytes1(Value:Smallint):Smallint;
```

```
Func ReverseBytes2(Value:Int): Int;
Func ReverseBytes3(Value:Card):Card;
Func ReverseBytes4(Value:Int64): Int64;
Func ReverseString(const AText :Str) :Str
Func ReverseDNSLookup(const IPAddrs:Str;DNSServer:Str;Timeout,Retries:Int;var HName:Str):Bool;
Func Revert : HResult
Func RGB(R,G,B: Byte): TColor;
Func RGB2BGR( const Color : TColor) : TColor
Func RGB2TColor( R, G, B : Byte) : TColor
Func RGBToWebColorName( RGB : Int) :Str
Func RGBToWebColorStr( RGB : Int) :Str
Func RgbToHtml( Value : TColor) :Str
Func HtmlToRgb(const Value:Str): TColor;
Func RightStr(const AStr :Str; Len : Int) :Str
Func RightStr(const AText : Ansistr; const ACount : Int): Ansistr;
Func RightStr(const AText : WideString; const ACount : Int): WideString;
Func ROL( AVal: LongWord; AShift : Byte): LongWord
Func ROR( AVal: LongWord; AShift : Byte): LongWord
Func RotatePoint(Point:TFloatPoint;const Center:TFloatPoint;const Angle:Float):TFloatPoint
Func RotatePoint(Point:TFloatPoint;const Center:TFloatPoint;const Angle:Double):TFloatPoint;
Func Round(e : Extended) : Longint;
Func Round64(e: extended): Int64;
Func RoundAt( const Value :Str; Position : SmallInt) :Str
type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
Func RoundTo(const AValue:Extende;const ADigit:TRoundToEXRangeExtended):Extended;;
Func SimpleRoundTo(const AValue:Extended;const ADigit: TRoundToRange): Extended;;
Func RoundFrequency( const Frequency : Int) : Int
Func RoundInt( Value : Int; StepSize : Int) : Int
Func RoundPoint( const X, Y : Double) : TPoint
Func RoundRect( const ALeft, ATop, ARight,ABottom :Double):TRect
Func RowCount : Int
Func RowRequest(const Row:OleVariant;RequestType:Int;var OwnerData:OleVariant):OleVariant
Func RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
Func RPos( const ASub, AIn :Str; AStart : Int) : Int
Func RRot( const Value : Byte; const Count : TBitRange) : Byte;
Func RRot1( const Value : Word; const Count : TBitRange) : Word;
Func RRot2( const Value : Int; const Count : TBitRange) : Int;
Func RunDLL32(const ModuleNa,FuncName,CmdLine:str;WaitForCompletion:Bool;CmdShow:Int):Bool
Func RunningProcessesList( const List : TStrings; FullPath :Bool) :Bool
Func RunByteCode(Bytecode: Ansistr; out RuntimeErrors: Ansistr):Bool;;
Func RunCompiledScript2(Bytecode: Ansistr;out RuntimeErrors:Ansistr): Bool;
Func S_AddBackSlash( const ADirName :Str) :Str
Func S_AllTrim( const cStr :Str) :Str
Func S_AtRepl( const cAT, cStr, cRepl :Str) :Str
Func S_Cut( const cStr :Str; const iLen : Int) :Str
Func S_DecryptCRC32( const crc :Str; StartKey, MultKey, AddKey : Int): Int
Func S_DirExists( const ADir :Str) :Bool
Func S_Empty( const cStr :Str) :Bool
Func S_EncryptCRC32(const crc:LongWORD;StartKey,MultKey,AddKey:Int):str
Func S_LargeFontsActive :Bool
Func S_LimitDigits( AValue : Extended; ANumDigits : Int) : Extended
Func S_LTrim( const cStr :Str) :Str
Func S_ReadNextTextLineFromStream( stream : TStream) :Str
Func S_RepeatChar( const iLen : Int; const AChar : Char) :Str
Func S_ReplFirst( const cAT, cStr, cRepl :Str) :Str
Func S_RoundDecimal( AValue : Extended; APlaces : Int) : Extended
Func S_RTrim( const cStr :Str) :Str
Func S_RTrimCopy( const cStr :Str; iPos, iLen : Int) :Str
//Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
Func S_ShellExecute(aFilename:str;aParameters:str;aCommand:TS_ShellExecuteCmd):str
Func S_Space( const iLen : Int) :Str
Func S_StrBlanks( const cStr :Str; const iLen : Int) :Str
Func S_StrBlanksCuttooLong( const cStr :Str; const iLen : Int) :Str
Func S_StrCRC32( const Text :Str) : LongWORD
Func S_StrDecrypt96(const InString:Str;StartKey,MultKey,AddKey:Int):str
Func S_StrEncrypt96(const InString:Str;StartKey,MultKey,AddKey:int):Str
Func S_StringtoUTF_8( const AString :Str) :Str
Func S_StrLBlanks( const cStr :Str; const iLen : Int) :Str
Func S_StrToReal(const cStr:Str; var R: Double):Bool
Func S_TokenEnd( cBuffer : PChar; lEmptyToken :Bool) :Bool
Func S_TokenNext( cBuffer : PChar; lEmptyToken :Bool):Str
Func S_UTF_8ToString( const AString :Str) :Str
Func S_WBox( const AText :Str) : Int
Func SameDate( const A, B : TDateTime) :Bool
Func SameDate(const A, B: TDateTime):Bool;
Func SameDateTime( const A, B : TDateTime) :Bool
Func SameDateTime(const A, B: TDateTime):Bool;
Func SameFileName( S1, S2 :Str) :Bool
Func SameText( S1, S2 :Str) :Bool
Func SameText(const S1:Str; const S2:Str):Bool)
Func SameTime( const A, B : TDateTime) :Bool
Func SameTime(const A, B: TDateTime):Bool;
Func SameValue(const A, B: Extended; Epsilon: Extended):Bool //overload;
Func SameValue1(const A, B: Double; Epsilon: Double):Bool //overload;
Func SameValue2(const A, B: Single; Epsilon: Single):Bool //overload;
Func SampleVariance( const X : TDynFloatArray) : Float
Func Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
Func Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
Func Sar2( const Value : Int; const Count : TBitRange) : Int;
```

```
2449: Func SaveToFile( const AFileName : TFileName) :Bool
2450: Func SaveAsExcelFile(AGrid:TStringGrid;ASheetName,AFileName:str;open:bool):Bool;
2451: Func SaveAsExcel(aGrid:TStringGrid;aSheetName,aFileName:str;openexcel:boolean):Bool;
2452: Func ScanF(const aformat:Str; const args: array of const):Str;
2453: Func SCREENTOCLIENT(POINT:TPOINT):TPOINT
2454: Func SearchBuf(Buf:PChar;BufLen:Int;SelStart,
      SelLength:Int;SearchString:str;Options:TStringSearchOptions):PChar
2455: Func SearchBuf2(Buf:str;SelStart,SelLength:Int;SearchString:str;Options:TStringSearchOptions):Int;
2456: Func SearchRecattr: Int;
2457: Func SearchRecExcludeAttr: Int;
2458: Func SearchRecFileSize64( const SearchRec: TSearchRec):Int64
2459: Func SearchRecname:Str;
2460: Func SearchRecsize: Int;
2461: Func SearchRecTime: Int;
2462: Func Sec( const X : Extended) : Extended
2463: Func Secant( const X : Extended) : Extended
2464: Func SecH( const X : Extended) : Extended
2465: Func SecondOf( const AValue : TDateTime) : Word
2466: Func SecondOfTheDay( const AValue : TDateTime) : LongWord
2467: Func SecondOfTheHour( const AValue : TDateTime) : Word
2468: Func SecondOfTheMinute( const AValue : TDateTime) : Word
2469: Func SecondOfTheMonth( const AValue : TDateTime) : LongWord
2470: Func SecondOfTheWeek( const AValue : TDateTime) : LongWord
2471: Func SecondOfTheYear( const AValue : TDateTime) : LongWord
2472: Func SecondsBetween( const ANow, AThen : TDateTime) : Int64
2473: Func SecondSpan( const ANow, AThen : TDateTime) : Double
2474: Func SectionExists( const Section :Str) :Bool
2475: Func Seek( const KeyValues : Variant; SeekOption : TSeekOption) :Bool
2476: Func Seek(dlibMove: Longint; dwOrigin: Longint; out libNewPosition : Largeint):HResult
2477: Func Seek(Offset:LongInt;Origin:Word):LongInt
2478: Func SelectDirectory(var Directory:str; Options:TSelectDirOpts;HelpCtx:Longint):Bool;
2479: Func SelectDirectory1(const Caption:str;const Root:WideString;var Directory:str;
      Options:TSelectDirExtOpts;Parent:TWinControl):Bool;
2480: Func SelectImage(var AFileName:str;const Extensions,Filter:str):Boolean
2481: Func SendAppMessage(Msg:Card; WParam, LParam: Longint): Longint
2482: Func SendBuf( var Buf, Count : Int) : Int
2483: Func SendCmd( const AOut :Str; const AResponse : SmallInt) : SmallInt;
2484: Func SendCmd1( const AOut :Str; const AResponse : array of SmallInt) : SmallInt;
2485: Func SendKey( AppName :Str; Key : Char) :Bool
2486: Func SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint):Bool;
2487: Func SendStream( AStream : TStream) :Bool
2488: Func SendStreamThenDrop( AStream : TStream) :Bool
2489: Func SendText( const S :Str) : Int
2490: Func SendSerialData(Data: TByteArray; DataSize:Card):Card
2491: Func SendSerialText(Data:Str):Card
2492: Func Sent :Bool
2493: Func ServicesFilePath:Str
2494: Func SetAlpha( const Color32 : TColor32; NewAlpha : Int) : TColor32
2495: Func SetBit( const Value:Byte; const Bit : TBitRange) : Byte;
2496: Func SetBit1(const Value:Shortint; const Bit : TBitRange) : Shortint;
2497: Func SetBit2(const Value:Smallint; const Bit : TBitRange) : Smallint;
2498: Func SetBit3(const Value:Word; const Bit : TBitRange) : Word;
2499: Func SetBit4(const Value:Card; const Bit : TBitRange) Cardinal;
2500: Func SetBit4(const Value:Int; const Bit : TBitRange) : Int;
2501: Func SetBit5(const Value:Int64; const Bit : TBitRange) : Int64;
2502: Func SetClipboard( NewClipboard : TClipboard) : TClipboard
2503: Func SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2504: Func SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2505: Func SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2506: Func SetColorRed( const Color : TColor; const Red : Byte) : TColor
2507: Func SetCurrentDir( Dir :Str) :Bool
2508: Func SetCurrentDir(const Dir:Str):Bool)
2509: Func SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2510: Func SetDirCreation( const DirName :Str; const DateTime : TDateTime):Bool
2511: Func SetDirLastAccess(const DirName:str;const DateTime: TDateTime) : Bool
2512: Func SetDirLastWrite(const DirName:str; const DateTime : TDateTime): Bool
2513: Func SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2514: Func SetEndOfFile(Handle: Int): LongBool; stdcall;
2515: Func SetEnvironmentVar( const Name, Value :Str) :Bool
2516: Func SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2517: Func SetFileCreation(const FileName:str; const DateTime : TDateTime): Bool
2518: Func SetFileLastAccess( const FileName:str; const DateTime:TDateTime): Bool
2519: Func SetFileLastWrite(const FileName:str;const DateTime:TDateTime):BoolFunc SetFileTimeStamp(const
      FileName:str;TimeStamp:Int):Bool
2520: Func SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2521: Func SetLocalTime( Value : TDateTime) :Bool
2522: Func SetPrecisionTolerance( NewTolerance : Float) : Float
2523: Func SetPrinter( NewPrinter : TPrinter) : TPrinter
2524: Func SetPrivilege(privilegeName:Str; enable:Bool):Bool;
2525: Func SetRGBValue( const Red, Green, Blue : Byte) : TColor
2526: Func SetSequence( S, Localizar, Substituir : shortstring): shortstring
2527: Func SetSize( libNewSize : Longint) : HResult
2528: Func SetUserObjectFullAccess( hUserObject : THandle) :Bool
2529: Func Sgn( const X : Extended) : Int
2530: Func SHA1(const fileName:Str):Str;
2531: Func SHA256(astr:Str; amode: char):Str)
2532: Func SHA512(astr:Str; amode: char):Str)
2533: Func ShareMemoryManager :Bool
2534: Func ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:str;ShowCmd:Int):Int;stdcall;
```

```
2535: Func Shellexecute2(hwnd: HWND; const FileName:Str):Int; stdcall;
2536: Func ShellExecute3(aFilename:str;aParameters:str;aCommand:TS_ShellExecuteCmd):str;
2537: Func SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE): TSHORTCUT
2538: Func SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT):Str
2539: Func ShortDateFormat:str;
2540: Func ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2541: Func ShortTimeFormat:Str;
2542: Func SHOWMODAL:Int
2543: Func
      ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:TColor;BW:Int;Title:str;
2544: Func ShowModalPanel(aPnl:TCustomPanel;Titl:str;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2545: Func ShowWindow(C1: HWND; C2: Int):Bool;
2546: Proc ShowMemory     //in Dialog
2547: Func ShowMemory2:Str;
2548: Func ShutDownOS :Bool
2549: Func Signe( const X, Y : Extended) : Extended
2550: Func Sign( const X : Extended) : Int
2551: Func Sin(e : Extended) : Extended;
2552: //Assuming trigonometric arguments in degrees //| Use instead
2553: PrintF('in deg: %.18f',[sin(degtorad(60.0))]);  //0.866025403784438647
2554: Func sinc( const x : Double) : Double
2555: Func SinJ( X : Float) : Float
2556: Func Size( const AFileName :Str) : Int
2557: Func SizeOf: Longint;
2558: Func SizeofResource( ModuleHandle : HMODULE; ResHandle: TResourceHandle): Int
2559: Func SlashSep(const Path, S:Str):Str
2560: Func SLNDepreciation( const Cost, Salvage : Extended; Life : Int) : Extended
2561: Func SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2562: Func SmallPoint(X, Y: Int): TSmallPoint)
2563: Func Soundex( const AText :Str; ALength : TSoundexLength) :Str
2564: Func SoundexCompare( const AText, AOther :Str; ALength : TSoundexLength) : Int
2565: Func SoundexInt( const AText :Str; ALength : TSoundexIntLength) : Int
2566: Func SoundexProc( const AText, AOther :Str) :Bool
2567: Func SoundexSimilar(const AText,AOther:str;ALength:TSoundexLength):Boolean
2568: Func SoundexWord( const AText :Str) : Word
2569: Func SourcePos : Longint
2570: Func SourcePos:LongInt
2571: Func Split0( Str :Str; const substr :Str) : TStringList
2572: Proc SplitNameValue( const Line :Str; var Name, Value :Str)
2573: Func SQLRequiresParams( const SQL : WideString) :Bool
2574: Func Sqr(e : Extended) : Extended;
2575: Func Sqrt(e : Extended) : Extended;
2576: Func StartIP :Str
2577: Func StartPan( WndHandle : THandle; AControl : TControl) :Bool
2578: Func StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2579: Func StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2580: Func StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2581: Func StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek:Word):TDateTime
2582: Func StartOfAYear( const AYear : Word) : TDateTime
2583: Func StartOfTheDay( const AValue : TDateTime) : TDateTime
2584: Func StartOfTheMonth( const AValue : TDateTime) : TDateTime
2585: Func StartOfTheWeek( const AValue : TDateTime) : TDateTime
2586: Func StartOfTheYear( const AValue : TDateTime) : TDateTime
2587: Func StartsStr( const ASubText, AText :Str) :Bool
2588: Func StartsText( const ASubText, AText :Str) :Bool
2589: Func StartsWith( const ANSIStr, APattern :Str) :Bool
2590: Func StartsWith( const str :Str; const sub :Str) :Bool
2591: Func StartsWithACE( const ABytes : TIdBytes) :Bool
2592: Func StatusString( StatusCode : Int) :Str
2593: Func StdDev( const Data : array of Double) : Extended
2594: Func Stop : Float
2595: Func StopCount( var Counter : TJclCounter) : Float
2596: Func StoreColumns :Bool
2597: Func StrAfter( const sString :Str; const sDelimiters :Str):Str;
2598: Func StrAfter1(const sString:str; const sDelimiters:str;out cDelimiter:char):str;
2599: Func StrAlloc( Size :Card) : PChar
2600: Func StrAlloc(Size:Card): PChar)
2601: Func StrBefore( const sString :Str; const sDelimiters :Str) :Str;
2602: Func StrBefore1(const sString:str;const sDelimiters:str;out cDelimiter:char):str;
2603: Func StrBufSize( Str : PChar) :Card
2604: Func StrBufSize(const Str: PChar):Card)
2605: Func StrByteType( Str : PChar; Index :Card) : TMbcsByteType
2606: Func StrByteType(Str: PChar; Index:Card): TMbcsByteType)
2607: Func StrCat( Dest : PChar; Source : PChar) : PChar
2608: Func StrCat(Dest: PChar; const Source: PChar): PChar)
2609: Func StrCharLength( Str : PChar) : Int
2610: Func StrComp( Str1, Str2 : PChar) : Int
2611: Func StrComp(const Str1: PChar; const Str2: PChar): Int)
2612: Func StrCopy( Dest : PChar; Source : PChar) : PChar
2613: Func StrCopy(Dest: PChar; const Source: PChar): PChar)
2614: Func Stream_to_Ansistr( Source : TStream) : Ansistr
2615: Func Stream_to_Base64( Source : TStream) : Ansistr
2616: Func Stream_to_decimalbytes( Source : TStream) :Str
2617: Func Stream2WideString( oStream : TStream) : WideString
2618: Func StreamtoAnsistr( Source : TStream) : Ansistr
2619: Func StreamToByte( Source : TStream) :Str
2620: Func StreamToDecimalbytes( Source : TStream) :Str
2621: Func StreamtoOrd( Source : TStream) :Str
```

```
2622: Func StreamToString( Source: TStream) :Str
2623: Func StreamToString2(Source: TStream) :Str
2624: Func StreamToString3(Source: TStream) :Str
2625: Func StreamToString4(Source: TStream) :Str
2626: Func StrECopy( Dest : PChar; Source : PChar) : PChar
2627: Func StrEmpty( const sString :Str) :Bool
2628: Func StrEnd( Str : PChar) : PChar
2629: Func StrEnd(const Str: PChar): PChar)
2630: Func StrFilter( const sString :Str; xValidChars : TCharSet) :Str
2631: Func StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2632: Func StrGet(var S :Str; I : Int) : Char;
2633: Func StrGet2(S :Str; I : Int) : Char;
2634: Func StrHasPrefix( const sString :Str; const sPrefix :Str) :Bool
2635: Func StrHasSuffix( const sString :Str; const sSuffix :Str) :Bool
2636: Func StrHtmlDecode( const AStr :Str) :Str
2637: Func StrHtmlEncode( const AStr :Str) :Str
2638: Func StrToBytes(const Value:Str): TBytes;
2639: Func StrIComp( Str1, Str2 : PChar) : Int
2640: Func StringOfChar(c : char;I : longInt) :Str;
2641: Func StringOfChar2( ch : WideChar; Count : Int) : WideString;
2642: Func StringPad(InputStr,FillChar:Str; StrLen:Int; StrJustify:Boolean):Str;
2643: Func StringRefCount(const s:Str): Int;
2644: Func StringReplace(S,OldPattern,NewPattern:str;Flags:TReplaceFlags):str
2645: Func JStringReplace(const S,OldPattern, NewPattern:Str; Flags : TReplaceFlags):str
2646: Func StringReplace(const SourceString,OldPattern,NewPattern:str;Flags:TReplaceFlags):str;
2647: Func StringRemove( const S, Pattern :Str; Flags : TReplaceFlags) :Str
2648: Func StringToBoolean( const Ps :Str) :Bool
2649: Func StringToColor(const S:Str): TColor)
2650: Func StringToCursor(const S:Str): TCursor;
2651: Func StringToGUID(const S:Str): TGUID)
2652: Func StringTokenizer( const str :Str; const delim :Str) : IStringTokenizer
2653: Func StringToStringArray( const str :Str; const delim:Str): TStringDynArray
2654: Func StringWidth( S :Str) : Int
2655: Func StrInternetToDateTime( Value :Str) : TDateTime
2656: Func StrIsDateTime( const Ps :Str) :Bool
2657: Func StrIsFloatMoney( const Ps :Str) :Bool
2658: Func StrIsInt( const S :Str) :Bool
2659: Func StrLCat( Dest : PChar; Source : PChar; MaxLen :Card) : PChar
2660: Func StrLComp( Str1, Str2 : PChar; MaxLen :Card) : Int
2661: Func StrLCopy( Dest : PChar; Source : PChar; MaxLen :Card) : PChar
2662: Func StrLen( Str : PChar) :Card
2663: Func StrLen(const Str: PChar):Card)
2664: Func StrLessPrefix( const sString :Str; const sPrefix:Str):Str
2665: Func StrLessSuffix( const sString :Str; const sSuffix:Str):Str
2666: Func StrLIComp( Str1, Str2 : PChar; MaxLen :Card) : Int
2667: Func StrLower( Str : PChar) : PChar
2668: Func StrMove( Dest : PChar; Source : PChar; Count :Card) : PChar
2669: Func StrMove(Dest: PChar; const Source: PChar; Count:Card): PChar)
2670: Func StrNew( Str : PChar) : PChar
2671: Func StrNew(const Str: PChar): PChar)
2672: Func StrNextChar( Str : PChar) : PChar
2673: Func StrPad( const sString:str;const sPad:str;const iLength:Int):Str
2674: Func StrParse( var sString :Str; const sDelimiters :Str) :Str;
2675: Func StrParse1(var sString:str;const sDelimiters:str;out cDelimiter:char) string;
2676: Func StrPas( Str : PChar) :Str
2677: Func StrPas(const Str: PChar):Str)
2678: Func StrPCopy( Dest : PChar; Source :Str) : PChar
2679: Func StrPCopy(Dest: PChar; const Source:Str): PChar)
2680: Func StrPLCopy( Dest : PChar; Source :Str; MaxLen :Card) : PChar
2681: Func StrPos( Str1, Str2 : PChar) : PChar
2682: Func StrScan(const Str: PChar; Chr: Char): PChar)
2683: Func StrRScan(const Str: PChar; Chr: Char): PChar)
2684: Func StrToBcd( const AValue :Str) : TBcd
2685: Func StrToBool( S :Str) :Bool
2686: Func StrToBoolDef( S :Str; Default :Bool) :Bool
2687: Func StrToCard( const AStr :Str) :Card
2688: Func StrToConv( AText :Str; out AType : TConvType) : Double
2689: Func StrToCurr( S :Str) : Currency;
2690: Func StrToCurr(const S:Str): Currency)
2691: Func StrToCurrDef( S :Str; Default : Currency) : Currency;
2692: Func StrToDate( S :Str) : TDateTime;
2693: Func StrToDate(const s:Str): TDateTime;
2694: Func StrToDateDef( S :Str; Default : TDateTime) : TDateTime;
2695: Func StrToDateTime( S :Str) : TDateTime;
2696: Func StrToDateTime(const S:Str): TDateTime)
2697: Func StrToDateTimeDef( S :Str; Default : TDateTime) : TDateTime;
2698: Func StrToDay( const ADay :Str) : Byte
2699: Func StrToFloat( S :Str) : Extended;
2700: Func StrToFloat(s:Str): Extended;
2701: Func StrToFloatDef( S :Str; Default : Extended) : Extended;
2702: Func StrToFloatDef(const S:Str; const Default: Extended): Extended)
2703: Func StrToFloat( S :Str) : Extended;
2704: Func StrToFloat2( S :Str; FormatSettings : TFormatSettings) : Extended;
2705: Func StrToFloatDef( S :Str; Default : Extended) : Extended;
2706: Func StrToFloatDef2(S:str;Default:Extended;FormatSettings:TFormatSettings):Extended;
2707: Func StrToCurr( S :Str) : Currency;
2708: Func StrToCurr2( S :Str; FormatSettings : TFormatSettings) : Currency;
2709: Func StrToCurrDef( S :Str; Default : Currency) : Currency;
2710: Func StrToCurrDef2(S:str; Default:Currency; FormatSettings:TFormatSettings):Currency;
```

```
Func StrToTime2( S :Str; FormatSettings : TFormatSettings): TDateTime;
Func StrToTimeDef( S :Str; Default : TDateTime) : TDateTime;
Func StrToTimeDef2(S:str; Default:TDateTime;FormatSettings:TFormatSettings):TDateTime;
Func TryStrToTime( S :Str; Value : TDateTime) :Bool;
Func StrToDateTime( S :Str) : TDateTime;
Func StrToDateTime2( S :Str; FormatSettings : TFormatSettings) : TDateTime;
Func StrToDateTimeDef( S :Str; Default : TDateTime) : TDateTime;
Func StrToFloatRegionalIndependent(aValue:str;aDecimalSymbol:Char;aDigitGroupSymbol:Char):Extended
Func StrToInt( S :Str) : Int
Func StrToInt(s:Str): Longint;
Func StrToInt64( S :Str) : Int64
Func StrToInt64(s:Str): int64;
Func StrToInt64Def( S :Str; Default : Int64) : Int64
Func StrToInt64Def(const S:Str; const Default: Int64):Int64)
Func StrToIntDef( S :Str; Default : Int) : Int
Func StrToIntDef(const S:Str; Default: Int): Int)
Func StrToIntDef(s:Str; def: Longint): Longint;
Func StrToMonth( const AMonth :Str) : Byte
Func StrToTime( S :Str) : TDateTime;
Func StrToTime(const S:Str): TDateTime)
Func StrToTimeDef( S :Str; Default : TDateTime) : TDateTime;
Func StrToWord( const Value :Str) : Word
Func StrToXmlDate( const DateStr :Str; const Format :Str) :Str
Func StrToXmlDateTime( const DateStr :Str; const Format :Str) :Str
Func StrToXmlTime( const TimeStr :Str; const Format :Str) :Str
Func StrUpper( Str : PChar) : PChar
Func StuffString(const AText:str;AStart, ALength:Card;const ASubText:str):str
Func Sum( const Data : array of Double) : Extended
Func SumFloatArray( const B : TDynFloatArray) : Float
Func SumInt( const Data : array of Int) : Int
Func SumOfSquares( const Data : array of Double) : Extended
Func SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
Func SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
Func SumSquareFloatArray( const B : TDynFloatArray) : Float
Func Supports( CursorOptions : TCursorOptions) :Bool
Func SupportsClipboardFormat( AFormat : Word) :Bool
Func SwapWord(w : word): word)
Func SwapInt(i : Int): Int)
Func SwapLong(L : longint): longint)
Func Swap(i : Int): Int)
Func SYDDepreciation(const Cost,Salvage:Extended;Life,Period:Int): Extended
Func SyncTime :Bool
Func SysErrorMessage( ErrorCode : Int) :Str
Func SysErrorMessage(ErrorCode: Int):Str)
Func SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
Func SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
Func SysStringLen(const S: WideString): Int; stdcall;
Func TabRect( Index : Int) : TRect
Func Tan( const X : Extended) : Extended
Func TaskMessageDlg(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint):Int;
Func TaskMessageDlg1(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;
HelpCtx:Longint;DefaultButton:TMsgDlgBtn):Int;
Func TaskMessageDlgPos(const Title,Msg:str;DlgType:TMsgDlgType;Bts:TMsgDlgButtons;HelpCtx:Longint;X,
Y:Int):Int;
Func TaskMessageDlgPos1(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
Y:Int;DefaultButton:TMsgDlgBtn):Int;
Func TaskMessageDlgPosHelp(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:
TMsgDlgButtons;HelpCtx:Longint;X,Y:Int;const HelpFileName:str): Int;
Func TaskMessageDlgPosHelp1(const Title, Msg:str;DlgType: TMsgDlgType; Buttons :
TMsgDlgButtons;HelpCtx:Longint;X,Y:Int;const HelpFileName:str;DefaultButton:TMsgDlgBtn):Int;
Func TenToY( const Y : Float) : Float
Func TerminateApp( ProcessID : DWORD; Timeout : Int) : TJclTerminateAppResult
Func TerminateTask( Wnd : HWND; Timeout : Int) : TJclTerminateAppResult
Func TestBit( const Value:Byte; const Bit : TBitRange) : Bool;
Func TestBit2(const Value:Shortint; const Bit : TBitRange): Bool;
Func TestBit3(const Value:Smallint; const Bit : TBitRange): Bool;
Func TestBit4(const Value:Word; const Bit : TBitRange):Bool;
Func TestBit5(const Value:Card; const Bit : TBitRange): Bool;
Func TestBit6(const Value:Int; const Bit : TBitRange): Bool;
Func TestBit7(const Value:Int64; const Bit : TBitRange): Bool;
Func TestBits(const Value, Mask:Byte) :Bool;
Func TestBits1(const Value,Mask:Shortint) :Bool;
Func TestBits2(const Value,Mask:Smallint) :Bool;
Func TestBits3(const Value,Mask:Word) :Bool;
Func TestBits4(const Value,Mask:Card) :Bool;
Func TestBits5(const Value,Mask:Int) :Bool;
Func TestBits6(const Value,Mask:Int64) :Bool;
Func TestFDIVInstruction :Bool
Func TestStreamFormat(Stream: TStream): TStreamOriginalFormat
Func TextExtent( const Text :Str) : TSize
Func TextHeight(Text:Str): Int;
Func TextIsSame( const A1 :Str; const A2 :Str) :Bool
Func TextStartsWith( const S, SubS :Str) :Bool
Func TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue):Boolean)
Func ConvInt(i : Int):str;
Func IntToText(i : Int):str;
Func TEXTTOSHORTCUT( TEXT :Str) : TSHORTCUT
Func TextWidth(Text:Str): Int;
Func ThreadCount : Int
```

```
2795: Func ThousandSeparator: char;
2796: Func Ticks :Card
2797: Func Time : TDateTime
2798: Func Time: TDateTime;
2799: Func TimeGetTime: int64;
2800: Func TimeOf( const AValue:TDateTime):TDateTime
2801: Func TimeSeparator: char;
2802: Func TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2803: Func TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2804: Func TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2805: Func TimeToStr( DateTime : TDateTime) :Str;
2806: Func TimeToStr(const DateTime: TDateTime):Str;
2807: Func TimeZoneBias : TDateTime
2808: Func ToCommon( const AValue : Double) : Double
2809: Func ToCommon(const AValue: Double): Double;
2810: Func Today : TDateTime
2811: Func ToggleBit( const Value:Byte; const Bit : TBitRange) : Byte;
2812: Func ToggleBit1(const Value:Shortint; const Bit : TBitRange): Shortint;
2813: Func ToggleBit2(const Value:Smallint; const Bit : TBitRange): Smallint;
2814: Func ToggleBit3(const Value:Word; const Bit : TBitRange) : Word;
2815: Func ToggleBit4(const Value:Card; const Bit : TBitRange):Card;
2816: Func ToggleBit5(const Value:Int; const Bit : TBitRange) : Int;
2817: Func ToggleBit6(const Value:Int64; const Bit : TBitRange) : Int64;
2818: Func TokenComponentIdent:str
2819: Func TokenFloat : Extended
2820: Func TokenFloat:Extended
2821: Func TokenInt : Longint
2822: Func TokenInt:LongInt
2823: Func TokenString :Str
2824: Func TokenString:str
2825: Func TokenSymbolIs( const S :Str) :Bool
2826: Func TokenSymbolIs(S:str):Boolean
2827: Func Tomorrow : TDateTime
2828: Func ToRightOf( const pc : TControl; piSpace : Int) : Int
2829: Func ToString :Str
2830: Func TotalVariance( const Data : array of Double) : Extended
2831: Func Trace2( AURL :Str) :Str;
2832: Func TrackMenu( Button : TToolButton) :Bool
2833: Func TRANSLATE( SRC, DEST : PCHAR; TOOEM :Bool) : Int
2834: Func TranslateURI( const URI :Str) :Str
2835: Func TranslationMatchesLanguages( Exact :Bool) :Bool
2836: Func TransparentStretchBlt(DstDC:HDC;DstX,DstY,DstW,DstH:Int;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Int;MaskDC:HDC;MaskX, MaskY:Int):Bool
2837: Func Trim( S :Str) :Str;
2838: Func Blank( S :Str) :Str; //alias to Trim
2839: Func Trim( S : WideString) : WideString;
2840: Func Trim(s : AnyString) : AnyString;
2841: Func TrimAllOf( ATrim, AText :Str):str
2842: Func TrimLeft( S :Str) :Str;
2843: Func TrimLeft( S : WideString) : WideString;
2844: Func TrimLeft(const S:Str):Str)
2845: Func TrimRight( S:Str) :Str;
2846: Func TrimRight( S: WideString) : WideString;
2847: Func TrimRight(const S:Str):Str)
2848: Func TrueBoolStrs: array of string
2849: Func Trunc(e : Extended) : Longint;
2850: Func Trunc64(e: extended): Int64;
2851: Func TruncPower( const Base, Exponent : Float) : Float
2852: Func TryConvTypeToFamily(const AFrom,ATo:TConvType; out AFamily:TConvFamily):Bool;
2853: Func TryConvTypeToFamily1(const AType:TConvType;out AFamily:TConvFamily):Bool;
2854: Func TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime):Bool;
2855: Func TryEncodeDateDay(const AYear,ADayOfYear:Word out AValue:TDateTime): Bool
2856: Func TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime):Bool
2857: Func TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word;out AValue:TDateTime):Bool
2858: Func TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word):Bool
2859: Func TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out AVal:TDateTime):Bool
2860: Func TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime):Bool;
2861: Func TryFloatToDateTime( Value : Extended; AResult : TDateTime) :Bool
2862: Func TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) :Bool
2863: Func TryLock :Bool
2864: Func TryModifiedJulianDateToDateTime(const AValue:Double;out ADateTime:TDateTime):Boolean
2865: Func TryRecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHour,AMinute,ASecond,
      AMilliSecond:Word; out AResult:TDateTime):Boolean
2866: Func TryStrToBcd( const AValue :Str; var Bcd : TBcd) :Bool
2867: Func TryStrToConv( AText :Str; out AValue : Double; out AType : TConvType) :Bool
2868: Func TryStrToDate( S :Str; Value : TDateTime) :Bool;
2869: Func TryStrToDateTime( S :Str; Value : TDateTime) :Bool;
2870: Func TryStrToTime( S :Str; Value : TDateTime) :Bool;
2871: Func TryStrToInt(const S: Ansistr; var I: Int):Bool;
2872: Func TryStrToInt64(const S: Ansistr; var I: Int64):Bool;
2873: Func TryStrToBool(const S:Str; out Value:Bool):Bool;
2874: Func TwoByteToWord( AByte1, AByte2 : Byte) : Word
2875: Func TwoCharToWord( AChar1, AChar2 : Char) : Word
2876: Func TwoToY( const Y : Float) : Float
2877: Func UCS4StringToWideString( const S : UCS4String) : WideString
2878: Func UIDL( const ADest : TStrings; const AMsgNum : Int) :Bool
2879: Func Unassigned: Variant;
2880: Func UndoLastChange( FollowChange :Bool) :Bool
2881: Func UniCodeToStr(Value:Str):Str;
```

```
2882: Func UnionRect( out Rect : TRect; const R1, R2 : TRect) :Bool
2883: Func UnionRect(out Rect: TRect; const R1, R2: TRect):Bool)
2884: Func UnixDateTimeToDelphiDateTime( UnixDateTime :Card) : TDateTime
2885: Func UnixPathToDosPath( const Path :Str) :Str
2886: Func UnixToDateTime( const AValue : Int64) : TDateTime
2887: Func UnixToDateTime(U: Int64): TDateTime;
2888: Func UnlockRegion(libOffset Longint;cb: Largeint;dwLockType:Longint):HResult
2889: Func UnlockResource( ResData : HGLOBAL) : LongBool
2890: Func UnlockVolume( var Handle : THandle) :Bool
2891: Func UnMaskString( Mask, Value :Str) :Str
2892: Func UpCase(ch : Char ) : Char;
2893: Func UpCaseFirst( const AStr :Str) :Str
2894: Func UpCaseFirstWord( const AStr :Str) :Str
2895: Func UpdateAction( Action : TBasicAction) :Bool
2896: Func UpdateKind : TUpdateKind
2897: Func UPDATESTATUS : TUPDATESTATUS
2898: Func UpperCase( S :Str) :Str
2899: Func Uppercase(s : AnyString) : AnyString;
2900: Func URLDecode( ASrc :Str) :Str
2901: Func URLEncode( const ASrc :Str) :Str
2902: Func UseRightToLeftAlignment :Bool
2903: Func UseRightToLeftAlignmentForField(const AField:TField;Alignment:TAlignment):Bool
2904: Func UseRightToLeftReading :Bool
2905: Func UTF8CharLength( Lead : Char) : Int
2906: Func UTF8CharSize( Lead : Char) : Int
2907: Func UTF8Decode( const S : UTF8String) : WideString
2908: Func UTF8Encode( const WS : WideString) : UTF8String
2909: Func UTF8LowerCase( const S : UTF8string) : UTF8string
2910: Func Utf8ToAnsi( const S : UTF8String) :Str
2911: Func Utf8ToAnsiEx( const S : UTF8String; const cp : Int) :Str
2912: Func UTF8UpperCase( const S : UTF8string) : UTF8string
2913: Func ValidFieldIndex( FieldIndex : Int) :Bool
2914: Func ValidParentForm(control: TControl): TForm
2915: Func Value : Variant
2916: Func ValueExists( const Section, Ident :Str) :Bool
2917: Func ValueOf( const Key :Str) : Int
2918: Func ValueInSet(AValue: Variant; ASet: Variant):Bool;
2919: Func VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2920: Func VarArrayFromStrings( Strings : TStrings) : Variant
2921: Func VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2922: Func VarArrayGet(var S : Variant; I : Int) : Variant;
2923: Func VarFMTBcd : TVarType
2924: Func VarFMTBcdCreate1 : Variant;
2925: Func VarFMTBcdCreate2(const AValue :Str; Precision, Scale : Word) : Variant;
2926: Func VarFMTBcdCreate3(const AValue: Double;Precision: Word; Scale: Word):Variant;
2927: Func VarFMTBcdCreate4(const ABcd : TBcd) : Variant;
2928: Func Variance(const Data : array of Double) : Extended
2929: Func VariantAdd2( const V1 : Variant; const V2 : Variant):Variant
2930: Func VariantAnd2( const V1 : Variant; const V2 : Variant):Variant
2931: Func VariantDiv2( const V1 : Variant; const V2 : Variant):Variant
2932: Func VariantGetElement( const V : Variant; i1 : Int) : Variant;
2933: Func VariantGetElement1( const V : Variant; i1, i2 : Int) : Variant;
2934: Func VariantGetElement2( const V : Variant; i1, i2, i3 : Int) : Variant;
2935: Func VariantGetElement3( const V : Variant; i1, i2, i3, i4 : Int):Variant;
2936: Func VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5: Int):Variant;
2937: Func VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2938: Func VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2939: Func VariantNeg(const V1: Variant) : Variant
2940: Func VariantNot(const V1: Variant) : Variant
2941: Func VariantOr2(const V1: Variant; const V2 : Variant) :Variant
2942: Func VariantShl2(const V1: Variant; const V2 : Variant):Variant
2943: Func VariantShr2(const V1: Variant; const V2 : Variant):Variant
2944: Func VariantSub2(const V1: Variant; const V2 : Variant):Variant
2945: Func VariantXor2(const V1: Variant; const V2 : Variant):Variant
2946: Func VarIsEmpty(const V: Variant):Bool;
2947: Func VarIsFMTBcd( const AValue : Variant) :Bool;
2948: Func VarIsNull(const V: Variant):Bool;
2949: Func VarToBcd( const AValue : Variant) : TBcd
2950: Func VarType(const V: Variant): TVarType;
2951: Func VarType( const V : Variant) : TVarType
2952: Func VarAsType( const V : Variant; AVarType : TVarType): Variant
2953: Func VarIsType( const V : Variant; AVarType : TVarType):Bool;
2954: Func VarIsType1(const V: Variant;const AVarTypes: array of TVarType):Bool;
2955: Func VarIsByRef( const V : Variant) : Bool
2956: Func VarIsEmpty( const V : Variant) : Bool
2957: Proc VarCheckEmpty( const V : Variant)
2958: Func VarIsNull( const V : Variant) : Bool
2959: Func VarIsClear( const V : Variant) : Bool
2960: Func VarIsCustom( const V : Variant) : Bool
2961: Func VarIsOrdinal( const V : Variant) : Bool
2962: Func VarIsFloat( const V : Variant) : Bool
2963: Func VarIsNumeric( const V : Variant) : Bool
2964: Func VarIsStr( const V : Variant) :Bool
2965: Func VarToStr( const V : Variant) :Str
2966: Func VarToStrDef( const V : Variant; const ADefault :Str) :Str
2967: Func VarToWideStr( const V : Variant) : WideString
2968: Func VarToWideStrDef( const V : Variant; const ADefault : WideString):WideString
2969: Func VarToDateTime( const V : Variant) : TDateTime
2970: Func VarFromDateTime( const DateTime : TDateTime) : Variant
```

```
2971: Func VarInRange( const AValue, AMin, AMax : Variant) :Bool
2972: Func VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2973: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual)
2974: Func VarSameValue( const A, B : Variant) :Bool
2975: Func VarCompareValue( const A, B : Variant) : TVariantRelationship
2976: Func VarIsEmptyParam( const V : Variant) :Bool
2977: Func VarIsError( const V : Variant; out AResult : HRESULT):Boolean;
2978: Func VarIsError1( const V : Variant) :Bool;
2979: Func VarAsError( AResult : HRESULT) : Variant
2980: Proc VarCopyNoInd( var Dest : Variant; const Source : Variant)
2981: Func VarIsArray( const A : Variant) :Bool;
2982: Func VarIsArray1( const A : Variant; AResolveByRef :Bool) :Bool;
2983: Func VarArrayCreate( const Bounds : array of Int; AVarType : TVarType):Variant
2984: Func VarArrayOf( const Values : array of Variant) : Variant
2985: Func VarArrayRef( const A : Variant) : Variant
2986: Func VarTypeIsValidArrayType( const AVarType : TVarType):Bool
2987: Func VarTypeIsValidElementType( const AVarType : TVarType):Boolean
2988: Func VarArrayDimCount( const A : Variant) : Int
2989: Func VarArrayLowBound( const A : Variant; Dim : Int) : Int
2990: Func VarArrayHighBound( const A : Variant; Dim : Int) : Int
2991: Func VarArrayLock( const A : Variant) : ___Pointer
2992: Proc VarArrayUnlock( const A : Variant)
2993: Func VarArrayGet( const A : Variant; const Indices : array of Int) : Variant
2994: Proc VarArrayPut(var A:Variant;const Value:Variant;const Indices: array of Int)
2995: Proc DynArrayToVariant(var V: Variant;const DynArray: ___Pointer;TypeInfo: ___Pointer)
2996: Proc DynArrayFromVariant(var DynArray:___Pointer;const V: Variant;TypeInfo:___Pointer)
2997: Func Unassigned : Variant
2998: Func Null : Variant
2999: Func VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
3000: Func VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
3001: Func VectorDot( const V1, V2 : TFloatPoint) : Double
3002: Func VectorDot(const V1,V2: TFloatPoint): Double;
3003: Func VectorLengthSqr( const V : TFloatPoint) : Double
3004: Func VectorLengthSqr(const V: TFloatPoint): Double;
3005: Func VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
3006: Func VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
3007: Func VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
3008: Func VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
3009: Func Verify( AUserName :Str) :Str
3010: Func Versine( X : Float) : Float
3011: Func VersionCheck:Bool;
3012: Func VersionCheckAct:Str;
3013: Func VersionLanguageId( const LangIdRec : TLangIdRec) :Str
3014: Func VersionLanguageName( const LangId : Word) :Str
3015: Func VersionResourceAvailable( const FileName :Str):Boolean
3016: Func Visible :Bool
3017: Func VolumeID(DriveChar: Char):Str
3018: Func WaitFor( const AString :Str) :Str
3019: Func WaitFor( const TimeOut :Card): TJclWaitResult
3020: Func WaitFor1 : TWaitResult;
3021: Func WaitForData( Timeout : Longint) :Bool
3022: Func WebColorNameToColor( WebColorName :Str) : TColor
3023: Func WebColorStrToColor( WebColor :Str) : TColor
3024: Func WebColorToRGB( WebColor : Int) : Int
3025: Func wGet(aURL, afile:Str):Bool;'
3026: Func wGet2(aURL, afile:Str):Bool;'  //without file open
3027: Func wGetX(aURL, afile:Str):Bool;
3028: Func wGetX2(aURL, afile:Str):Bool;'  //without file open
3029: Func WebGet(aURL, afile:Str):Bool;'
3030: Func WebExists:Bool;   //alias to isinternet
3031: Func WeekOf( const AValue : TDateTime) : Word
3032: Func WeekOfTheMonth( const AValue : TDateTime) : Word;
3033: Func WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
3034: Func WeekOfTheYear( const AValue : TDateTime) : Word;
3035: Func WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
3036: Func WeeksBetween( const ANow, AThen : TDateTime) : Int
3037: Func WeeksInAYear( const AYear : Word) : Word
3038: Func WeeksInYear( const AValue : TDateTime) : Word
3039: Func WeekSpan( const ANow, AThen : TDateTime): Double
3040: Func WideAdjustLineBreaks( const S : WideString; Style:TTextLineBreakStyle):WideString
3041: Func WideCat( const x, y : WideString) : WideString
3042: Func WideCompareStr( S1, S2 : WideString) : Int
3043: Func WideCompareStr(const S1: WideString; const S2: WideString): Int)
3044: Func WideCompareText( S1, S2 : WideString) : Int
3045: Func WideCompareText(const S1: WideString; const S2: WideString): Int)
3046: Func WideCopy( const src : WideString; index, count : Int) : WideString
3047: Func WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
3048: Func WideEqual( const x, y : WideString) :Bool
3049: Func WideFormat(const Format: WideString; const Args: array of const):WideString)
3050: Func WideGreater( const x, y : WideString) :Bool
3051: Func WideLength( const src : WideString) : Int
3052: Func WideLess( const x, y : WideString) :Bool
3053: Func WideLowerCase( S : WideString) : WideString
3054: Func WideLowerCase(const S: WideString): WideString)
3055: Func WidePos( const src, sub : WideString) : Int
3056: Func WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
3057: Func WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
3058: Func WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
3059: Func WideSameStr( S1, S2 : WideString) :Bool
```

```
3060: Func WideSameStr(const S1: WideString; const S2: WideString):Bool)
3061: Func WideSameText( S1, S2 : WideString) :Bool
3062: Func WideSameText(const S1: WideString; const S2: WideString):Bool)
3063: Func WideStringReplace(const S,OldPattern,NewPattern:Widestr;Flags:TReplaceFlags):Widestr
3064: Func WideStringToUCS4String( const S : WideString) : UCS4String
3065: Func WideUpperCase( S : WideString) : WideString
3066: Func Win32BackupFile( const FileName :Str; Move :Bool) :Bool
3067: Func Win32Check(RetVal:Bool):Bool)
3068: Func Win32DeleteFile(const FileName:str; MoveToRecycleBin:Boolean):Bool
3069: Func Win32RestoreFile( const FileName :Str) :Bool
3070: Func Win32Type : TIdWin32Type
3071: Func WinColor( const Color32 : TColor32) : TColor
3072: Func winexec(FileName: pchar; showCmd: Int): Int;
3073: Func WinExec32( const Cmd :Str; const CmdShow : Int) :Bool
3074: Func WinExec32AndWait( const Cmd :Str; const CmdShow : Int) :Card
3075: Func WithinPastDays( const ANow, AThen : TDateTime; const ADays : Int) : Bool
3076: Func WithinPastHours(const ANow, AThen:TDateTime; const AHours : Int64): Bool
3077: Func WithinPastMilliSeconds(const ANow,AThen:TDateTime;const AMilliSeconds:Int64):Bool
3078: Func WithinPastMinutes(const ANow,AThen:TDateTime;const AMinutes:Int64):Bool
3079: Func WithinPastMonths(const ANow, AThen:TDateTime; const AMonths:Int): Bool
3080: Func WithinPastSeconds(const ANow,AThen: DateTime;const ASeconds:Int64): Bool
3081: Func WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Int): Bool
3082: Func WithinPastYears( const ANow, AThen : TDateTime; const AYears : Int): Bool
3083: Func WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar): DWORD
3084: Func WordToStr( const Value : Word) :Str
3085: Func WordGridFormatIdentToInt(const Ident:Str; var Value:Longint): Bool
3086: Func IntToWordGridFormatIdent( Value : Longint; var Ident :Str): Bool
3087: Proc GetWordGridFormatValues( Proc : TGetStrProc)
3088: Func WorkArea : Int
3089: Func WrapText( Line :Str; MaxCol : Int) :Str;
3090: Func WrapText2(Line, BreakStr:str;BreakChars:TSysCharSet;MaxCol:Int):str;
3091: Func Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3092: Func Write(Buffer:str;Count:LongInt):LongInt
3093: Func WriteClient( var Buffer, Count : Int) : Int
3094: Func WriteFile(const AFile:str; const AEnableTransferFile:Boolean):Card
3095: Func WriteHeaders(StatusCode: Int; const ReasonString, Headers:Str):Boolean
3096: Func WriteString( const AString :Str): Bool
3097: Func WStrGet(var S : AnyString; I : Int): WideChar;
3098: Func wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Int
3099: Func wsprintf( Output : PChar; Format : PChar) : Int
3100: Func XmlDateTimeToStr( const XmlDateTime :Str; const Format:Str):str
3101: Func XmlTimeToStr( const XmlTime :Str; const Format :Str):Str
3102: Func XorDecode( const Key, Source :Str) :Str
3103: Func XorEncode( const Key, Source :Str) :Str
3104: Func XorString( const Key, Src : ShortString): ShortString
3105: Func Yield : Bool
3106: Func YearOf( const AValue : TDateTime) : Word
3107: Func YearsBetween( const ANow, AThen : TDateTime) : Int
3108: Func YearSpan( const ANow, AThen : TDateTime) : Double
3109: Func Yesterday : TDateTime
3110: Func YesNoDialog(const ACaption, AMsg:Str):Bool;
3111: Function(const Name :Str; Proc : TUserFunction)
3112: Func using Special_Scholz from V 3.8.5.0
3113: Func TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3114: Func FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3115: Func FloatToTime2Dec(value:Extended):Extended;
3116: Func MinToStd(value:Extended):Extended;
3117: Func MinToStdAsString(value:Extended):str;
3118: Func RoundFloatToStr(zahl:Extended; decimals:Int):str;
3119: Func RoundFloat(zahl:Extended; decimals:Int):Extended;
3120: Func Round2Dec (zahl:Extended):Extended;
3121: Func GetAngle(x,y:Extended):Double;
3122: Func AddAngle(a1,a2:Double):Double;
3123:
3124: ********************************************************************
3125: unit uPSI_StText;
3126: ********************************************************************
3127: Func TextSeek( var F : TextFile; Target : LongInt) :Bool
3128: Func TextFileSize( var F : TextFile) : LongInt
3129: Func TextPos( var F : TextFile) : LongInt
3130: Func TextFlush( var F : TextFile) :Bool
3131:
3132: ********************************************************************
3133: from  JvVCLUtils;
3134: ********************************************************************
3135: { Windows resources (bitmaps and icons) VCL-oriented routines }
3136: Proc DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Int;Bitmap:TBitmap;TransparentColor:TColor);
3137: Proc DrawBitmapRectTransparent(Dest: TCanvas;DstX,
      DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3138: Proc StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Int; SrcRect: TRect; Bitmap:
      TBitmap; TransparentColor:TColor);
3139: Func MakeBitmap(ResID: PChar): TBitmap;
3140: Func MakeBitmapID(ResID: Word): TBitmap;
3141: Func MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3142: Func CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3143: Func CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3144: Func CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3145:   HighlightColor, ShadowColor: TColor; DrawHighlight:Bool): TBitmap;
3146: Func CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
```

```
3147: Func ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3148: Proc AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Int);
3149: {$IFDEF WIN32}
3150: Proc ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3151:   X, Y, Index: Int; HighlightColor, GrayColor: TColor; DrawHighlight:Bool);
3152: {$ENDIF}
3153: Func MakeIcon(ResID: PChar): TIcon;
3154: Func MakeIconID(ResID: Word): TIcon;
3155: Func MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3156: Func CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3157: {$IFDEF WIN32}
3158: Func CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3159: {$ENDIF}
3160: { Service routines }
3161: Proc NotImplemented;
3162: Proc ResourceNotFound(ResID: PChar);
3163: Func PointInRect(const P: TPoint; const R: TRect):Bool;
3164: Func PointInPolyRgn(const P: TPoint; const Points: array of TPoint):Bool;
3165: Func PaletteColor(Color: TColor): Longint;
3166: Func WidthOf(R: TRect): Int;
3167: Func HeightOf(R: TRect): Int;
3168: Proc PaintInverseRect(const RectOrg, RectEnd: TPoint);
3169: Proc DrawInvertFrame(ScreenRect: TRect; Width: Int);
3170: Proc CopyParentImage(Control: TControl; Dest: TCanvas);
3171: Proc Delay(MSecs: Longint);
3172: Proc DeleteLine(StrList: TStringList; SearchPattern:Str);
3173: Proc CenterControl(Control: TControl);
3174: Func PaletteEntries( Palette : HPALETTE) : Int
3175: Func WindowClassName( Wnd : HWND) :Str
3176: Func ScreenWorkArea : TRect
3177: Proc MoveWindowOrg( DC : HDC; DX, DY : Int)
3178: Proc SwitchToWindow( Wnd : HWND; Restore :Bool)
3179: Proc ActivateWindow( Wnd : HWND)
3180: Proc ShowWinNoAnimate( Handle : HWND; CmdShow : Int)
3181: Proc CenterWindow( Wnd : HWND)
3182: Proc ShadeRect( DC : HDC; const Rect : TRect)
3183: Proc KillMessage( Wnd : HWND; Msg :Card)
3184: Func DialogsToPixelsX(Dlgs:Word):Word
3185: Func DialogsToPixelsY(Dlgs:Word):Word
3186: Func PixelsToDialogsX(Pixs:Word):Word
3187: Func PixelsToDialogsY(Pixs:Word):Word
3188: {$IFDEF WIN32}
3189: Proc ShowMDIClientEdge(ClientHandle: THandle; ShowEdge:Bool);
3190: Func MakeVariant(const Values: array of Variant): Variant;
3191: {$ENDIF}
3192: Func CreateRotatedFont(Font: TFont; Angle: Int): HFONT;
3193: Func MsgBox(const Caption, Text:Str; Flags: Int): Int;
3194: Func MsgDlg(const Msg:str;AType:TMsgDlgType;AButtons:TMsgDlgButtons;HelpCtx:Longint):Word;
3195: {$IFDEF CBUILDER}
3196: Func FindPrevInstance(const MainFormClass: ShortString; const ATitle:Str): HWND;
3197: Func ActivatePrevInstance(const MainFormClass: ShortString; const ATitle:Str):Boolean;
3198: {$ELSE}
3199: Func FindPrevInstance(const MainFormClass, ATitle:Str): HWND;
3200: Func ActivatePrevInstance(const MainFormClass, ATitle:Str):Bool;
3201: {$ENDIF CBUILDER}
3202: Func IsForegroundTask:Bool;
3203: Proc MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show:Bool);
3204: Func GetAveCharSize(Canvas: TCanvas): TPoint;
3205: Func MinimizeText(const Text:Str; Canvas: TCanvas; MaxWidth: Int):Str;
3206: Proc FreeUnusedOle;
3207: Proc Beep;
3208: Func GetWindowsVersionJ:Str;
3209: Func LoadDLL(const LibName:Str): THandle;
3210: Func RegisterServer(const ModuleName:Str):Bool;
3211: {$IFNDEF WIN32}
3212: Func IsLibrary:Bool;
3213: {$ENDIF}
3214: { Gradient filling routine }
3215: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3216: Proc GradientFillRect(Canvas:TCanvas; ARect:TRect;StartColor,
      EndColor:TColor;Direction:TFillDirection;Colors: Byte);
3217: { String routines }
3218: Func GetEnvVar(const VarName:Str):Str;
3219: Func AnsiUpperFirstChar(const S:Str):Str;
3220: Func StringToPChar(var S:Str): PChar;
3221: Func StrPAlloc(const S:Str): PChar;
3222: Proc SplitCommandLine(const CmdLine:Str; var ExeName,Params:str);
3223: Func DropT(const S:Str):Str;
3224: { Memory routines }
3225: Func AllocMemo(Size: Longint): Pointer;
3226: Func ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3227: Proc FreeMemo(var fpBlock: Pointer);
3228: Func GetMemoSize(fpBlock: Pointer): Longint;
3229: Func CompareMem(fpBlock1, fpBlock2: Pointer; Size:Card):Bool;
3230: {$IFNDEF COMPILER5_UP}
3231: Proc FreeAndNil(var Obj);
3232: {$ENDIF}
3233: // from PNGLoader
3234: Func OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Int;TransparentColor:TColor):Int
```

```
3235: Proc TransformRGB2LOCO( Image : TLinearBitmap)
3236: Proc TransformLOCO2RGB( Image : TLinearBitmap)
3237: Proc SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3238: Func DrawButtonFace(Canvas:TCanvas;const Client:TRect;BevelWidth:Int;Style:TButtonStyle; IsRounded,IsDown,
      IsFocused:Boolean):TRect  //TButtons
3239:  Func IsAnAllResult( const AModalResult : TModalResult) :Bool
3240:  Func InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam: Longint):Longint
3241:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3242:  //Proc ChangeBiDiModeAlignment( var Alignment : TAlignment)
3243:  //Func SendAppMessage( Msg :Card; WParam, LParam : Longint) : Longint
3244:  //Proc MoveWindowOrg( DC : HDC; DX, DY : Int)
3245:  Proc SetImeMode( hWnd : HWND; Mode : TImeMode)
3246:  Proc SetImeName( Name : TImeName)
3247:  Func Win32NLSEnableIME( hWnd : HWND; Enable :Bool) :Bool
3248:  Func Imm32GetContext( hWnd : HWND) : HIMC
3249:  Func Imm32ReleaseContext( hWnd: HWND; hImc : HIMC) :Bool
3250:  Func Imm32GetConversionStatus(hImc : HIMC; var Conversion, Sentence : longword):Bool
3251:  Func Imm32SetConversionStatus(hImc : HIMC; Conversion, Sentence : longword):Bool
3252:  Func Imm32SetOpenStatus( hImc : HIMC; fOpen :Bool) :Bool
3253: // Func Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM):Bool
3254:  //Func Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) :Bool
3255:  Func Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:str;dwBufLen:longint):Longint
3256:  Func Imm32IsIME( hKl : longword) :Bool
3257:  Func Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3258:  Proc DragDone( Drop :Bool)
3259:
3260:
3261: //*****************************************add  from jvjvclutils
3262: Func CanvasMaxTextHeight(Canvas: TCanvas): Int;
3263: Func ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Bool;
3264: Proc DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Int);
3265: Func IsPositiveResult(Value: TModalResult):Bool;
3266: Func IsNegativeResult(Value: TModalResult):Bool;
3267: Func IsAbortResult(const Value: TModalResult):Bool;
3268: Func StripAllFromResult(const Value: TModalResult): TModalResult;
3269: // returns either BrightColor or DarkColor depending on the luminance of AColor
3270: // This Func gives the same result (AFAIK) as the Func used in Windows to
3271: // calculate the desktop icon text color based on the desktop background color
3272: Func SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3273: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3274: Proc HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3275:   const State: TOwnerDrawState; const Text:Str; var Width: Int;
3276:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Int; var MouseOnLink:Bool;
3277:   var LinkName:Str; Scale: Int = 100); overload;
3278: Proc HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3279:   const State: TOwnerDrawState; const Text:Str; var Width, Height: Int;
3280:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Int; var MouseOnLink:Bool;
3281:   var LinkName:Str; Scale: Int = 100); overload;
3282: Func HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3283:   const State: TOwnerDrawState; const Text:Str; Scale: Int = 100):Str;
3284: Func HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3285:   const State: TOwnerDrawState; const Text:Str; MouseX, MouseY: Int;
3286:   Scale: Int = 100):Str;
3287: Func HTMLPlainText(const Text:Str):Str;
3288: Func HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3289:   const State: TOwnerDrawState; const Text:Str; Scale: Int = 100): TSize;
3290: Func HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3291:   const State: TOwnerDrawState; const Text:Str; Scale: Int = 100): Int;
3292: Func HTMLTextHeight(Canvas: TCanvas; const Text:str;Scale:Int = 100):Int;
3293: Func HTMLPrepareText(const Text:Str):Str;
3294:
3295: ****************************************** uPSI_JvAppUtils;
3296: Func GetDefaultSection( Component : TComponent) :Str
3297: Proc GetDefaultIniData(Control:TControl;var IniFileName,Section:str;UseRegistry:Bool)
3298: Proc GetDefaultIniData( Control : TControl; var IniFileName, Section :Str)
3299: Func GetDefaultIniName :Str
3300:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3301: Func GetDefaultIniRegKey :Str
3302: Func FindForm( FormClass : TFormClass) : TForm
3303: Func FindShowForm( FormClass : TFormClass; const Caption :Str) : TForm
3304: Func ShowDialog( FormClass : TFormClass) :Bool
3305:  //Func InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3306: Proc SaveFormPlacement( Form : TForm; const IniFileName :Str; UseRegistry :Bool)
3307: Proc RestoreFormPlacement(Form : TForm; const IniFileName :Str; UseRegistry :Bool)
3308: Proc SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3309: Proc SaveFormPlacement( Form : TForm; const IniFileName :Str)
3310: Proc RestoreFormPlacement( Form : TForm; const IniFileName :Str)
3311: Func GetUniqueFileNameInDir( const Path, FileNameMask :Str) :Str
3312: Func StrToIniStr( const Str :Str) :Str
3313: Func IniStrToStr( const Str :Str) :Str
3314: Func IniReadString( IniFile : TObject; const Section, Ident, Default:Str):Str
3315: Proc IniWriteString( IniFile : TObject; const Section, Ident, Value:Str)
3316: Func IniReadInt(IniFile:TObject;const Section, Ident :Str; Default:Longint): Longint
3317: Proc IniWriteInt( IniFile : TObject; const Section, Ident :Str; Value : Longint)
3318: Func IniReadBool(IniFile:TObject; const Section,Ident:str; Default:Boolean):Bool
3319: Proc IniWriteBool( IniFile : TObject; const Section, Ident :Str; Value :Bool)
3320: Proc IniReadSections( IniFile : TObject; Strings : TStrings)
3321: Proc IniEraseSection( IniFile : TObject; const Section :Str)
3322: Proc IniDeleteKey( IniFile : TObject; const Section, Ident :Str)
```

```
3323: Proc AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3324: Proc AppBroadcast( Msg, wParam : Word; lParam : Longint)
3325: Proc AppTaskbarIcons( AppOnly :Bool)
3326: Proc InternalSaveGridLayout(Grid : TCustomGrid; IniFile : TObject; const Section:str)
3327: Proc InternalRestoreGridLayout(Grid:TCustomGrid;IniFile : TObject; const Section:str)
3328: Proc InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3329: Proc InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3330: ******************************************* uPSI_JvDBUtils;
3331: Func CreateLocate( DataSet : TDataSet) : TJvLocateObject
3332: Func IsDataSetEmpty( DataSet : TDataSet) :Bool
3333: Proc RefreshQuery( Query : TDataSet)
3334: Func DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:str;CaseInsensitive:Bool):Bool
3335: Func DataSetSectionName( DataSet : TDataSet) :Str
3336: Proc InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section :Str)
3337: Proc InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:str;RestoreVisible:Bool)
3338: Func DataSetLocateThrough(DataSet:TDataSet;const KeyFields:str;const
      KeyValues:Variant;Options:TLocateOptions):Bool
3339: Proc SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3340: Proc RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible :Bool)
3341: Proc AssignRecord( Source, Dest : TDataSet; ByName :Bool)
3342: Func ConfirmDelete :Bool
3343: Proc ConfirmDataSetCancel( DataSet : TDataSet)
3344: Proc CheckRequiredField( Field : TField)
3345: Proc CheckRequiredFields( const Fields : array of TField)
3346: Func DateToSQL( Value : TDateTime) :Str
3347: Func FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName :Str) :Str
3348: Func FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName :Str) :Str
3349: Func FormatSQLNumericRange(const FieldName:str;LowVal,HighVal,LowEmpty,HighEmpty:Double;Inclusive:Bool):str
3350: Func StrMaskSQL( const Value :Str) :Str
3351: Func FormatSQLCondition(const FieldName,Operator,Val:str;FieldType:TFieldType;Exact:Bool):str
3352: Func FormatAnsiSQLCondition(const FieldName,Operator,Val:str;FieldType:TFieldType;Exact:Bool):str
3353: Proc _DBError( const Msg :Str)
3354:  Const('TrueExpr','String '0=0
3355:  Const('sdfStandard16','String '''"''mm''/''dd''/''yyyy''"''
3356:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3357:  Const('sdfOracle','String '"TO_DATE('''dd/mm/yyyy"''', ''DD/MM/YYYY'')"
3358:  Const('sdfInterbase','String '"CAST('''mm"/"dd"/"yyyy"'' AS DATE)"
3359:  Const('sdfMSSQL','String '"CONVERT(datetime, '''mm"/"dd"/"yyyy"''', 103)"
3360:  AddTypeS('Largeint', 'Longint
3361:  TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3362:    erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3363:   erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3364:   erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3365: 'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederror);
3366: (*----------------------------------------------------------------------------*)
3367: Proc SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3368: begin
3369:  Func JIniReadBool( const FileName, Section, Line :Str) :Bool
3370:  Func JIniReadInt( const FileName, Section, Line :Str) : Int
3371:  Func JIniReadString( const FileName, Section, Line :Str) :Str
3372:  Proc JIniWriteBool( const FileName, Section, Line :Str; Value :Bool)
3373:  Proc JIniWriteInt( const FileName, Section, Line :Str; Value : Int)
3374:  Proc JIniWriteString( const FileName, Section, Line, Value :Str)
3375:  Proc JIniReadStrings(IniFile:TCustomIniFile; const Section :Str; Strings TStrings)
3376:  Proc JIniWriteStrings(IniFile: TCustomIniFile; const Section:str; Strings:TStrings)
3377: end;
3378:
3379: (* === compile-time registration functions === *)
3380: (*----------------------------------------------------------------------------*)
3381: Proc SIRegister_JclDateTime(CL: TPSPascalCompiler);
3382: begin
3383:  'UnixTimeStart','LongInt'( 25569);
3384:  Func JEncodeDate( const Year : Int; Month, Day : Word) : TDateTime
3385:  Proc JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3386:  Proc DecodeDate1( Date : TDateTime; var Year : Int; var Month, Day : Word);
3387:  Proc DecodeDate2( Date : TDateTime; var Year, Month, Day : Int);
3388:  Func CenturyOfDate( const DateTime : TDateTime) : Int
3389:  Func CenturyBaseYear( const DateTime : TDateTime) : Int
3390:  Func DayOfDate( const DateTime : TDateTime) : Int
3391:  Func MonthOfDate( const DateTime : TDateTime) : Int
3392:  Func YearOfDate( const DateTime : TDateTime) : Int
3393:  Func JDayOfTheYear( const DateTime : TDateTime; var Year : Int) : Int;
3394:  Func DayOfTheYear1( const DateTime : TDateTime) : Int;
3395:  Func DayOfTheYearToDateTime( const Year, Day : Int) : TDateTime
3396:  Func HourOfTime( const DateTime : TDateTime) : Int
3397:  Func MinuteOfTime( const DateTime : TDateTime) : Int
3398:  Func SecondOfTime( const DateTime : TDateTime) : Int
3399:  Func GetISOYearNumberOfDays( const Year : Word) : Word
3400:  Func IsISOLongYear( const Year : Word) :Bool;
3401:  Func IsISOLongYear1( const DateTime : TDateTime) :Bool;
3402:  Func ISODayOfWeek( const DateTime : TDateTime) : Word
3403:  Func JISOWeekNumber(DateTime:TDateTime; var YearOfWeekNumber,WeekDay: Int) : Int;
3404:  Func ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Int) : Int;
3405:  Func ISOWeekNumber2( DateTime : TDateTime) : Int;
3406:  Func ISOWeekToDateTime( const Year, Week, Day : Int) : TDateTime
3407:  Func JIsLeapYear( const Year : Int) :Bool;
3408:  Func IsLeapYear1( const DateTime : TDateTime) :Bool;
3409:  Func JDaysInMonth( const DateTime : TDateTime) : Int
3410:  Func Make4DigitYear( Year, Pivot : Int) : Int
```

```
3411:  Func JMakeYear4Digit( Year, WindowsillYear : Int) : Int
3412:  Func JEasterSunday( const Year : Int) : TDateTime // TDosDateTime', 'Int
3413:  Func JFormatDateTime( Form :Str; DateTime : TDateTime) :Str
3414:  Func FATDatesEqual( const FileTime1, FileTime2 : Int64) :Bool;
3415:  Func FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) :Bool;
3416:  Func HoursToMSecs( Hours : Int) : Int
3417:  Func MinutesToMSecs( Minutes : Int) : Int
3418:  Func SecondsToMSecs( Seconds : Int) : Int
3419:  Func TimeOfDateTimeToSeconds( DateTime : TDateTime) : Int
3420:  Func TimeOfDateTimeToMSecs( DateTime : TDateTime) : Int
3421:  Func DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3422:  Func LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3423:  Func DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3424:  Func JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3425:  Func JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3426:  Proc DateTimeToSystemTime1( DateTime : TDateTime; var SysTime:TSystemTime);
3427:  Func LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3428:  Func DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3429:  Func JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3430:  Proc DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3431:  Func DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3432:  Func DosDateTimeToStr( DateTime : Int) :Str
3433:  Func JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3434:  Func FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3435:  Func JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3436:  Proc FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time: Word);
3437:  Func JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3438:  Proc FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3439:  Func FileTimeToStr( const FileTime : TFileTime) :Str
3440:  Func SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3441:  Func JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3442:  Proc SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3443:  Func SystemTimeToStr( const SystemTime : TSystemTime) :Str
3444:  Func CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3445:  Func LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3446:  Func LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3447:   TJclUnixTime32', 'Longword
3448:  Func JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3449:  Func JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3450:  Func FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3451:  Func UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3452:  Func JNullStamp : TTimeStamp
3453:  Func JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp): Int64
3454:  Func JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) :Bool
3455:  Func JIsNullTimeStamp( const Stamp : TTimeStamp) :Bool
3456:  Func TimeStampDOW( const Stamp : TTimeStamp) : Int
3457:  Func FirstWeekDay( const Year, Month : Int; var DOW : Int): Int;
3458:  Func FirstWeekDay1( const Year, Month : Int) : Int;
3459:  Func LastWeekDay( const Year, Month : Int; var DOW : Int): Int;
3460:  Func LastWeekDay1( const Year, Month : Int) : Int;
3461:  Func IndexedWeekDay( const Year, Month : Int; Index : Int): Int
3462:  Func FirstWeekendDay( const Year, Month : Int; var DOW : Int): Int;
3463:  Func FirstWeekendDay1( const Year, Month : Int) : Int;
3464:  Func LastWeekendDay( const Year, Month : Int; var DOW : Int): Int;
3465:  Func LastWeekendDay1( const Year, Month : Int) : Int;
3466:  Func IndexedWeekendDay( const Year, Month : Int; Index : Int): Int
3467:  Func FirstDayOfWeek( const Year, Month, DayOfWeek : Int) : Int
3468:  Func LastDayOfWeek( const Year, Month, DayOfWeek : Int) : Int
3469:  Func IndexedDayOfWeek( const Year, Month, DayOfWeek, Index:Int):Int
3470:   FindClass('TOBJECT'),'EJclDateTimeError
3471: end;
3472:
3473: Proc SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3474: begin
3475:  Func SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3476:  Func CreateDOSProcessRedirected(const CommandLine, InputFile,OutputFile:str):Bool
3477:  Func CreateDOSProcessRedirected2(const CommandLine, InputFile,OutputFile:str):Bool
3478:  Func CreateDOSProcessRedirected3(const CommandLine, InputFile, OutputFile, ErrMsg:Str):Bool;
3479:  Func WinExec32( const Cmd :Str; const CmdShow : Int) :Bool
3480:  Func WinExec32AndWait( const Cmd :Str; const CmdShow : Int) :Card
3481:  Func WinExec32AndRedirectOutput(const Cmd:str;var Output:str;RawOutput:Bool):Card
3482:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3483:  Func ExitWindows( ExitCode :Card) :Bool
3484:  Func LogOffOS( KillLevel : TJclKillLevel) :Bool
3485:  Func PowerOffOS( KillLevel : TJclKillLevel) :Bool
3486:  Func ShutDownOS( KillLevel : TJclKillLevel) :Bool
3487:  Func RebootOS( KillLevel : TJclKillLevel) :Bool
3488:  Func HibernateOS( Force, DisableWakeEvents :Bool) :Bool
3489:  Func SuspendOS( Force, DisableWakeEvents :Bool) :Bool
3490:  Func ShutDownDialog(const DialogMessage:str;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3491:  Func ShutDownDialog1(const MachineName,DialogMessage:str;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3492:  Func AbortShutDown :Bool;
3493:  Func AbortShutDown1( const MachineName :Str) :Bool;
3494:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3495:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3496:  Func GetAllowedPowerOperations : TJclAllowedPowerOperations
3497:   FindClass('TOBJECT'),'EJclCreateProcessError
3498:  Proc CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine :Str)
3499:  Proc CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:str;const Environment:PChar);
```

```
3500:   // with Add(EJclCreateProcessError) do
3501: end;
3502:
3503: Proc SIRegister_JclAnsistrs(CL: TPSPascalCompiler);
3504: begin
3505:   //'AnsiSigns','Set').SetSet(['-', '+']);
3506:  'C1_UPPER','LongWord( $0001);
3507:  'C1_LOWER','LongWord( $0002);
3508:  'C1_DIGIT','LongWord').SetUInt( $0004);
3509:  'C1_SPACE','LongWord').SetUInt( $0008);
3510:  'C1_PUNCT','LongWord').SetUInt( $0010);
3511:  'C1_CNTRL','LongWord').SetUInt( $0020);
3512:  'C1_BLANK','LongWord').SetUInt( $0040);
3513:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3514:  'C1_ALPHA','LongWord').SetUInt( $0100);
3515:   AnsiChar', 'Char
3516:  Func StrIsAlpha( const S : Ansistr) :Bool
3517:  Func StrIsAlphaNum( const S : Ansistr) :Bool
3518:  Func StrIsAlphaNumUnderscore( const S : Ansistr) :Bool
3519:  Func StrContainsChars(const S:Ansistr;Chars:TSysCharSet; CheckAll:Boolean):Bool
3520:  Func StrConsistsOfNumberChars( const S : Ansistr) :Bool
3521:  Func StrIsDigit( const S : Ansistr) :Bool
3522:  Func StrIsSubset( const S : Ansistr; const ValidChars : TSysCharSet): Bool
3523:  Func StrSame( const S1, S2 : Ansistr) :Bool
3524:  //Func StrCenter( const S : Ansistr; L : Int; C : AnsiChar) : Ansistr
3525:  Func StrCharPosLower( const S : Ansistr; CharPos : Int) : Ansistr
3526:  Func StrCharPosUpper( const S : Ansistr; CharPos : Int) : Ansistr
3527:  Func StrDoubleQuote( const S : Ansistr) : Ansistr
3528:  Func StrEnsureNoPrefix( const Prefix, Text : Ansistr) : Ansistr
3529:  Func StrEnsureNoSuffix( const Suffix, Text : Ansistr) : Ansistr
3530:  Func StrEnsurePrefix( const Prefix, Text : Ansistr) : Ansistr
3531:  Func StrEnsureSuffix( const Suffix, Text : Ansistr) : Ansistr
3532:  Func StrEscapedToString( const S : Ansistr) : Ansistr
3533:  Func JStrLower( const S : Ansistr) : Ansistr
3534:  Proc StrLowerInPlace( var S : Ansistr)
3535:  ///Proc StrLowerBuff( S : PAnsiChar)
3536:  Proc JStrMove( var Dest:AnsiStr;const Source:AnsiStr;const ToIndex,FromIndex,Count:Int;
3537:  Func StrPadLeft( const S : Ansistr; Len : Int; C : AnsiChar) : Ansistr
3538:  Func StrPadRight( const S : Ansistr; Len : Int; C : AnsiChar) : Ansistr
3539:  Func StrProper( const S : Ansistr) : Ansistr
3540:  //Proc StrProperBuff( S : PAnsiChar)
3541:  Func StrQuote( const S : Ansistr; C : AnsiChar) : Ansistr
3542:  Func StrRemoveChars( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
3543:  Func StrKeepChars( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
3544:  Proc JStrReplace(var S:Ansistr;const Search,Replace:Ansistr;Flags: TReplaceFlags)
3545:  Func StrReplaceChar( const S : Ansistr; const Source, Replace : AnsiChar) : Ansistr
3546:  Func StrReplaceChars(const S:AnsiStr;const Chars:TSysCharSet;Replace:AnsiChar):Ansistr
3547:  Func StrReplaceButChars(const S:AnsiStr;const Chars:TSysCharSet;Replace:AnsiChar):AnsiStr;
3548:  Func StrRepeat( const S : Ansistr; Count : Int) : Ansistr
3549:  Func StrRepeatLength( const S : Ansistr; const L : Int) : Ansistr
3550:  Func StrReverse( const S : Ansistr) : Ansistr
3551:  Proc StrReverseInPlace( var S : Ansistr)
3552:  Func StrSingleQuote( const S : Ansistr) : Ansistr
3553:  Func StrSmartCase( const S : Ansistr; Delimiters : TSysCharSet) : Ansistr
3554:  Func StrStringToEscaped( const S : Ansistr) : Ansistr
3555:  Func StrStripNonNumberChars( const S : Ansistr) : Ansistr
3556:  Func StrToHex( const Source : Ansistr) : Ansistr
3557:  Func StrTrimCharLeft( const S : Ansistr; C : AnsiChar) : Ansistr
3558:  Func StrTrimCharsLeft( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
3559:  Func StrTrimCharRight( const S : Ansistr; C : AnsiChar) : Ansistr
3560:  Func StrTrimCharsRight( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
3561:  Func StrTrimQuotes( const S : Ansistr) : Ansistr
3562:  Func JStrUpper( const S : Ansistr) : Ansistr
3563:  Proc StrUpperInPlace( var S : Ansistr)
3564:  //Proc StrUpperBuff( S : PAnsiChar)
3565:  Func StrOemToAnsi( const S : Ansistr) : Ansistr
3566:  Func StrAnsiToOem( const S : Ansistr) : Ansistr
3567:  Proc StrAddRef( var S : Ansistr)
3568:  Func StrAllocSize( const S : Ansistr) : Longint
3569:  Proc StrDecRef( var S : Ansistr)
3570:  //Func StrLen( S : PAnsiChar) : Int
3571:  Func StrLength( const S : Ansistr) : Longint
3572:  Func StrRefCount( const S : Ansistr) : Longint
3573:  Proc StrResetLength( var S : Ansistr)
3574:  Func StrCharCount( const S : Ansistr; C : AnsiChar) : Int
3575:  Func StrCharsCount( const S : Ansistr; Chars : TSysCharSet) : Int
3576:  Func StrStrCount( const S, SubS : Ansistr) : Int
3577:  Func StrCompare( const S1, S2 : Ansistr) : Int
3578:  Func StrCompareRange( const S1, S2 : Ansistr; const Index, Count:Int) : Int
3579:  //Func StrFillChar( const C : AnsiChar; Count : Int) : Ansistr;
3580:  Func StrFillChar1( const C : Char; Count : Int) : Ansistr;
3581:  Func StrFillChar(const C: Char; Count: Int):Str);
3582:  Func IntFillChar(const I: Int; Count: Int):Str);
3583:  Func ByteFillChar(const B: Byte; Count: Int):Str);
3584:  Func ArrFillChar(const AC: Char; Count: Int): TCharArray;;
3585:  Func ArrByteFillChar(const AB: Char; Count: Int): TByteArray;
3586:  Func StrFind( const Substr, S : Ansistr; const Index : Int) : Int
3587:  //Func StrHasPrefix( const S : Ansistr; const Prefixes : array of Ansistr) :Bool
3588:  Func StrIndex( const S : Ansistr; const List : array of Ansistr) : Int
```

```
3589:  Func StrILastPos( const SubStr, S : Ansistr) : Int
3590:  Func StrIPos( const SubStr, S : Ansistr) : Int
3591:  Func StrIsOneOf( const S : Ansistr; const List : array of Ansistr) :Bool
3592:  Func StrLastPos( const SubStr, S : Ansistr) : Int
3593:  Func StrMatch( const Substr, S : Ansistr; const Index : Int) : Int
3594:  Func StrMatches( const Substr, S : Ansistr; const Index : Int) :Bool
3595:  Func StrNIPos( const S, SubStr : Ansistr; N : Int) : Int
3596:  Func StrNPos( const S, SubStr : Ansistr; N : Int) : Int
3597:  Func StrPrefixIndex( const S : Ansistr; const Prefixes : array of Ansistr) : Int
3598:  Func StrSearch( const Substr, S : Ansistr; const Index : Int) : Int
3599:  //Func StrAfter( const SubStr, S : Ansistr) : Ansistr
3600:  //Func StrBefore( const SubStr, S : Ansistr) : Ansistr
3601:  Func StrBetween( const S : Ansistr; const Start, Stop : AnsiChar) : Ansistr
3602:  Func StrChopRight( const S : Ansistr; N : Int) : Ansistr
3603:  Func StrLeft( const S : Ansistr; Count : Int) : Ansistr
3604:  Func StrMid( const S : Ansistr; Start, Count : Int) : Ansistr
3605:  Func StrRestOf( const S : Ansistr; N : Int) : Ansistr
3606:  Func StrRight( const S : Ansistr; Count : Int) : Ansistr
3607:  Func CharEqualNoCase( const C1, C2 : AnsiChar) :Bool
3608:  Func CharIsAlpha( const C : AnsiChar) :Bool
3609:  Func CharIsAlphaNum( const C : AnsiChar) :Bool
3610:  Func CharIsBlank( const C : AnsiChar) :Bool
3611:  Func CharIsControl( const C : AnsiChar) :Bool
3612:  Func CharIsDelete( const C : AnsiChar) :Bool
3613:  Func CharIsDigit( const C : AnsiChar) :Bool
3614:  Func CharIsLower( const C : AnsiChar) :Bool
3615:  Func CharIsNumberChar( const C : AnsiChar) :Bool
3616:  Func CharIsPrintable( const C : AnsiChar) :Bool
3617:  Func CharIsPunctuation( const C : AnsiChar) :Bool
3618:  Func CharIsReturn( const C : AnsiChar) :Bool
3619:  Func CharIsSpace( const C : AnsiChar) :Bool
3620:  Func CharIsUpper( const C : AnsiChar) :Bool
3621:  Func CharIsWhiteSpace( const C : AnsiChar) :Bool
3622:  Func CharType( const C : AnsiChar) : Word
3623:  Func CharHex( const C : AnsiChar) : Byte
3624:  Func CharLower( const C : AnsiChar) : AnsiChar
3625:  Func CharUpper( const C : AnsiChar) : AnsiChar
3626:  Func CharToggleCase( const C : AnsiChar) : AnsiChar
3627:  Func CharPos( const S : Ansistr; const C : AnsiChar; const Index : Int) : Int
3628:  Func CharLastPos( const S : Ansistr; const C : AnsiChar; const Index : Int) : Int
3629:  Func CharIPos( const S : Ansistr; C : AnsiChar; const Index : Int) : Int
3630:  Func CharReplace( var S : Ansistr; const Search, Replace : AnsiChar) : Int
3631:  Proc StrIToStrings(S,Sep:Ansistr; const List:TStrings;const AllowEmptyString :Bool)
3632:  Proc StrToStrings(S,Sep: Ansistr; const List:TStrings;const AllowEmptyString :Bool)
3633:  Func StringsToStr(const List:TStrings;const Sep:AnsiStr;const AllowEmptyString:Bool):AnsiStr;
3634:  Proc TrimStrings( const List : TStrings; DeleteIfEmpty :Bool)
3635:  Proc TrimStringsRight( const List : TStrings; DeleteIfEmpty :Bool)
3636:  Proc TrimStringsLeft( const List : TStrings; DeleteIfEmpty :Bool)
3637:  Func AddStringToStrings(const S:Ansistr;Strings:TStrings; const Unique:Boolean):Bool
3638:  Func BooleanToStr( B :Bool) : Ansistr
3639:  Func FileToString( const FileName : Ansistr) : Ansistr
3640:  Proc StringToFile( const FileName, Contents: Ansistr; Append:Bool) //binary save!
3641:  Proc StringToFile2( const S, FileName :Str);
3642:  Func StrToken( var S : Ansistr; Separator : AnsiChar) : Ansistr
3643:  Proc StrTokens( const S : Ansistr; const List : TStrings)
3644:  Proc StrTokenToStrings( S : Ansistr; Separator : AnsiChar; const List : TStrings)
3645:  //Func StrWord( var S : PAnsiChar; out Word : Ansistr) :Bool
3646:  Func StrToFloatSafe( const S : Ansistr) : Float
3647:  Func StrToIntSafe( const S : Ansistr) : Int
3648:  Proc StrNormIndex( const StrLen : Int; var Index : Int; var Count : Int);
3649:  Func ArrayOf( List : TStrings) : TDynStringArray;
3650:   EJclStringError', 'EJclError
3651:  Func IsClass(Address: TObject):Bool;
3652:  Func IsObject(Address: TObject):Bool;
3653:  // Console Utilities
3654:  //Func ReadKey: Char;
3655:  Func IntToStrZeroPad(Value, Count: Int): Ansistr;
3656:  Func JclGUIDToString(const GUID: TGUID):Str;
3657:  Func JclStringToGUID(const S:Str): TGUID;
3658: end;
3659:
3660: ********************************************** uPSI_JvDBUtil;
3661: Proc ExecuteSQLScript(Base:TDataBase;const Script:str; const Commit:TCommit;OnProgress:TOnProgress; const
      UserData : Int)
3662: Func GetQueryResult( const DatabaseName, SQL :Str) : Variant
3663: Func GetStoredProcResult(const ADatabaseName,AStoredProcName:str; AParams: array of Variant;const
      AResultName:str):Variant
3664: //Func StrFieldDesc( Field : FLDDesc) :Str
3665: Func Var2Type( V : Variant; const VarType : Int) : Variant
3666: Proc CopyRecord( DataSet : TDataSet)
3667: //Proc AddReference( Tbl : TTable; RefName:Str; RefField: Word; MasterTable :Str; MasterField : Word;
      ModOp, DelOp : RINTQual)
3668: Proc AddMasterPassword( Table : TTable; pswd :Str)
3669: Proc PackTable( Table : TTable)
3670: Proc PackEncryptedTable( Table : TTable; pswd :Str)
3671: Func EncodeQuotes( const S :Str) :Str
3672: Func Cmp( const S1, S2 :Str) :Bool
3673: Func SubStr( const S :Str; const Index : Int; const Separator :Str) :Str
3674: Func SubStrEnd( const S :Str; const Index : Int; const Separator :Str):Str
```

```
3675: Func ReplaceString( S :Str; const OldPattern, NewPattern :Str) :Str
3676: Proc GetXYByPos( const S :Str; const Pos : Int; var X, Y : Int)
3677: ***********************************************uPSI_JvJvBDEUtils;***************
3678: //JvBDEUtils
3679: Func CreateDbLocate : TJvLocateObject
3680:  //Func CheckOpen( Status : DBIResult) :Bool
3681: Proc FetchAllRecords( DataSet : TBDEDataSet)
3682: Func TransActive( Database : TDatabase) :Bool
3683: Func AsyncQrySupported( Database : TDatabase) :Bool
3684: Func GetQuoteChar( Database : TDatabase) :Str
3685: Proc ExecuteQuery( const DbName, QueryText :Str)
3686: Proc ExecuteQueryEx( const SessName, DbName, QueryText :Str)
3687: Func FieldLogicMap( FldType : TFieldType) : Int
3688: Func FieldSubtypeMap( FldType : TFieldType) : Int Value :Str; Buffer : Pointer)
3689: Func GetAliasPath( const AliasName :Str) :Str
3690: Func IsDirectory( const DatabaseName :Str) :Bool
3691: Func GetBdeDirectory :Str
3692: Func LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) :Bool
3693: Func DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName :Str) :Bool
3694: Func DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName :Str) :Bool
3695: Func DataSetRecNo( DataSet : TDataSet) : Longint
3696: Func DataSetRecordCount( DataSet : TDataSet) : Longint
3697: Func DataSetPositionStr( DataSet : TDataSet) :Str
3698: Proc DataSetShowDeleted( DataSet : TBDEDataSet; Show :Bool)
3699: Func CurrentRecordDeleted( DataSet : TBDEDataSet) :Bool
3700: Func IsFilterApplicable( DataSet : TDataSet) :Bool
3701: Func IsBookmarkStable( DataSet : TBDEDataSet) :Bool
3702: Proc SetIndex( Table : TTable; const IndexFieldNames :Str)
3703: Proc RestoreIndex( Table : TTable)
3704: Proc DeleteRange(Table: TTable; IndexFields:array of const; FieldValues: array of const)
3705: Proc PackTable( Table : TTable)
3706: Proc ReindexTable( Table : TTable)
3707: Proc BdeFlushBuffers
3708: Func GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Int) : Pointer
3709: Proc ToggleDebugLayer( Active :Bool; const DebugFile :Str)
3710: Proc DbNotSupported
3711: Proc ExportDataSet(Source:TBDEDataSet;DestTable:TTable;TableType:TTableType;const AsciiCharSe
      :str;AsciiDelimited Boolean;MaxRecordCount:Longint)
3712: Proc ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:str;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3713: Proc
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3714: Proc InitRSRUN(Database:TDatabase;const ConName:str;ConType:Int;const ConServer:str);
3715: ***********************************************uPSI_JvDateUtil;
3716: Func CurrentYear: Word;
3717: Func IsLeapYear(AYear: Int):Bool;
3718: Func DaysPerMonth(AYear, AMonth: Int): Int;
3719: Func FirstDayOfPrevMonth: TDateTime;
3720: Func LastDayOfPrevMonth: TDateTime;
3721: Func FirstDayOfNextMonth: TDateTime;
3722: Func ExtractDay(ADate: TDateTime): Word;
3723: Func ExtractMonth(ADate: TDateTime): Word;
3724: Func ExtractYear(ADate: TDateTime): Word;
3725: Func IncDate(ADate: TDateTime; Days, Months, Years: Int): TDateTime;
3726: Func IncDay(ADate: TDateTime; Delta: Int): TDateTime;
3727: Func IncMonth(ADate: TDateTime; Delta: Int): TDateTime;
3728: Func IncYear(ADate: TDateTime; Delta: Int): TDateTime;
3729: Func ValidDate(ADate: TDateTime):Bool;
3730: Proc DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3731: Func MonthsBetween(Date1, Date2: TDateTime): Double;
3732: Func DaysInPeriod(Date1, Date2: TDateTime): Longint;
3733: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3734: Func DaysBetween(Date1, Date2: TDateTime): Longint;
3735: { The same as previous but if Date2 < Date1 result = 0 }
3736: Func IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Int): TDateTime;
3737: Func IncHour(ATime: TDateTime; Delta: Int): TDateTime;
3738: Func IncMinute(ATime: TDateTime; Delta: Int): TDateTime;
3739: Func IncSecond(ATime: TDateTime; Delta: Int): TDateTime;
3740: Func IncMSec(ATime: TDateTime; Delta: Int): TDateTime;
3741: Func CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3742: { String to date conversions }
3743: Func GetDateOrder(const DateFormat:Str): TDateOrder;
3744: Func MonthFromName(const S:Str; MaxLen: Byte): Byte;
3745: Func StrToDateDef(const S:Str; Default: TDateTime): TDateTime;
3746: Func StrToDateFmt(const DateFormat, S:Str): TDateTime;
3747: Func StrToDateFmtDef(const DateFormat, S:Str; Default: TDateTime): TDateTime;
3748: Func DefDateFormat(FourDigitYear:Bool):Str;
3749: Func DefDateMask(BlanksChar: Char; FourDigitYear:Bool):Str;
3750: ---------------------------------------------------------------------------
3751: ********************************** JvUtils;******************************
3752: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3753: Func GetWordOnPos(const S:Str; const P: Int):Str;
3754: {GetWordOnPosEx worklike GetWordOnPosfunction,also returns Wordposition in iBeg,iEnd variables}
3755: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
3756: { SubStr returns substring from string, S, separated with Separator string}
3757: Func SubStr(const S:Str; const Index: Int; const Separator:Str):Str;
3758: { SubStrEnd same to previous Func but Index numerated from the end of string }
3759: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
3760: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
```

```
3761: Func SubWord(P: PChar; var P2: PChar):Str;
3762: { NumberByWord returns the text representation of
3763:   the number, N, in normal russian language. Was typed from Monitor magazine }
3764: Func NumberByWord(const N: Longint):Str;
3765: //  Func CurrencyByWord(Value : Currency) :Str; GetLineByPos returns Line number, there
3766:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3767: Func GetLineByPos(const S:Str; const Pos: Int): Int;
3768: { GetXYByPos is same to previous function, but returns X position in line too}
3769: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
3770: { ReplaceString searches for all substrings,OldPattern,in a string,S, replaces them with NewPattern }
3771: Func ReplaceString(S:Str; const OldPattern, NewPattern:Str):Str;
3772: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3773: Func ConcatSep(const S, S2, Separator:Str):Str;
3774: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3775: Func ConcatLeftSep(const S, S2, Separator:Str):Str;
3776: { MinimizeString truncs long string,S,appends '...' symbols,if Length of S is more than MaxLen}
3777: Func MinimizeString(const S:Str; const MaxLen: Int):Str;
3778: { Next 4 Func for russian chars transliterating.
3779:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3780: Proc Dos2Win(var S:Str);
3781: Proc Win2Dos(var S:Str);
3782: Func Dos2WinRes(const S:Str):Str;
3783: Func Win2DosRes(const S:Str):Str;
3784: Func Win2Koi(const S:Str):Str;
3785: { Spaces returns string consists on N space chars }
3786: Func Spaces(const N: Int):Str;
3787: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3788: Func AddSpaces(const S:Str; const N: Int):Str;
3789: { Func LastDate for russian users only } { returns date relative to current date: '' }
3790: Func LastDate(const Dat: TDateTime):Str;
3791: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3792: Func CurrencyToStr(const Cur: currency):Str;
3793: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3794: Func Cmp(const S1, S2:Str):Bool;
3795: { StringCat add S2 string to S1 and returns this string }
3796: Func StringCat(var S1:Str; S2:Str):Str;
3797: { HasChar returns True, if Char, Ch, contains in string, S }
3798: Func HasChar(const Ch: Char; const S:Str):Bool;
3799: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
3800: Func CharInSet(const Ch: Char; const SetOfChar: TSetOfChar):Bool;
3801: Func CountOfChar(const Ch: Char; const S:Str): Int;
3802: Func DefStr(const S:Str; Default:Str):Str;
3803: {**** files routines}
3804: { GetWinDir returns Windows folder name }
3805: Func GetWinDir: TFileName;
3806: Func GetSysDir:Str;
3807: { GetTempDir returns Windows temporary folder name }
3808: Func GetTempDir:Str;
3809: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3810: Func GenTempFileName(FileName:Str):Str;
3811: { GenTempFileNameExt sameto previous function,but returning filename has given extension, FileExt }
3812: Func GenTempFileNameExt(FileName:Str; const FileExt:Str):Str;
3813: { ClearDir clears folder Dir }
3814: Func ClearDir(const Dir:Str):Bool;
3815: { DeleteDir clears and than delete folder Dir }
3816: Func DeleteDir(const Dir:Str):Bool;
3817: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3818: Func FileEquMask(FileName, Mask: TFileName):Bool;
3819: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3820:   Masks must be separated with comma (';') }
3821: Func FileEquMasks(FileName, Masks: TFileName):Bool;
3822: Proc DeleteFiles(const Folder: TFileName; const Masks:Str);
3823: { LZFileExpand expand file,FileSource into FileDest.File must be compressed,using MS Compress program }
3824: Func LZFileExpand(const FileSource, FileDest:Str):Bool;
3825: { FileGetInfo fills SearchRec record for specified file attributes}
3826: Func FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec):Bool;
3827: { HasSubFolder returns True, if folder APath contains other folders }
3828: Func HasSubFolder(APath: TFileName):Bool;
3829: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3830: Func IsEmptyFolder(APath: TFileName):Bool;
3831: { AddSlash add slash Char to Dir parameter, if needed }
3832: Proc AddSlash(var Dir: TFileName);
3833: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3834: Func AddSlash2(const Dir: TFileName):Str;
3835: { AddPath returns FileName with Path, if FileName not contain any path }
3836: Func AddPath(const FileName, Path: TFileName): TFileName;
3837: Func AddPaths(const PathList, Path:Str):Str;
3838: Func ParentPath(const Path: TFileName): TFileName;
3839: Func FindInPath(const FileName, PathList:Str): TFileName;
3840: Func FindInPaths(const fileName,paths:Str):Str;
3841: {$IFNDEF BCB1}
3842: { BrowseForFolder displays Browse For Folder dialog }
3843: Func BrowseForFolder(const Handle: HWND; const Title:Str; var Folder:Str):Boolean;
3844: {$ENDIF BCB1}
3845: Func BrowseForFolder(const ATitle:Str; AllowCreate :Bool; var ADirectory :str; AHelpContext :
      THelpContext) :Bool
3846: Func BrowseForComputer(const ATitle :Str; AllowCreate :Bool; var ADirectory:str; AHelpContext :
      THelpContext): Bool
3847: Func BrowseDirectory(var AFolderName:str;const DlgText:str;AHelpContext:THelpContext):Bool
```

```
3848: Func BrowseComputer(var AComputerName:str;const DlgText:str;AHelpContext:THelpContext):Bool
3849: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3850: Func DeleteReadOnlyFile(const FileName: TFileName):Bool;
3851: { HasParam returns True, if program running with specified parameter, Param }
3852: Func HasParam(const Param:Str):Bool;
3853: Func HasSwitch(const Param:Str):Bool;
3854: Func Switch(const Param:Str):Str;
3855: { ExePath returns ExtractFilePath(ParamStr(0)) }
3856: Func ExePath: TFileName;
3857: Func CopyDir(const SourceDir, DestDir: TFileName):Bool;
3858: Func FileTimeToDateTime(const FT: TFileTime): TDateTime;
3859: Func MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3860: {**** Graphic routines }
3861: { TTFontSelected returns True, if True Type font is selected in specified device context }
3862: Func TTFontSelected(const DC: HDC):Bool;
3863: { TrueInflateRect inflates rect in other method, than InflateRect API Func }
3864: Func TrueInflateRect(const R: TRect; const I: Int): TRect;
3865: {**** Windows routines }
3866: { SetWindowTop put window to top without recreating window }
3867: Proc SetWindowTop(const Handle: HWND; const Top:Bool);
3868: {**** other routines }
3869: { KeyPressed returns True, if Key VK is now pressed }
3870: Func KeyPressed(VK: Int):Bool;
3871: Proc SwapInt(var Int1, Int2: Int);
3872: Func IntPower(Base, Exponent: Int): Int;
3873: Func ChangeTopException(E: TObject): TObject;
3874: Func StrToBool(const S:Str):Bool;
3875: {$IFNDEF COMPILER3_UP}
3876: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3877:   Length of MaxLen bytes. The compare operation is controlled by the
3878:   current Windows locale. The return value is the same as for CompareStr. }
3879: Func AnsiStrLIComp(S1, S2: PChar; MaxLen:Card): Int;
3880: Func AnsiStrIComp(S1, S2: PChar): Int;
3881: {$ENDIF}
3882: Func Var2Type(V: Variant; const VarType: Int): Variant;
3883: Func VarToInt(V: Variant): Int;
3884: Func VarToFloat(V: Variant): Double;
3885: {following functions are not documented because they are don't work properly,so don't use them}
3886: Func ReplaceSokr1(S:Str; const Word, Frase:Str):Str;
3887: { ReplaceSokr1 is full equal to ReplaceString Func - only for compatibility - don't use }
3888: { GetSubStr is full equal to SubStr Func - only for compatibility - don't use }
3889: Func GetSubStr(const S:Str; const Index: Int; const Separator: Char):Str;
3890: Func GetParameter:Str;
3891: Func GetLongFileName(FileName:Str):Str;
3892: {* from  FileCtrl}
3893: Func DirectoryExists(const Name:Str):Bool;
3894: Proc ForceDirectories(Dir:Str);
3895: {# from  FileCtrl}
3896: Func FileNewExt(const FileName, NewExt: TFileName): TFileName;
3897: Func GetComputerID:Str;
3898: Func GetComputerName:Str;
3899: {**** string routines }
3900: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 Func }
3901: Func ReplaceAllSokr(S:Str; Words, Frases: TStrings):Str;
3902: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3903:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with same
      Index, and then update NewSelStart variable }
3904: Func ReplaceSokr(S:str;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):Str;
3905: { CountOfLines calculates linescount in a string,each line separated from another with CrLf sequence }
3906: Func CountOfLines(const S:Str): Int;
3907: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3908: Proc DeleteEmptyLines(Ss: TStrings);
3909: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3910:   Note:If strings SQL allready contains where-statement,it must be started on begining of any line}
3911: Proc SQLAddWhere(SQL: TStrings; const Where:Str);
3912: {**** files routines - }
3913: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3914:   Resource can be compressed using MS Compress program}
3915: Func ResSaveToFile(const Typ,Name:str;const Compressed:Bool;const FileName:str):Bool;
3916: Func ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:str): Bool
3917: Func ResSaveToString(Instance: HINST; const Typ, Name:Str; var S:Str):Bool;
3918: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3919: Func ExecuteJ(const CommandLine, WorkingDirectory:Str): Int;
3920: { IniReadSection read section, Section, from ini-file,
3921:   IniFileName, into strings, Ss.This Func reads ALL strings from specified section.
3922:   Note: TIninFile.ReadSection Func reads only strings with '=' symbol.}
3923: Func IniReadSection(const IniFileName:TFileName;const Section:str;Ss:TStrings):Bool;
3924: { LoadTextFile load text file, FileName, into string }
3925: Func LoadTextFile(const FileName: TFileName):Str;
3926: Proc SaveTextFile(const FileName: TFileName; const Source:Str);
3927: { ReadFolder reads files list from disk folder,Folder,that are equal Mask,into strings,FileList}
3928: Func ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Int;
3929: Func ReadFolders(const Folder: TFileName; FolderList: TStrings): Int;
3930: {$IFDEF COMPILER3_UP}
3931: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3932: Func TargetFileName(const FileName: TFileName): TFileName;
3933: { return filename ShortCut linked to }
3934: Func ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
```

```
3935: {$ENDIF COMPILER3_UP}
3936: {**** Graphic routines - }
3937: {LoadIcoToImage loads 2 icons from resource named NameRes,into twoimage lists ALarge and ASmall}
3938: Proc LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes:Str);
3939: { RATextOut same with TCanvas.TextOut procedure,but can clipping drawing with rectangle,RClip.}
3940: Proc RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S:Str);
3941: {RATextOutEx same with RATextOut funct, but can calculate needed height for correct output }
3942: Func RATextOutEx(Canvas:TCanvas;const R,RClip:TRect;const S:str;const CalcHeight:Bool):Int;
3943: {RATextCalcHeight calculate needed height to correct output,using RATextOut or RATextOutEx functions}
3944: Func RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S:Str): Int;
3945: { Cinema draws some visual effect }
3946: Proc Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3947: { Roughed fills rect with special 3D pattern }
3948: Proc Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert:Bool);
3949: { 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}
3950: Func BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Int): TBitmap;
3951: { TextWidth calculate text with for writing using standard desktop font }
3952: Func TextWidth(AStr:Str): Int;
3953: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3954: Func DefineCursor(Identifer: PChar): TCursor;
3955: {**** other routines - }
3956: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3957: Func FindFormByClass(FormClass: TFormClass): TForm;
3958: Func FindFormByClassName(FormClassName:Str): TForm;
3959: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3960:   having Tag property value, equaled to Tag parameter }
3961: Func FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Int):TComponent;
3962: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3963: Func ControlAtPos2(Parent: TWinControl; X, Y: Int): TControl;
3964: { RBTag searches WinControl.Controls for checked RadioButton and returns Tag property value }
3965: Func RBTag(Parent: TWinControl): Int;
3966: { AppMinimized returns True, if Application is minimized }
3967: Func AppMinimized:Bool;
3968: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3969:   if Caption parameter = '', it replaced with Application.Title }
3970: Func MessageBoxJ(const Msg:Str; Caption:Str;const Flags: Int): Int;
3971: Func MsgDlg2(const Msg, ACaption:Str; DlgType: TMsgDlgType;
3972:   Buttons: TMsgDlgButtons; HelpContext: Int; Control: TWinControl): Int;
3973: Func MsgDlgDef(const Msg, ACaption:Str; DlgType: TMsgDlgType;
3974:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Int;Control: TWinControl):Int;
3975: { Delay stop program execution to MSec msec }
3976: Proc Delay(MSec: Longword);
3977: Proc CenterHor(Parent: TControl; MinLeft: Int; Controls: array of TControl);
3978: Proc EnableControls(Control: TWinControl; const Enable:Bool);
3979: Proc EnableMenuItems(MenuItem: TMenuItem; const Tag: Int; const Enable:Bool);
3980: Proc ExpandWidth(Parent: TControl; MinWidth: Int; Controls: array of TControl);
3981: Func PanelBorder(Panel: TCustomPanel): Int;
3982: Func Pixels(Control: TControl; APixels: Int): Int;
3983: Proc SetChildPropOrd(Owner: TComponent; PropName:Str; Value: Longint);
3984: Proc Error(const Msg:Str);
3985: Proc ItemHtDrawEx(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
3986:   const HideSelColor:Bool; var PlainItem:Str; var Width: Int; CalcWidth:Bool);
3987:   {ex.Text parameter:'Item1<b>bold</b><i>italic ITALIC<c:Red>red<c:Green>green<c:blue>blue</i>'}
3988: Func ItemHtDraw(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
3989:   const HideSelColor:Bool):Str;
3990: Func ItemHtWidth(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
3991:   const HideSelColor:Bool): Int;
3992: Func ItemHtPlain(const Text:Str):Str;
3993: { ClearList - clears list of TObject }
3994: Proc ClearList(List: TList);
3995: Proc MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3996: Proc ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3997: { RTTI support }
3998: Func GetPropType(Obj: TObject; const PropName:Str): TTypeKind;
3999: Func GetPropStr(Obj: TObject; const PropName:Str):Str;
4000: Func GetPropOrd(Obj: TObject; const PropName:Str): Int;
4001: Func GetPropMethod(Obj: TObject; const PropName:Str): TMethod;
4002: Proc PrepareIniSection(SS: TStrings);
4003: {following functions are not documented because they are don't work properly,so don't use them}
4004: {$IFDEF COMPILER2}
4005: Func CompareMem(P1, P2: Pointer; Length: Int):Bool; assembler;
4006: {$ENDIF}
4007:
4008: Proc SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
4009: begin
4010:  Proc BoxMoveSelectedItems( SrcList, DstList : TWinControl)
4011:  Proc BoxMoveAllItems( SrcList, DstList : TWinControl)
4012:  Proc BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var Accept:Bool;Sorted:Bool;
4013:  Proc BoxMoveFocusedItem( List : TWinControl; DstIndex : Int)
4014:  Proc BoxMoveSelected( List : TWinControl; Items : TStrings)
4015:  Proc BoxSetItem( List : TWinControl; Index : Int)
4016:  Func BoxGetFirstSelection( List : TWinControl) : Int
4017:  Func BoxCanDropItem( List:TWinControl; X,Y:Int; var DragIndex : Int):Boolean
4018: end;
4019:
4020: Proc SIRegister_JvCsvParse(CL: TPSPascalCompiler);
4021: begin
4022:  Const('MaxInitStrNum','LongInt'( 9);
```

```
Func JvAnsiStrSplit( const InString : Ansistr; const SplitChar, QuoteChar:AnsiChar; var OutStrings : array of Ansistr; MaxSplit : Int) : Int
Func JvStrSplit(const InString:str;const SplitChar,QuoteChar:Char;var OutStrings:array of string;MaxSplit:Int):Int
Func JvAnsiStrSplitStrings(const InStr:Ansistr;const SplitChar,QuoteChar:AnsiChar;OutStrs:TStrings):Int;
Func JvAnsiStrStrip( S : Ansistr) : Ansistr
Func JvStrStrip( S :Str) :Str
Func GetString( var Source : Ansistr; const Separator : Ansistr) : Ansistr
Func PadString( const S : Ansistr; Len : Int; PadChar : AnsiChar) : Ansistr
Func BuildPathName( const PathName, FileName : Ansistr) : Ansistr
Func StrEatWhiteSpace( const S :Str) :Str
Func HexToAscii( const S : Ansistr) : Ansistr
Func AsciiToHex( const S : Ansistr) : Ansistr
Func StripQuotes( const S1 : Ansistr) : Ansistr
Func ValidNumericLiteral( S1 : PAnsiChar) :Bool
Func ValidIntLiteral( S1 : PAnsiChar) :Bool
Func ValidHexLiteral( S1 : PAnsiChar) :Bool
Func HexPCharToInt( S1 : PAnsiChar) : Int
Func ValidStringLiteral( S1 : PAnsiChar) :Bool
Func StripPCharQuotes( S1 : PAnsiChar) : Ansistr
Func JvValidIdentifierAnsi( S1 : PAnsiChar) :Bool
Func JvValidIdentifier( S1 :Str) :Bool
Func JvEndChar( X : AnsiChar) :Bool
Proc JvGetToken( S1, S2 : PAnsiChar)
Func IsExpressionKeyword( S1 : PAnsiChar) :Bool
Func IsKeyword( S1 : PAnsiChar) :Bool
Func JvValidVarReference( S1 : PAnsiChar) :Bool
Func GetParenthesis( S1, S2 : PAnsiChar) :Bool
Proc JvGetVarReference( S1, S2, SIdx : PAnsiChar)
Proc JvEatWhitespaceChars( S1 : PAnsiChar);
Proc JvEatWhitespaceChars1( S1 : PWideChar);
Func GetTokenCount : Int
Proc ResetTokenCount
end;

Proc SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
begin
  SIRegister_TJvQueryParamsDialog(CL);
 Func EditQueryParams(DataSet:TDataSet;List:TParams;AHelpContext:THelpContext):Bool
end;

********************************** JvStrUtil /  JvStrUtils;****************************
Func FindNotBlankCharPos(const S:Str): Int;
Func AnsiChangeCase(const S:Str):Str;
Func GetWordOnPos(const S:Str; const P: Int):Str;
Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
Func Cmp(const S1, S2:Str):Bool;
{ Spaces returns string consists on N space chars }
Func Spaces(const N: Int):Str;
{ HasChar returns True, if char, Ch, contains in string, S }
Func HasChar(const Ch: Char; const S:Str):Bool;
Func HasAnyChar(const Chars:Str; const S:Str):Bool;
{ SubStr returns substring from string, S, separated with Separator string}
Func SubStr(const S:Str; const Index: Int; const Separator:Str):Str;
{ SubStrEnd same to previous Func but Index numerated from the end of string }
Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
{ ReplaceString searches forall substrings,OldPattern,in a string,S,replaces with NewPattern }
Func ReplaceString(S:Str; const OldPattern, NewPattern:Str):Str;
Func CharInSet(const Ch: Char; const SetOfChar: TSetOfChar):Bool;
{ GetXYByPos is same to previous function, but returns X position in line too}
Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
{ AddSlash returns string with added slash char to Dir parameter, if needed }
Func AddSlash2(const Dir: TFileName):Str;
{ AddPath returns FileName with Path, if FileName not contain any path }
Func AddPath(const FileName, Path: TFileName): TFileName;
{ ExePath returns ExtractFilePath(ParamStr(0)) }
Func ExePath: TFileName;
Func LoadTextFile(const FileName: TFileName):Str;
Proc SaveTextFile(const FileName: TFileName; const Source:Str);
{ ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
Func ConcatSep(const S, S2, Separator:Str):Str;
{ FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
Func FileEquMask(FileName, Mask: TFileName):Bool;
{ FileEquMasks returns True if file, FileName, is compatible with given Masks.
  Masks must be separated with comma (';') }
Func FileEquMasks(FileName, Masks: TFileName):Bool;
Func StringEndsWith(const Str, SubStr:Str):Bool;
Func ExtractFilePath2(const FileName:Str):Str;
Func StrToOem(const AnsiStr:Str):Str;
{ StrToOem translates a string from the Windows character set into the OEM character set. }
Func OemToAnsiStr(const OemStr:Str):Str;
{ OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
Func IsEmptyStr(const S:Str; const EmptyChars: TCharSet):Bool;
{ EmptyStr returns true if the given string contains only character from the EmptyChars. }
Func ReplaceStr(const S, Srch, Replace:Str):Str;
{ Returns string with every occurrence of Srch string replaced with Replace string. }
Func DelSpace(const S:Str):Str;
{ DelSpace return a string with all white spaces removed. }
Func DelChars(const S:Str; Chr: Char):Str;
```

```
4110: { DelChars return a string with all Chr characters removed. }
4111: Func DelBSpace(const S:Str):Str;
4112: { DelBSpace trims leading spaces from the given string. }
4113: Func DelESpace(const S:Str):Str;
4114: { DelESpace trims trailing spaces from the given string. }
4115: Func DelRSpace(const S:Str):Str;
4116: { DelRSpace trims leading and trailing spaces from the given string. }
4117: Func DelSpace1(const S:Str):Str;
4118: { DelSpace1 return a string with all non-single white spaces removed. }
4119: Func Tab2Space(const S:Str; Numb: Byte):Str;
4120: { Tab2Space converts any tabulation chars in given string to the Numb spaces characters. }
4121: Func NPos(const C:Str; S:Str; N: Int): Int;
4122: { NPos searches for a N-th position of substring C in a given string. }
4123: Func MakeStr(C: Char; N: Int):Str;
4124: Func MS(C: Char; N: Int):Str;
4125: { MakeStr return a string of length N filled with character C. }
4126: Func AddChar(C: Char; const S:Str; N: Int):Str;
4127: { AddChar return a string left-padded to length N with characters C. }
4128: Func AddCharR(C: Char; const S:Str; N: Int):Str;
4129: { AddCharR return a string right-padded to length N with characters C. }
4130: Func LeftStr(const S:Str; N: Int):Str;
4131: { LeftStr return a string right-padded to length N with blanks. }
4132: Func RightStr(const S:Str; N: Int):Str;
4133: { RightStr return a string left-padded to length N with blanks. }
4134: Func CenterStr(const S:Str; Len: Int):Str;
4135: { CenterStr centers the characters in the string based upon the Len specified. }
4136: Func CompStr(const S1, S2:Str): Int;
4137: {CompStr compares S1 to S2,case-sensitivity.return val is -1 if S1<S2,0 if S1=S2,or 1 if S1>S2.}
4138: Func CompText(const S1, S2:Str): Int;
4139: {CompText compares S1 to S2,without case-sensitivity. return value is the same as for CompStr.}
4140: Func Copy2Symb(const S:Str; Symb: Char):Str;
4141: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4142: Func Copy2SymbDel(var S:Str; Symb: Char):Str;
4143: {Copy2SymbDel returns substr of str S from first character Symb and removes substring from S.}
4144: Func Copy2Space(const S:Str):Str;
4145: { Copy2Symb returns a substring of a string S from begining to first white space. }
4146: Func Copy2SpaceDel(var S:Str):Str;
4147: { Copy2SpaceDel returns a substring of a string S from begining to first
4148:   white space and removes this substring from S. }
4149: Func AnsiProperCase(const S:Str; const WordDelims: TCharSet):Str;
4150: { Returns string, with the first letter of each word in uppercase,
4151:   all other letters in lowercase. Words are delimited by WordDelims. }
4152: Func WordCount(const S:Str; const WordDelims: TCharSet): Int;
4153: { WordCount given a set of word delimiters, returns number of words in S. }
4154: Func WordPosition(const N: Int; const S:Str; const WordDelims: TCharSet): Int;
4155: { Given a set of word delimiters, returns start position of N'th word in S. }
4156: Func ExtractWord(N: Int; const S:Str; const WordDelims: TCharSet):Str;
4157: Func ExtractWordPos(N:Int; const S:str; const WordDelims:TCharSet;var Pos: Int):str;
4158: Func ExtractDelimited(N: Int; const S:Str; const Delims: TCharSet):Str;
4159: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4160:   delimiters, return the N'th word in S. }
4161: Func ExtractSubstr(const S:Str; var Pos: Int; const Delims: TCharSet):Str;
4162: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4163: Func IsWordPresent(const W, S:Str; const WordDelims: TCharSet):Bool;
4164: { IsWordPresent given set of word delimiters, returns True if word W is present in string S. }
4165: Func QuotedString(const S:Str; Quote: Char):Str;
4166: { QuotedString returns given string as a quoted string, using the provided Quote character. }
4167: Func ExtractQuotedString(const S:Str; Quote: Char):Str;
4168: { ExtractQuotedString removes the Quote characters from beginning and end of a quoted string,
4169:    and reduces pairs of Quote characters within the quoted string to a single character. }
4170: Func FindPart(const HelpWilds, InputStr:Str): Int;
4171: {FindPart compares a string with '?' and another,returns position of HelpWilds in InputStr. }
4172: Func IsWild(InputStr, Wilds:Str; IgnoreCase:Bool):Bool;
4173: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4174: Func XorString(const Key, Src: ShortString): ShortString;
4175: Func XorEncode(const Key, Source:Str):Str;
4176: Func XorDecode(const Key, Source:Str):Str;
4177: { ** Command line routines ** }
4178: {$IFNDEF COMPILER4_UP}
4179: Func FindCmdLineSwitch(const Switch:str;SwitchChars TCharSet;IgnoreCase: Bool):Bool;
4180: {$ENDIF}
4181: Func GetCmdLineArg(const Switch:Str; SwitchChars: TCharSet):Str;
4182: { ** Numeric string handling routines ** }
4183: Func Numb2USA(const S:Str):Str;
4184: { Numb2USA converts numeric string S to USA-format. }
4185: Func Dec2Hex(N: Longint; A: Byte):Str;
4186: Func D2H(N: Longint; A: Byte):Str;
4187: { Dec2Hex converts the given value to a hexadecimal string representation
4188:   with the minimum number of digits (A) specified. }
4189: Func Hex2Dec(const S:Str): Longint;
4190: Func H2D(const S:Str): Longint;
4191: { Hex2Dec converts the given hexadecimal string to the corresponding Int value. }
4192: Func Dec2Numb(N: Longint; A, B: Byte):Str;
4193: { Dec2Numb converts the given value to a string representation with the
4194:   base equal to B and with the minimum number of digits (A) specified. }
4195: Func Numb2Dec(S:Str; B: Byte): Longint;
4196: { Numb2Dec converts given B-based numeric string to the corresponding Int value. }
4197: Func IntToBin(Value: Longint; Digits, Spaces: Int):Str;
4198: {IntToBin converts given value to bin str representation with min number of digits specified.}
```

```
4199: Func IntToRoman(Value: Longint):Str;
4200: { IntToRoman converts the given value to a roman numeric string representation. }
4201: Func RomanToInt(const S:Str): Longint;
4202: { RomanToInt converts the given string to an Int value. If the string
4203:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4204: Proc I64ToCardinals(I: Int64; var LowPart, HighPart:Card);
4205: Proc CardinalsToI64(var I: Int64; const LowPart, HighPart:Card);
4206: ********************************** JvFileUtil;**********************************
4207: Proc CopyFileJ(const FileName, DestName:Str; ProgressControl: TControl);
4208: Proc CopyFileEx(const FileName,DestName:str;OverwriteReadOnly,ShellDialog:Bool;ProgressControl:TControl);
4209: Proc MoveFile(const FileName, DestName: TFileName);
4210: Proc MoveFileEx(const FileName, DestName: TFileName; ShellDialog:Bool);
4211: {$IFDEF COMPILER4_UP}
4212: Func GetFileSize(const FileName:Str): Int64;
4213: {$ELSE}
4214: Func GetFileSize(const FileName:Str): Longint;
4215: {$ENDIF}
4216: Func FileDateTime(const FileName:Str): TDateTime;
4217: Func HasAttr(const FileName:Str; Attr: Int):Bool;
4218: Func DeleteFiles(const FileMask:Str):Bool;
4219: Func DeleteFilesEx(const FileMasks: array of string):Bool;
4220: Func ClearDir(const Path:Str; Delete:Bool):Bool;
4221: Func NormalDir(const DirName:Str):Str;
4222: Func RemoveBackSlash(const DirName:Str):Str;
4223: Func ValidFileName(const FileName:Str):Bool;
4224: Func DirExists(Name:Str):Bool;
4225: Proc ForceDirectories(Dir:Str);
4226: Func FileLock(Handle: Int; Offset, LockSize: Longint): Int;
4227:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4228: {$IFDEF COMPILER4_UP}
4229: Func FileLock(Handle:Int; Offset, LockSize: Int64): Int; overload;
4230: {$ENDIF}
4231: Func FileUnlock(Handle:Int; Offset, LockSize: Longint): Int;
4232: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4233: {$IFDEF COMPILER4_UP}
4234: Func FileUnlock(Handle:Int; Offset, LockSize: Int64): Int; overload;
4235: {$ENDIF}
4236: Func GetTempDir:Str;
4237: Func GetWindowsDir:Str;
4238: Func GetSystemDir:Str;
4239: Func BrowseDirectory(var AFolderName:str; const DlgText:str; AHelpContext:THelpContext):Boolean;
4240: {$IFDEF WIN32}
4241: Func BrowseComputer(var ComputerName:str;const DlgText:str;AHelpContext:THelpContext):Bool;
4242: Func ShortToLongFileName(const ShortName:Str):Str;
4243: Func ShortToLongPath(const ShortName:Str):Str;
4244: Func LongToShortFileName(const LongName:Str):Str;
4245: Func LongToShortPath(const LongName:Str):Str;
4246: Proc CreateFileLink(const FileName, DisplayName:Str; Folder: Int);
4247: Proc DeleteFileLink(const DisplayName:Str; Folder: Int);
4248: {$ENDIF WIN32}
4249: {$IFNDEF COMPILER3_UP}
4250: Func IsPathDelimiter(const S:Str; Index: Int):Bool;
4251: {$ENDIF}
4252: Func CreateCalculatorForm(AOwner:TComponent;AHelpContext:THelpContext):TJvCalculatorForm
4253: Func CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4254: Func CreatePopupCalculator( AOwner : TComponent) : TWinControl
4255: Proc SetupPopupCalculator(PopupCalc:TWinControl;APrecision Byte; ABeepOnError: Bool)
4256:
4257: //***************************Proc SIRegister_VarHlpr(CL: TPSPascalCompiler);
4258:  Proc VariantClear( var V : Variant)
4259:  Proc VariantArrayRedim( var V : Variant; High : Int)
4260:  Proc VariantCast( const src : Variant; var dst : Variant; vt : Int)
4261:  Proc VariantCpy(const src:Variant;var dst:Variant)
4262:  Proc VariantAdd(const src:Variant;var dst:Variant)
4263:  Proc VariantSub(const src:Variant;var dst:Variant)
4264:  Proc VariantMul(const src:Variant;var dst:Variant)
4265:  Proc VariantDiv(const src:Variant;var dst:Variant)
4266:  Proc VariantMod(const src:Variant;var dst:Variant)
4267:  Proc VariantAnd(const src:Variant;var dst:Variant)
4268:  Proc VariantOr( const src:Variant;var dst:Variant)
4269:  Proc VariantXor(const src:Variant;var dst:Variant)
4270:  Proc VariantShl(const src:Variant;var dst:Variant)
4271:  Proc VariantShr(const src:Variant;var dst:Variant)
4272:  Func VariantAdd2(const V1:Variant; const V2:Variant):Variant
4273:  Func VariantSub2(const V1:Variant; const V2:Variant):Variant
4274:  Func VariantMul2(const V1:Variant; const V2:Variant):Variant
4275:  Func VariantDiv2(const V1:Variant; const V2:Variant):Variant
4276:  Func VariantMod2(const V1:Variant; const V2:Variant):Variant
4277:  Func VariantAnd2(const V1:Variant; const V2:Variant):Variant
4278:  Func VariantOr2( const V1:Variant; const V2:Variant):Variant
4279:  Func VariantXor2(const V1: Variant; const V2: Variant): Variant
4280:  Func VariantShl2(const V1  Variant; const V2: Variant): Variant
4281:  Func VariantShr2(const V1: Variant; const V2: Variant): Variant
4282:  Func VariantNot(const V1: Variant) : Variant
4283:  Func VariantNeg(const V1: Variant) : Variant
4284:  Func VariantGetElement( const V : Variant; i1 : Int) : Variant;
4285:  Func VariantGetElement1(const V : Variant; i1, i2 : Int) : Variant;
4286:  Func VariantGetElement2(const V : Variant; i1, i2, i3 : Int) : Variant;
4287:  Func VariantGetElement3(const V : Variant; i1, i2, i3, i4 : Int) : Variant;
```

```
4288:  Func VariantGetElement4(const V : Variant; i1, i2, i3, i4, i5 : Int) : Variant;
4289:  Proc VariantPutElement(var V : Variant; const data : Variant; i1 : Int);
4290:  Proc VariantPutElement1(var V : Variant; const data : Variant; i1, i2 : Int);
4291:  Proc VariantPutElement2(var V : Variant; const data : Variant; i1, i2, i3 : Int);
4292:  Proc VariantPutElement3(var V: Variant; const data : Variant; i1, i2, i3, i4 : Int);
4293:  Proc VariantPutElement4(var V: Variant; const data : Variant; i1, i2, i3, i4, i5 : Int);
4294: end;
4295:
4296: ************************************unit uPSI_JvgUtils;************************************
4297: Func IsEven(I: Int):Bool;
4298: Func InchesToPixels(DC: HDC; Value: Single; IsHorizontal:Bool): Int;
4299: Func CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal:Bool): Int;
4300: Proc SwapInt(var I1, I2: Int);
4301: Func Spaces(Count: Int):Str;
4302: Func DupStr(const Str:Str; Count: Int):Str;
4303: Func DupChar(C: Char; Count: Int):Str;
4304: Proc Msg(const AMsg:Str);
4305: Func RectW(R: TRect): Int;
4306: Func RectH(R: TRect): Int;
4307: Func IncColor(AColor: Longint; AOffset: Byte): Longint;
4308: Func DecColor(AColor: Longint; AOffset: Byte): Longint;
4309: Func IsItAFilledBitmap(Bmp: TBitmap):Bool;
4310: Proc DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text:Str;
4311:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4312: Proc DrawTextInRect(DC:HDC;R:TRect;const Text:str;Style:TglTextStyle;Fnt:TFont;Flags:UINT);
4313: Proc ExtTextOutExt(DC: HDC; X, Y: Int; R: TRect; const Text:Str;
4314:   Style: TglTextStyle; ADelineated, ASupress3D:Bool; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4315: Proc DrawBox(DC:HDC;var R:TRect;Style:TglBoxStyle;BackgrColor:Longint;ATransparent: Bool);
4316: Func DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4317:   BevelInner,BevelOuter:TPanelBevel;Bold:Boolean;BackgrColor:Longint;ATransparent:Bool):TRect;
4318: Proc GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Int);
4319: Proc ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4320: Proc DrawBitmapExt(DC: HDC; { DC - background & result}
4321:   SourceBitmap: TBitmap; R: TRect; X, Y: Int; //...X,Y _in_ rect!
4322:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4323:   ATransparent:Bool; TransparentColor: TColor; DisabledMaskColor: TColor);
4324: Proc CreateBitmapExt(DC: HDC; { DC - background & result}
4325:   SourceBitmap: TBitmap; R: TRect; X, Y: Int; //...X,Y _in_ rect!
4326:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4327:   ATransparent:Bool; TransparentColor: TColor; DisabledMaskColor: TColor);
4328: Proc BringParentWindowToTop(Wnd: TWinControl);
4329: Func GetParentForm(Control: TControl): TForm;
4330: Proc GetWindowImageFrom(Control:TWinContrl;X,Y:Int;ADrawSelf,ADrawChildWindows:Bool;DC:HDC)
4331: Proc GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows:Bool; DC: HDC);
4332: Proc GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4333: Func CreateRotatedFont(F: TFont; Escapement: Int): HFONT;
4334: Func FindMainWindow(const AWndClass, AWndTitle:Str): THandle;
4335: Proc CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4336: Func CalcMathString(AExpression:Str): Single;
4337: Func IIF(AExpression:Bool; IfTrue, IfFalse: Variant): Variant; overload;
4338: Func IIF(AExpression:Bool; const IfTrue, IfFalse:Str):Str; overload;
4339: Func GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4340: Proc TypeStringOnKeyboard(const S:Str);
4341: Func NextStringGridCell( Grid: TStringGrid ):Bool;
4342: Proc DrawTextExtAligned(Canvas:TCanvas;const Text:str;R:TRect;Alignment:TglAlignment;WordWrap:Bool;
4343: Proc LoadComponentFromTextFile(Component: TComponent; const FileName:Str);
4344: Proc SaveComponentToTextFile(Component: TComponent; const FileName:Str);
4345: Func ComponentToString(Component: TComponent):Str;
4346: Proc StringToComponent(Component: TComponent; const Value:Str);
4347: Func PlayWaveResource(const ResName:Str):Bool;
4348: Func UserName:Str;
4349: Func ComputerName:Str;
4350: Func CreateIniFileName:Str;
4351: Func ExpandString(const Str:Str; Len: Int):Str;
4352: Func Transliterate(const Str:Str; RusToLat:Bool):Str;
4353: Func IsSmallFonts:Bool;
4354: Func SystemColorDepth: Int;
4355: Func GetFileTypeJ(const FileName:Str): TglFileType;
4356: Func GetFileType( hFile : THandle) : DWORD;
4357: Func FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass):TControl;
4358: Func StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4359:
4360: { ************************************Utility routines of unit classes}
4361: Func LineStart(Buffer, BufPos: PChar): PChar
4362: Func ExtractStrings(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strings:TStrings):Int
4363: Func TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4364: Proc RegisterClass( AClass : TPersistentClass)
4365: Proc RegisterClasses( AClasses : array of TPersistentClass)
4366: Proc RegisterClassAlias( AClass : TPersistentClass; const Alias :Str)
4367: Proc UnRegisterClass( AClass : TPersistentClass)
4368: Proc UnRegisterClasses( AClasses : array of TPersistentClass)
4369: Proc UnRegisterModuleClasses( Module : HMODULE)
4370: Func FindGlobalComponent( const Name :Str) : TComponent
4371: Func IsUniqueGlobalComponentName( const Name :Str) :Bool
4372: Func InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) :Bool
4373: Func InitComponentRes( const ResName :Str; Instance : TComponent) :Bool
4374: Func ReadComponentRes( const ResName :Str; Instance : TComponent) : TComponent
4375: Func ReadComponentResEx( HInstance : THandle; const ResName :Str) : TComponent
```

```
4376: Func ReadComponentResFile( const FileName :Str; Instance : TComponent) : TComponent
4377: Proc WriteComponentResFile(const FileName:Str; Instance : TComponent)
4378: Proc GlobalFixupReferences
4379: Proc GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4380: Proc GetFixupInstanceNames(Root:TComponent;const ReferenceRootName string;Names:TStrings)
4381: Proc RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName :Str)
4382: Proc RemoveFixupReferences( Root : TComponent; const RootName :Str)
4383: Proc RemoveFixups( Instance : TPersistent)
4384: Func FindNestedComponent( Root : TComponent; const NamePath :Str) : TComponent
4385: Proc BeginGlobalLoading
4386: Proc NotifyGlobalLoading
4387: Proc EndGlobalLoading
4388: Func GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4389: Func GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4390:  // AddTypeS('TWndMethod', 'Proc (var Message : TMessage)
4391:  //Func MakeObjectInstance( Method : TWndMethod) : Pointer
4392:  Proc FreeObjectInstance( ObjectInstance : Pointer)
4393: // Func AllocateHWnd( Method : TWndMethod) : HWND
4394:  Proc DeallocateHWnd( Wnd : HWND)
4395:  Func AncestorIsValid( Ancestor:TPersistent; Root, RootAncestor: TComponent) :Bool
4396: ****************************unit uPSI_SqlTimSt and DB;************************************
4397: Proc VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4398: Func VarSQLTimeStampCreate3: Variant;
4399: Func VarSQLTimeStampCreate2( const AValue :Str) : Variant;
4400: Func VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4401: Func VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4402: Func VarSQLTimeStamp : TVarType
4403: Func VarIsSQLTimeStamp( const aValue : Variant) :Bool;
4404: Func LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp): TSQLTimeStamp //beta
4405: Func UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp): TSQLTimeStamp //beta
4406: Func VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4407: Func SQLTimeStampToStr( const Format :Str; DateTime : TSQLTimeStamp):Str
4408: Func SQLDayOfWeek( const DateTime : TSQLTimeStamp) : Int
4409: Func DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4410: Func SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4411: Func TryStrToSQLTimeStamp( const S :Str; var TimeStamp : TSQLTimeStamp): Bool
4412: Func StrToSQLTimeStamp( const S :Str) : TSQLTimeStamp
4413: Proc CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4414: Func ExtractFieldName( const Fields :Str; var Pos : Int) :Str;
4415: Func ExtractFieldName( const Fields : WideString; var Pos : Int) : WideString;
4416:  //'Proc RegisterFields( const FieldClasses : array of TFieldClass)
4417: Proc DatabaseError( const Message : WideString; Component : TComponent)
4418: Proc DatabaseErrorFmt(const Message:WIdeString;const Args:array of const;Component:TComponent)
4419: Proc DisposeMem( var Buffer, Size : Int)
4420: Func BuffersEqual( Buf1, Buf2 : Pointer; Size : Int) :Bool
4421: Func GetFieldProperty(DataSet:TDataSet;Control:TComponent;const FieldName:WideStr):TField
4422: Func VarTypeToDataType( VarType : Int) : TFieldType
4423: ******************************unit JvStrings;****************************************
4424: {template functions}
4425: Func ReplaceFirst(const SourceStr, FindStr, ReplaceStr:Str):Str;
4426: Func ReplaceLast(const SourceStr, FindStr, ReplaceStr:Str):Str;
4427: Func InsertLastBlock(var SourceStr:Str; BlockStr:Str):Bool;
4428: Func RemoveMasterBlocks(const SourceStr:Str):Str;
4429: Func RemoveFields(const SourceStr:Str):Str;
4430: {http functions}
4431: Func URLEncode(const Value:Ansistr):Ansistr;// Converts string To A URLEncoded string
4432: Func URLDecode(const Value:Ansistr):Ansistr;// Converts string From A URLEncoded string
4433: {set functions}
4434: Proc SplitSet(AText:Str; AList: TStringList);
4435: Func JoinSet(AList: TStringList):Str;
4436: Func FirstOfSet(const AText:Str):Str;
4437: Func LastOfSet(const AText:Str):Str;
4438: Func CountOfSet(const AText:Str): Int;
4439: Func SetRotateRight(const AText:Str):Str;
4440: Func SetRotateLeft(const AText:Str):Str;
4441: Func SetPick(const AText:Str; AIndex: Int):Str;
4442: Func SetSort(const AText:Str):Str;
4443: Func SetUnion(const Set1, Set2:Str):Str;
4444: Func SetIntersect(const Set1, Set2:Str):Str;
4445: Func SetExclude(const Set1, Set2:Str):Str;
4446: {replace any <,> etc by &lt; &gt;}
4447: Func XMLSafe(const AText:Str):Str;
4448: {simple hash, Result can be used in Encrypt}
4449: Func Hash(const AText:Str): Int;
4450: { Base64 encode and decode a string }
4451: Func B64Encode(const S: Ansistr): Ansistr;
4452: Func B64Decode(const S: Ansistr): Ansistr;
4453: {Basic encryption from a Borland Example}
4454: Func Encrypt(const InString: Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4455: Func Decrypt(const InString: Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4456: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4457: Func EncryptB64(const InString:Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4458: Func DecryptB64(const InString:Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4459: Proc CSVToTags(Src, Dst: TStringList);
4460: // converts a csv list to a tagged string list
4461: Proc TagsToCSV(Src, Dst: TStringList);
4462: // converts a tagged string list to a csv list
4463: // only fieldnames from the first record are scanned ib the other records
4464: Proc ListSelect(Src, Dst: TStringList; const AKey, AValue:Str);
```

```
4465: {selects akey=avalue from Src and returns recordset in Dst}
4466: Proc ListFilter(Src: TStringList; const AKey, AValue:Str);
4467: {filters Src for akey=avalue}
4468: Proc ListOrderBy(Src: TStringList; const AKey:Str; Numeric: Bool);
4469: {orders a tagged Src list by akey}
4470: Func PosStr(const FindString, SourceString:Str;
4471:   StartPos: Int = 1): Int;
4472: { PosStr searches the first occurrence of a substring FindString in a string
4473:   given by SourceString with case sensitivity (upper and lower case characters
4474:   are differed). This Func returns the index value of the first character
4475:   of a specified substring from which it occurs in a given string starting with
4476:   StartPos character index. If a specified substring is not found Q_PosStr
4477:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings from www.torry.ru). }
4478: Func PosStrLast(const FindString, SourceString:Str): Int;
4479: {finds the last occurance}
4480: Func LastPosChar(const FindChar: Char; SourceString:Str): Int;
4481: Func PosText(const FindString, SourceString:Str; StartPos: Int = 1): Int;
4482: { PosText searches the first occurrence of a substring FindString in a string
4483:   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
4484: Func PosTextLast(const FindString, SourceString:Str): Int;
4485: {finds last occurance}
4486: Func NameValuesToXML(const AText:Str):Str;
4487: {$IFDEF MSWINDOWS}
4488: Proc LoadResourceFile(AFile:Str; MemStream: TMemoryStream);
4489: {$ENDIF MSWINDOWS}
4490: Proc DirFiles(const ADir, AMask:Str; AFileList: TStringList);
4491: Proc RecurseDirFiles(const ADir:Str; var AFileList: TStringList);
4492: Proc RecurseDirProgs(const ADir:Str; var AFileList: TStringList);
4493: Proc SaveString(const AFile, AText:Str);
4494: Proc SaveStringasFile( const AFile, AText :Str)
4495: Func LoadStringJ(const AFile:Str):Str;
4496: Func LoadStringofFile( const AFile :Str) :Str
4497: Proc SaveStringtoFile( const AFile, AText :Str)
4498: Func LoadStringfromFile( const AFile :Str) :Str
4499: Func HexToColor(const AText:Str): TColor;
4500: Func UppercaseHTMLTags(const AText:Str):Str;
4501: Func LowercaseHTMLTags(const AText:Str):Str;
4502: Proc GetHTMLAnchors(const AFile:Str; AList: TStringList);
4503: Func RelativePath(const ASrc, ADst:Str):Str;
4504: Func GetToken(var Start: Int; const SourceText:Str):Str;
4505: Func PosNonSpace(Start: Int; const SourceText:Str): Int;
4506: Func PosEscaped(Start: Int; const SourceText, FindText:Str; EscapeChar: Char): Int;
4507: Func DeleteEscaped(const SourceText:Str; EscapeChar: Char):Str;
4508: Func BeginOfAttribute(Start: Int; const SourceText:Str): Int;
4509: // parses the beginning of an attribute: space + alpha character
4510: Func ParseAttribute(var Start:Int;const SourceText:str; var AName,AValue:str):Bool;
4511: //parses a name="value" attrib from Start;returns 0 when not found or else position behind attribute
4512: Proc ParseAttributes(const SourceText:Str; Attributes: TStrings);
4513: // parses all name=value attributes to the attributes TStringList
4514: Func HasStrValue(const AText, AName:Str; var AValue:Str):Bool;
4515: // checks if a name="value" pair exists and returns any value
4516: Func GetStrValue(const AText, AName, ADefault:Str):Str;
4517: // retrieves string value from a line like:
4518: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4519: // returns ADefault when not found
4520: Func GetHTMLColorValue(const AText, AName:Str; ADefault: TColor):TColor;
4521: // same for a color
4522: Func GetIntValue(const AText, AName:Str; ADefault: Int): Int;
4523: // same for an Int
4524: Func GetFloatValue(const AText, AName:Str; ADefault: Extended):Extended;
4525: // same for a float
4526: Func GetBoolValue(const AText, AName:Str):Bool;
4527: // same for Boolean but without default
4528: Func GetValue(const AText, AName:Str):Str;
4529: //retrieves string value from a line like:name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4530: Proc SetValue(var AText:Str; const AName, AValue:Str);
4531: // sets a string value in a line
4532: Proc DeleteValue(var AText:Str; const AName:Str);
4533: // deletes a AName="value" pair from AText
4534: Proc GetNames(AText:Str; AList: TStringList);
4535: // get a list of names from a string with name="value" pairs
4536: Func GetHTMLColor(AColor: TColor):Str;
4537: // converts a color value to the HTML hex value
4538: Func BackPosStr(Start: Int; const FindString, SourceString:Str): Int;
4539: // finds a string backward case sensitive
4540: Func BackPosText(Start: Int; const FindString, SourceString:Str): Int;
4541: // finds a string backward case insensitive
4542: Func PosRangeStr(Start: Int; const HeadString, TailString, SourceString:Str;
4543:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4544: // finds a text range, e.g. <TD>....</TD> case sensitive
4545: Func PosRangeText(Start: Int; const HeadString, TailString, SourceString:Str;
4546:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4547: // finds a text range, e.g. <TD>....</td> case insensitive
4548: Func BackPosRangeStr(Start: Int; const HeadString, TailString, SourceString:Str;
4549:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4550: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4551: Func BackPosRangeText(Start: Int; const HeadString, TailString, SourceString:str;
```

```
4552:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4553: // finds a text range backward, e.g. <TD>....</td> case insensitive
4554: Func PosTag(Start: Int; SourceString:Str; var RangeBegin: Int;
4555:   var RangeEnd: Int):Bool;
4556: // finds a HTML or XML tag:  <....>
4557: Func InnerTag(Start: Int; const HeadString, TailString, SourceString:Str;
4558:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4559: // finds the innertext between opening and closing tags
4560: Func Easter(NYear: Int): TDateTime;
4561: // returns the easter date of a year.
4562: Func GetWeekNumber(Today: TDateTime):Str;
4563: //gets a datecode. Returns year and weeknumber in format: YYWW
4564: Func ParseNumber(const S:Str): Int;
4565: // parse number returns the last position, starting from 1
4566: Func ParseDate(const S:Str): Int;
4567: // parse a SQL style data string from positions 1, // starts and ends with #
4568:
4569: **************************unit JvJCLUtils;************************************
4570: PathDelim','String').SetString( '\;
4571:  DriveDelim','String').SetString( ':;
4572:  PathSep','String').SetString( ';;
4573:  AllFilesMask','String').SetString( '*.*;
4574:  PathDelim','String').SetString( '/;
4575:  AllFilesMask','String').SetString( '*;
4576:  NullHandle','LongInt').SetInt( 0);
4577:  USDecimalSeparator','String').SetString( '.;
4578:   ClassN(CL.FindClass('TOBJECT'),'EJvConvertError;
4579:   TypeS('TFileTime', 'Int;
4580:  // TypeS('TFormatSettings', 'record DecimalSeparator : Char; end;
4581:  Func SendRectMessage(Handle: THandle; Msg : Int; wParam : longint; var R : TRect): Int;
4582:  //Func SendStructMessage(Handle:THandle;Msg:Int;wParam:WPARAM;var Data): Int;
4583:  Func ReadCharsFromStream(Stream: TStream; var Buf:array of PChar;BufSize:Int):Int;
4584:  Func WriteStringToStream(Stream: TStream; const Buf: Ansistr;BufSize:Int): Int;
4585:  Func UTF8ToString( const S : UTF8String) :Str;
4586:  //ConstantN('DefaultDateOrder','').SetString('doDMY;
4587:  ConstantN('CenturyOffset','Byte').SetUInt( 60);
4588:
4589: Func VarIsInt(Value: Variant):Bool;
4590:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4591: {PosIdx returns index of first appearance of SubStr in Str.search starts at index "Index".}
4592: Func PosIdx(const SubStr, S:Str; Index: Int = 0): Int;
4593: Func PosIdxW(const SubStr, S: WideString; Index: Int = 0): Int;
4594: Func PosLastCharIdx(Ch: Char; const S:Str; Index: Int = 0): Int;
4595: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4596: Func GetWordOnPos(const S:Str; const P: Int):Str;
4597: Func GetWordOnPosW(const S: WideString; const P: Int): WideString;
4598: Func GetWordOnPos2(const S:Str; P: Int; var iBeg, iEnd: Int):Str;
4599: Func GetWordOnPos2W(const S: WideString; P: Int; var iBeg, iEnd: Int): WideString;
4600: { GetWordOnPosEx working like GetWordOnPos function, but
4601:   also returns Word position in iBeg, iEnd variables }
4602: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
4603: Func GetWordOnPosExW(const S: WideString; const P: Int; var iBeg, iEnd: Int): WideString;
4604: Func GetNextWordPosEx(const Text:Str; StartIndex: Int; var iBeg, iEnd: Int):Str;
4605: Func GetNextWordPosExW(const Text:WideString;StartIndex:Int; var iBeg,iEnd:Int):WideString;
4606: Proc GetEndPosCaret(const Text:Str; CaretX, CaretY: Int; var X, Y: Int);
4607: { GetEndPosCaret returns the caret position of the last char. For the position
4608:   after the last char of Text you must add 1 to the returned X value. }
4609: Proc GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Int;var X,Y:Int);
4610: { GetEndPosCaret returns the caret position of the last char. For the position
4611:   after the last char of Text you must add 1 to the returned X value. }
4612: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4613: Func SubStrBySeparator(const S:str;const Index:Int;const Separatr:str;StartIndex:Int=1):str;
4614: Func SubStrBySeparatorW(const S:WideString;const Index:Int;const
      Separator:WideString;StartIndex:Int:WideString;
4615: { SubStrEnd same to previous Func but Index numerated from the end of string }
4616: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
4617: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4618: Func SubWord(P: PChar; var P2: PChar):Str;
4619: Func CurrencyByWord(Value: Currency):Str;
4620: { GetLineByPos returns Line number,there symbol Pos is pointed.Lines separated with #13 symbol}
4621: Func GetLineByPos(const S:Str; const Pos: Int): Int;
4622: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4623: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
4624: Proc GetXYByPosW(const S: WideString; const Pos: Int; var X, Y: Int);
4625: { ReplaceString searches for all substrings, OldPattern,
4626:   in a string, S, and replaces them with NewPattern }
4627: Func ReplaceString(S:Str;const OldPattern,NewPattern:str;StartIndex:Int= 1):str;
4628: Func ReplaceStringW(S:WideString;const OldPattern,NewPattern:WideString;StartIndex:Int=1):WideString;
4629: { ConcatSep concatenate S1 and S2 strings with Separator. if S ='' then separator not included}
4630: Func ConcatSep(const S1, S2, Separator:Str):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4631: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4632: Func ConcatLeftSep(const S1,S2, Separator:str):str;{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF SUPPORTS_INLINE}
4633: { Next 4 Func for russian chars transliterating.
4634:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4635: Proc Dos2Win(var S: Ansistr);
4636: Proc Win2Dos(var S: Ansistr);
4637: Func Dos2WinRes(const S: Ansistr): Ansistr; inline; {$ENDIF SUPPORTS_INLINE}
4638: Func Win2DosRes(const S: Ansistr): Ansistr; inline; {$ENDIF SUPPORTS_INLINE}
4639: Func Win2Koi(const S: Ansistr): Ansistr;
```

```
4640: { FillString fills the string Buffer with Count Chars }
4641: Proc FillString(var Buffer:Str; Count: Int; const Value: Char); overload;
4642: Proc FillString1(var Buffer:Str; StartIndex,Count:Int; const Value: Char);overload;
4643: { MoveString copies Count Chars from Source to Dest }
4644: Proc MoveString(const Source:Str; var Dest:Str; Count: Int); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE} overload;
4645: Proc MoveString1(const Source:Str; SrcStartIdx: Int; var Dest:Str;
4646:   DstStartIdx: Int;Count:Int);{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}overload;
4647: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4648: Proc FillWideChar(var Buffer; Count: Int; const Value: WideChar);
4649: { MoveWideChar copies Count WideChars from Source to Dest }
4650: Proc MoveWideChar(const Source;var Dest;Count:Int);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF SUPPORTS_INLINE}
4651: { FillNativeChar fills Buffer with Count NativeChars }
4652: Proc FillNativeChar(var Buffer; Count: Int; const Value: Char);// D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4653: { MoveWideChar copies Count WideChars from Source to Dest }
4654: Proc MoveNativeChar(const Source; var Dest; Count: Int);// D2009 internal error {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE}
4655: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4656: Func IsSubString(const S:Str; StartIndex: Int; const SubStr:Str):Bool;
4657: { Spaces returns string consists on N space chars }
4658: Func Spaces(const N: Int):Str;
4659: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4660: Func AddSpaces(const S:Str; const N: Int):Str;
4661: Func SpacesW(const N: Int): WideString;
4662: Func AddSpacesW(const S: WideString; const N: Int): WideString;
4663: { Func LastDateRUS for russian users only }
4664: { returns date relative to current date: 'äâà äíÿ íàçàä' }
4665: Func LastDateRUS(const Dat: TDateTime):Str;
4666: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4667: Func CurrencyToStr(const Cur: Currency):Str;
4668: { HasChar returns True, if Char, Ch, contains in string, S }
4669: Func HasChar(const Ch: Char; const S:Str):Bool;
4670: Func HasCharW(const Ch: WideChar; const S: WideString):Bool; inline; {$ENDIF SUPPORTS_INLINE}
4671: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
4672: {$IFNDEF COMPILER12_UP}
4673: Func CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4674: {$ENDIF ~COMPILER12_UP}
4675: Func CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF SUPPORTS_INLINE}
4676: Func CountOfChar(const Ch: Char; const S:Str): Int;
4677: Func DefStr(const S:str; Default:str):str; {$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4678: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4679: Func StrLICompW2(S1, S2: PWideChar; MaxLen: Int): Int;
4680: Func StrPosW(S, SubStr: PWideChar): PWideChar;
4681: Func StrLenW(S: PWideChar): Int;
4682: Func TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4683: Func TrimLeftW(const S:WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4684: Func TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4685: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4686: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4687:  Func GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4688: Func GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4689: Proc SetBitmapPixelFormat(Bitmap: TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod)
4690: Func BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4691: Proc GrayscaleBitmap( Bitmap : TBitmap)
4692: Func BitmapToMemory( Bitmap : TBitmap; Colors : Int) : TStream
4693: Proc SaveBitmapToFile( const Filename :Str; Bitmap : TBitmap; Colors : Int)
4694: Func ScreenPixelFormat : TPixelFormat
4695: Func ScreenColorCount : Int
4696: Proc TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4697: Func ZoomImage( ImageW, ImageH, MaxW, MaxH : Int; Stretch :Bool): TPoint
4698: //  SIRegister_TJvGradient(CL);
4699:
4700: {***************************************** files routines}
4701: Proc SetDelimitedText(List: TStrings; const Text:Str; Delimiter: Char);
4702: const
4703:   {$IFDEF MSWINDOWS}
4704:   DefaultCaseSensitivity = False;
4705:   {$ENDIF MSWINDOWS}
4706:   {$IFDEF UNIX}
4707:   DefaultCaseSensitivity = True;
4708:   {$ENDIF UNIX}
4709: { GenTempFileName returns temporary file name on
4710:   drive, there FileName is placed }
4711: Func GenTempFileName(FileName:Str):Str;
4712: { GenTempFileNameExt same to previous function, but
4713:   returning filename has given extension, FileExt }
4714: Func GenTempFileNameExt(FileName:Str; const FileExt:Str):Str;
4715: { ClearDir clears folder Dir }
4716: Func ClearDir(const Dir:Str):Bool;
4717: { DeleteDir clears and than delete folder Dir }
4718: Func DeleteDir(const Dir:Str):Bool;
4719: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4720: Func FileEquMask(FileName,Mask:TFileName;CaseSensitive:Boolean=DefaultCaseSensitivity):Bool;
4721: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4722:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4723: Func FileEquMasks(FileName, Masks: TFileName;
4724:   CaseSensitive:Bool = DefaultCaseSensitivity):Bool;
4725: Func DeleteFiles(const Folder: TFileName; const Masks:Str):Bool;
```

```
4726: {$IFDEF MSWINDOWS}
4727: { LZFileExpand expand file, FileSource,
4728:   into FileDest. Given file must be compressed, using MS Compress program }
4729: Func LZFileExpand(const FileSource, FileDest:Str):Bool;
4730: {$ENDIF MSWINDOWS}
4731: { FileGetInfo fills SearchRec record for specified file attributes}
4732: Func FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec):Bool;
4733: { HasSubFolder returns True, if folder APath contains other folders }
4734: Func HasSubFolder(APath: TFileName):Bool;
4735: { IsEmptyFolder returns True, if there are no files or
4736:   folders in given folder, APath}
4737: Func IsEmptyFolder(APath: TFileName):Bool;
4738: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4739: Func AddSlash(const Dir: TFileName):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4740: { AddPath returns FileName with Path, if FileName not contain any path }
4741: Func AddPath(const FileName, Path: TFileName): TFileName;
4742: Func AddPaths(const PathList, Path:Str):Str;
4743: Func ParentPath(const Path: TFileName): TFileName;
4744: Func FindInPath(const FileName, PathList:Str): TFileName;
4745: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4746: Func DeleteReadOnlyFile(const FileName: TFileName):Bool;
4747: { HasParam returns True, if program running with specified parameter, Param }
4748: Func HasParam(const Param:Str):Bool;
4749: Func HasSwitch(const Param:Str):Bool;
4750: Func Switch(const Param:Str):Str;
4751: { ExePath returns ExtractFilePath(ParamStr(0)) }
4752: Func ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4753: Func CopyDir(const SourceDir, DestDir: TFileName):Bool;
4754: //Func FileTimeToDateTime(const FT: TFileTime): TDateTime;
4755: Proc FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4756: Func MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4757: {**** Graphic routines }
4758: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4759: Func IsTTFontSelected(const DC: HDC):Bool;
4760: Func KeyPressed(VK: Int):Bool;
4761: Func isKeypressed:Bool;  //true if key on memo2 (shell output) is pressed
4762: { TrueInflateRect inflates rect in other method, than InflateRect API Func }
4763: Func TrueInflateRect(const R: TRect; const I: Int): TRect;
4764: {**** Color routines }
4765: Proc RGBToHSV(R, G, B: Int; var H, S, V: Int);
4766: Func RGBToBGR(Value:Card):Card;
4767: //Func ColorToPrettyName(Value: TColor):Str;
4768: //Func PrettyNameToColor(const Value:Str): TColor;
4769: {**** other routines }
4770: Proc SwapInt(var Int1, Int2:Int);{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4771: Func IntPower(Base, Exponent:Int): Int;
4772: Func ChangeTopException(E: TObject):TObject;//Linux version writes error message to ErrOutput
4773: Func StrToBool(const S:Str):Bool;
4774: Func Var2Type(V: Variant; const DestVarType: Int): Variant;
4775: Func VarToInt(V: Variant): Int;
4776: Func VarToFloat(V: Variant): Double;
4777: {following functions are not documented cause not work properly sometimes,so do not use them}
4778: // (rom) ReplaceStrings1, GetSubStr removed
4779: Func GetLongFileName(const FileName:Str):Str;
4780: Func FileNewExt(const FileName, NewExt: TFileName): TFileName;
4781: Func GetParameter:Str;
4782: Func GetComputerID:Str;
4783: Func GetComputerName:Str;
4784: {**** string routines }
4785: { ReplaceAllStrings searches for all substrings, Words,
4786:   in a string, S, and replaces them with Frases with the same Index. }
4787: Func ReplaceAllStrings(const S:Str; Words, Frases: TStrings):Str;
4788: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4789:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with same
      Index, and then update NewSelStart variable }
4790: Func ReplaceStrings(const S:str;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):str;
4791: { CountOfLines calculates the lines count in a string, S,
4792:   each line must be separated from another with CrLf sequence }
4793: Func CountOfLines(const S:Str): Int;
4794: { DeleteLines deletes all lines from strings which in the words,  words.
4795:   The word of will be deleted from strings. }
4796: Proc DeleteOfLines(Ss: TStrings; const Words: array of string);
4797: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4798:   Lines contained only spaces also deletes. }
4799: Proc DeleteEmptyLines(Ss: TStrings);
4800: { SQLAddWhere addes or modifies existing where-statement, where,
4801:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4802:   it must be started on the begining of any line }
4803: Proc SQLAddWhere(SQL: TStrings; const Where:Str);
4804: {**** files routines - }
4805: {$IFDEF MSWINDOWS}
4806: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4807:   Resource can be compressed using MS Compress program}
4808: Func ResSaveToFile(const Typ,Name:str;const Compressed:Bool;const FileName:str):Bool;
4809: Func ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:str):Bool;
4810: Func ResSaveToString(Instance: HINST;const Typ,Name:str; var S:str):Boolean;
4811: {$ENDIF MSWINDOWS}
4812: { IniReadSection read section, Section, from ini-file,
4813:   IniFileName, into strings, Ss.This Func reads ALL strings from specified section.
```

```
  Note: TIninFile.ReadSection Func reads only strings with '=' symbol.}
Func IniReadSection(const IniFileName:TFileName;const Section:str; Ss: TStrings): Bool;
{ LoadTextFile load text file, FileName, into string }
Func LoadTextFile(const FileName: TFileName):Str;
Proc SaveTextFile(const FileName: TFileName; const Source:Str);
{ReadFolder reads fileslist from diskfolder,Folder,that are equal to mask,Mask,into strings, FileList}
Func ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Int;
Func ReadFolders(const Folder: TFileName; FolderList: TStrings): Int;
{ RATextOut same with TCanvas.TextOut procedure,but can clipping drawing with rectangle,RClip.}
Proc RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S:Str);
{ RATextOutEx same with RATextOut function, but can calculate needed height for correct output}
Func RATextOutEx(Canvas:TCanvas;const R,RClip:TRect;const S:str;const CalcHeight:Bool):Int;
{ RATextCalcHeight calculate needed height for
  correct output, using RATextOut or RATextOutEx functions }
Func RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S:Str): Int;
{ Cinema draws some visual effect }
Proc Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
{ Roughed fills rect with special 3D pattern }
Proc Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert:Bool);
{ 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 }
Func BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Int): TBitmap;
{ TextWidth calculate text with for writing using standard desktop font }
Func TextWidth(const AStr:Str): Int;
{ TextHeight calculate text height for writing using standard desktop font }
Func TextHeight(const AStr:Str): Int;
Proc SetChildPropOrd(Owner: TComponent; const PropName:Str; Value: Longint);
Proc Error(const Msg:Str);
Proc ItemHtDrawEx(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;
  const HideSelColor:Bool; var PlainItem:Str; var Width: Int; CalcWidth:Bool);
{ex.Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>'}
Func ItemHtDraw(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text:Str; const HideSelColor:Bool):Str;
Func ItemHtWidth(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text:Str; const HideSelColor:Bool): Int;
Func ItemHtPlain(const Text:Str):Str;
{ ClearList - clears list of TObject }
Proc ClearList(List: TList);
Proc MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
Proc ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
{ RTTI support }
Func GetPropType(Obj: TObject; const PropName:Str): TTypeKind;
Func GetPropStr(Obj: TObject; const PropName:Str):Str;
Func GetPropOrd(Obj: TObject; const PropName:Str): Int;
Func GetPropMethod(Obj: TObject; const PropName:Str): TMethod;
Proc PrepareIniSection(Ss: TStrings);
{following functions are not documented because they are don't work properly,so don't use them}
// (rom) from JvBandWindows to make it obsolete
Func PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
// (rom) from JvBandUtils to make it obsolete
Func iif(const Test:Bool; const ATrue, AFalse: Variant): Variant;
Proc CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
Func CreateIconFromClipboard: TIcon;
{ begin JvIconClipboardUtils } { Icon clipboard routines }
Func CF_ICON: Word;
Proc AssignClipboardIcon(Icon: TIcon);
{ Real-size icons support routines (32-bit only) }
Proc GetIconSize(Icon: HICON; var W, H: Int);
Func CreateRealSizeIcon(Icon: TIcon): HICON;
Proc DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Int);
{end JvIconClipboardUtils }
Func CreateScreenCompatibleDC: HDC;
Func InvalidateRect(hWnd: HWND; const lpRect:TRect; bErase:BOOL):BOOL; overload;{$IFDEF SUPPORTS_INLINE}
inline; {$ENDIF}
Func InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
{ begin JvRLE } // (rom) changed API for inclusion in JCL
Proc RleCompressTo(InStream, OutStream: TStream);
Proc RleDecompressTo(InStream, OutStream: TStream);
Proc RleCompress(Stream: TStream);
Proc RleDecompress(Stream: TStream);
{ end JvRLE } { begin JvDateUtil }
Func CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
Func IsLeapYear(AYear: Int):Bool;
Func DaysInAMonth(const AYear, AMonth: Word): Word;
Func DaysPerMonth(AYear, AMonth: Int): Int;
Func FirstDayOfPrevMonth: TDateTime;
Func LastDayOfPrevMonth: TDateTime;
Func FirstDayOfNextMonth: TDateTime;
Func ExtractDay(ADate: TDateTime): Word;
Func ExtractMonth(ADate: TDateTime): Word;
Func ExtractYear(ADate: TDateTime): Word;
Func IncDate(ADate: TDateTime; Days, Months, Years: Int): TDateTime;
Func IncDay(ADate: TDateTime;Delta:Int):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
Func IncMonth(ADate: TDateTime; Delta: Int): TDateTime;
Func IncYear(ADate: TDateTime; Delta: Int): TDateTime;
Func ValidDate(ADate: TDateTime):Bool;
Proc DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
Func MonthsBetween(Date1, Date2: TDateTime): Double;
Func DaysInPeriod(Date1, Date2: TDateTime): Longint;
```

```
4901: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4902: Func DaysBetween(Date1, Date2: TDateTime): Longint;
4903: { The same as previous but if Date2 < Date1 result = 0 }
4904: Func IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Int): TDateTime;
4905: Func IncHour(ATime: TDateTime; Delta: Int): TDateTime;
4906: Func IncMinute(ATime: TDateTime; Delta: Int): TDateTime;
4907: Func IncSecond(ATime: TDateTime; Delta: Int): TDateTime;
4908: Func IncMSec(ATime: TDateTime; Delta: Int): TDateTime;
4909: Func CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4910: { String to date conversions }
4911: Func GetDateOrder(const DateFormat:Str): TDateOrder;
4912: Func MonthFromName(const S:Str; MaxLen: Byte): Byte;
4913: Func StrToDateDef(const S:Str; Default: TDateTime): TDateTime;
4914: Func StrToDateFmt(const DateFormat, S:Str): TDateTime;
4915: Func StrToDateFmtDef(const DateFormat, S:Str; Default: TDateTime): TDateTime;
4916: //Func DefDateFormat(AFourDigitYear:Bool):Str;
4917: //Func DefDateMask(BlanksChar: Char; AFourDigitYear:Bool):Str;
4918: Func FormatLongDate(Value: TDateTime):Str;
4919: Func FormatLongDateTime(Value: TDateTime):Str;
4920: { end JvDateUtil }
4921: Func BufToBinStr(Buf: Pointer; BufSize: Int):Str;
4922: Func BinStrToBuf(Value:Str; Buf: Pointer; BufSize: Int): Int;
4923: { begin JvStrUtils } { ** Common string handling routines ** }
4924: {$IFDEF UNIX}
4925: Func iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes:Card;
4926:   const ToCode, FromCode: Ansistr):Bool;
4927: Func iconvString(const S, ToCode, FromCode: Ansistr):Str;
4928: Func iconvWideString(const S: WideString; const ToCode, FromCode:Ansistr):WideString;
4929: Func OemStrToAnsi(const S: Ansistr): Ansistr;
4930: Func AnsiStrToOem(const S: Ansistr): Ansistr;
4931: {$ENDIF UNIX}
4932: Func StrToOem(const AnsiStr: Ansistr): Ansistr;
4933: { StrToOem translates a string from the Windows character set into the OEM character set. }
4934: Func OemToAnsiStr(const OemStr: Ansistr): Ansistr;
4935: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4936: Func IsEmptyStr(const S:Str; const EmptyChars: TSysCharSet):Bool;
4937: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4938: Func ReplaceStr(const S, Srch, Replace:Str):Str;
4939: { Returns string with every occurrence of Srch string replaced with Replace string. }
4940: Func DelSpace(const S:Str):Str;
4941: { DelSpace return a string with all white spaces removed. }
4942: Func DelChars(const S:Str; Chr: Char):Str;
4943: { DelChars return a string with all Chr characters removed. }
4944: Func DelBSpace(const S:Str):Str;
4945: { DelBSpace trims leading spaces from given string. }
4946: Func DelESpace(const S:Str):Str;
4947: { DelESpace trims trailing spaces from  given string. }
4948: Func DelRSpace(const S:Str):Str;
4949: { DelRSpace trims leading and trailing spaces from the given string. }
4950: Func DelSpace1(const S:Str):Str;
4951: { DelSpace1 return a string with all non-single white spaces removed. }
4952: Func Tab2Space(const S:Str; Numb: Byte):Str;
4953: { Tab2Space converts any tabulation character in given string to the
4954:   Numb spaces characters. }
4955: Func NPos(const C:Str; S:Str; N: Int): Int;
4956: { NPos searches for a N-th position of substring C in a given string. }
4957: Func MakeStr(C: Char; N: Int):Str; overload;
4958: {$IFNDEF COMPILER12_UP}
4959: Func MakeStr(C: WideChar; N: Int): WideString; overload;
4960: {$ENDIF !COMPILER12_UP}
4961: Func MS(C: Char; N: Int):Str; {$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4962: { MakeStr return a string of length N filled with character C. }
4963: Func AddChar(C: Char; const S:Str; N: Int):Str;
4964: { AddChar return a string left-padded to length N with characters C. }
4965: Func AddCharR(C: Char; const S:Str; N: Int):Str;
4966: { AddCharR return a string right-padded to length N with characters C. }
4967: Func LeftStr(const S:Str; N: Int):Str;
4968: { LeftStr return a string right-padded to length N with blanks. }
4969: Func RightStr(const S:Str; N: Int):Str;
4970: { RightStr return a string left-padded to length N with blanks. }
4971: Func CenterStr(const S:Str; Len: Int):Str;
4972: { CenterStr centers the characters in the string based upon the Len specified. }
4973: Func CompStr(const S1, S2:Str): Int; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4974: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4975:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4976: Func CompText(const S1, S2:Str):Int; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4977: { CompText compares S1 to S2,without case-sensitivity. return value is same as for CompStr. }
4978: Func Copy2Symb(const S:Str; Symb: Char):Str;
4979: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4980: Func Copy2SymbDel(var S:Str; Symb: Char):Str;
4981: { Copy2SymbDel returns a substring of a string S from begining to first
4982:   character Symb and removes this substring from S. }
4983: Func Copy2Space(const S:Str):Str;
4984: { Copy2Symb returns a substring of a string S from begining to first white space. }
4985: Func Copy2SpaceDel(var S:Str):Str;
4986: { Copy2SpaceDel returns a substring of a string S from begining to first
4987:   white space and removes this substring from S. }
4988: Func AnsiProperCase(const S:Str; const WordDelims: TSysCharSet):Str;
4989: { Returns string, with the first letter of each word in uppercase,
```

```
4990:   all other letters in lowercase. Words are delimited by WordDelims. }
4991: Func WordCount(const S:Str; const WordDelims: TSysCharSet): Int;
4992: { WordCount given a set of word delimiters, returns number of words in S. }
4993: Func WordPosition(const N: Int; const S:Str; const WordDelims: TSysCharSet):Int;
4994: { Given a set of word delimiters, returns start position of N'th word in S. }
4995: Func ExtractWord(N: Int; const S:Str; const WordDelims: TSysCharSet):Str;
4996: Func ExtractWordPos(N:Int;const S:str;const WordDelims:TSysCharSet;var Pos:Int):str;
4997: Func ExtractDelimited(N: Int; const S:Str;const Delims: TSysCharSet):Str;
4998: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4999:   delimiters, return the N'th word in S. }
5000: Func ExtractSubstr(const S:Str; var Pos: Int;const Delims: TSysCharSet):Str;
5001: { ExtractSubstr given a set of word delimiters, returns the substring from S,
5002:   that started from position Pos. }
5003: Func IsWordPresent(const W, S:Str; const WordDelims: TSysCharSet):Bool;
5004: {IsWordPresent given a set of word delimiters, returns True if word W is present in string S.}
5005: Func QuotedString(const S:Str; Quote: Char):Str;
5006: {QuotedString returns the given string as a quoted string, using provided Quote character. }
5007: Func ExtractQuotedString(const S:Str; Quote: Char):Str;
5008: { ExtractQuotedString removes the Quote characters from beginning and
5009:   end of a quoted string,and reduces pairs of Quotecharacters within quoted string to single character.}
5010: Func FindPart(const HelpWilds, InputStr:Str): Int;
5011: { FindPart compares a string with '?' and another, returns position of HelpWilds in InputStr.}
5012: Func IsWild(InputStr, Wilds:Str; IgnoreCase:Bool):Bool;
5013: { IsWild compares InputString with WildCard string and returns True if corresponds. }
5014: Func XorString(const Key, Src: ShortString): ShortString;
5015: Func XorEncode(const Key, Source:Str):Str;
5016: Func XorDecode(const Key, Source:Str):Str;
5017: { ** Command line routines ** }
5018: Func GetCmdLineArg(const Switch:Str; ASwitchChars: TSysCharSet):Str;
5019: { ** Numeric string handling routines ** }
5020: Func Numb2USA(const S:Str):Str;
5021: { Numb2USA converts numeric string S to USA-format. }
5022: Func Dec2Hex(N: Longint; A: Byte):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
5023: { Dec2Hex converts the given value to a hexadecimal string representation
5024:   with the minimum number of digits (A) specified. }
5025: Func Hex2Dec(const S:Str): Longint;
5026: { Hex2Dec converts the given hexadecimal string to the corresponding Int value. }
5027: Func Dec2Numb(N: Int64; A, B: Byte):Str;
5028: { Dec2Numb converts the given value to a string representation with the
5029:   base equal to B and with the minimum number of digits (A) specified. }
5030: Func Numb2Dec(S:Str; B: Byte): Int64;
5031: { Numb2Dec converts the given B-based numeric string to corresponding Int value. }
5032: Func IntToBin(Value: Longint; Digits, Spaces: Int):Str;
5033: { IntToBin converts the given value to a binary string representation
5034:   with the minimum number of digits specified. }
5035: Func IntToRoman(Value: Longint):Str;
5036: { IntToRoman converts the given value to a roman numeric string representation. }
5037: Func RomanToInt(const S:Str): Longint;
5038: { RomanToInt converts the given string to an Int value. If the string
5039:   doesn't contain a valid roman numeric value, the 0 value is returned. }
5040: Func FindNotBlankCharPos(const S:Str): Int;
5041: Func FindNotBlankCharPosW(const S: WideString): Int;
5042: Func AnsiChangeCase(const S:Str):Str;
5043: Func WideChangeCase(const S:Str):Str;
5044: Func StartsText(const SubStr, S:Str):Bool;
5045: Func EndsText(const SubStr, S:Str):Bool;
5046: Func DequotedStr(const S:Str; QuoteChar: Char = '''''):Str;
5047: Func AnsiDequotedStr(const S:Str; AQuote:Char):str;//follow Delphi 2009's "Ansi" prefix
5048: {end JvStrUtils}
5049: {$IFDEF UNIX}
5050: Func GetTempFileName(const Prefix: Ansistr):Ansistr;
5051: {$ENDIF UNIX}
5052: { begin JvFileUtil }
5053: Func FileDateTime(const FileName:Str): TDateTime;
5054: Func HasAttr(const FileName:Str; Attr: Int):Bool;
5055: Func DeleteFilesEx(const FileMasks: array of string):Bool;
5056: Func NormalDir(const DirName:Str):Str;
5057: Func RemoveBackSlash(const DirName:Str):Str; //only for Windows/DOS Paths
5058: Func ValidFileName(const FileName:Str):Bool;
5059: {$IFDEF MSWINDOWS}
5060: Func FileLock(Handle: Int; Offset, LockSize: Longint): Int; overload;
5061: Func FileLock(Handle: Int; Offset, LockSize: Int64): Int; overload;
5062: Func FileUnlock(Handle: Int; Offset, LockSize: Longint): Int; overload;
5063: Func FileUnlock(Handle: Int; Offset, LockSize: Int64): Int;overload;
5064: {$ENDIF MSWINDOWS}
5065: Func GetWindowsDir:Str;
5066: Func GetSystemDir:Str;
5067: Func ShortToLongFileName(const ShortName:Str):Str;
5068: Func LongToShortFileName(const LongName:Str):Str;
5069: Func ShortToLongPath(const ShortName:Str):Str;
5070: Func LongToShortPath(const LongName:Str):Str;
5071: {$IFDEF MSWINDOWS}
5072: Proc CreateFileLink(const FileName, DisplayName:Str; Folder: Int);
5073: Proc DeleteFileLink(const DisplayName:Str; Folder: Int);
5074: {$ENDIF MSWINDOWS}
5075: { end JvFileUtil }
5076: // Works like PtInRect but includes all edges in comparision
5077: Func PtInRectInclusive(R: TRect; Pt: TPoint):Bool;
5078: // Works like PtInRect but excludes all edges from comparision
```

```
5079: Func PtInRectExclusive(R: TRect; Pt: TPoint):Bool;
5080: Func FourDigitYear:Bool; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5081: Func IsFourDigitYear:Bool;
5082: { moved from JvJVCLUTils }
5083: //Open an object with the shell (url or something like that)
5084: Func OpenObject(const Value:Str):Bool; overload;
5085: Func OpenObject(Value: PChar):Bool; overload;
5086: {$IFDEF MSWINDOWS}
5087: //Raise the last Exception
5088: Proc RaiseLastWin32; overload;
5089: Proc RaiseLastWin32(const Text:Str); overload;
5090: //Raise last Exception with a small comment from your part {GetFileVersion returns most significant 32
      bits of a file's binary version number. Typically, this includes major/minor version placed together in
      one 32-bit Int.
5091: Func GetFileVersion(const AFileName:Str):Card;
5092: {$EXTERNALSYM GetFileVersion}
5093: //Get version of Shell.dll
5094: Func GetShellVersion:Card;
5095: {$EXTERNALSYM GetShellVersion}
5096: // CD functions  on HW
5097: Proc OpenCdDrive;
5098: Proc CloseCdDrive;
5099: // returns True if Drive is accessible
5100: Func DiskInDrive(Drive: Char):Bool;
5101: {$ENDIF MSWINDOWS}
5102: //Same as linux Func ;)
5103: Proc PError(const Text:Str);
5104: // execute a program without waiting
5105: Proc Exec(const FileName, Parameters, Directory:Str);
5106: // execute a program and wait for it to finish
5107: Func ExecuteAndWait(CmdLine:str;const WorkingDirectory:str;Visibility:Int=SW_SHOW):Int;
5108: // returns True if this is the first instance of the program that is running
5109: Func FirstInstance(const ATitle:Str):Bool;
5110: // restores a window based on it's classname and Caption. Either can be left empty
5111: // to widen the search
5112: Proc RestoreOtherInstance(const MainFormClassName, MainFormCaption:Str);
5113: // manipulate the traybar and start button
5114: Proc HideTraybar;
5115: Proc ShowTraybar;
5116: Proc ShowStartButton(Visible:Bool = True);
5117: //(rom) SC_MONITORPOWER documented as Win 95 only(rom) do some testingset monitor functions
5118: Proc MonitorOn;
5119: Proc MonitorOff;
5120: Proc LowPower;
5121: // send a key to the window named AppName
5122: Func SendKey(const AppName:Str; Key: Char):Bool;
5123: {$IFDEF MSWINDOWS}
5124: // returns list of all win currently visible, Objects property is filled with window handle
5125: Proc GetVisibleWindows(List: TStrings);
5126: Func GetVisibleWindowsF( List : TStrings):TStrings;
5127: // associates an extension to a specific program
5128: Proc AssociateExtension(const IconPath, ProgramName, Path, Extension:Str);
5129: Proc AddToRecentDocs(const FileName:Str);
5130: Func GetRecentDocs: TStringList;
5131: {$ENDIF MSWINDOWS}
5132: Func CharIsMoney(const Ch: Char):Bool;
5133: //Func StrToCurrDef(const Str:Str; Def: Currency): Currency;
5134: Func IntToExtended(I: Int): Extended;
5135: { GetChangedText works out new text given the current cursor pos & the key pressed
5136:  is not very useful in other contexts,but in thisunit as itis needed in both MemoEx and TypedEdit}
5137: Func GetChangedText(const Text:Str; SelStart, SelLength: Int; Key: Char):Str;
5138: Func MakeYear4Digit(Year, Pivot: Int): Int;
5139: //Func StrIsInt(const S:Str):Bool;
5140: Func StrIsFloatMoney(const Ps:Str):Bool;
5141: Func StrIsDateTime(const Ps:Str):Bool;
5142: Func PreformatDateString(Ps:Str):Str;
5143: Func BooleanToInt(const B:Bool): Int;
5144: Func StringToBoolean(const Ps:Str):Bool;
5145: Func SafeStrToDateTime(const Ps:Str): TDateTime;
5146: Func SafeStrToDate(const Ps:Str): TDateTime;
5147: Func SafeStrToTime(const Ps:Str): TDateTime;
5148: Func StrDelete(const psSub, psMain:Str):Str;
5149:   { returns the fractional value of pcValue}
5150: Func TimeOnly(pcValue: TDateTime): TTime;
5151: { returns the integral value of pcValue }
5152: Func DateOnly(pcValue: TDateTime): TDate;
5153: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5154: const { TDateTime value used to signify Null value}
5155:   NullEquivalentDate: TDateTime = 0.0;
5156: Func DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind):Bool;
5157: // Replacement for Win32Check to avoid platform specific warnings in D6
5158: Func OSCheck(RetVal:Bool):Bool;
5159: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5160:   Same as FileCtrl.MinimizeName in functionality (but not implementation).Included here to
5161:   not be forced to use FileCtrl unnecessarily }
5162: Func MinimizeFileName(const FileName:Str; Canvas: TCanvas; MaxLen: Int):Str;
5163: Func MinimizeText(const Text:Str; Canvas: TCanvas; MaxWidth: Int):Str;
5164: {MinimizeString trunactes longstring,S,and appends'...'symbols,if Length ofS is morethan MaxLen}
5165: Func MinimizeString(const S:Str; const MaxLen: Int):Str;
```

```
5166: Proc RunDll32Internal(Wnd:THandle;const DLLName,FuncName,CmdLine:str;CmdShow:Int= SW_SHOWDEFAULT);
5167: { 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.}
5168: Func GetDLLVersion(const DLLName:Str; var pdwMajor, pdwMinor: Int):Boolean;
5169: {$ENDIF MSWINDOWS}
5170: Proc ResourceNotFound(ResID: PChar);
5171: Func EmptyRect: TRect;
5172: Func RectWidth(R: TRect): Int;
5173: Func RectHeight(R: TRect): Int;
5174: Func CompareRect(const R1, R2: TRect):Bool;
5175: Proc RectNormalize(var R: TRect);
5176: Func RectIsSquare(const R: TRect):Bool;
5177: Func RectSquare(var ARect: TRect; AMaxSize: Int = -1):Bool;
5178: //If AMaxSize = -1 ,then auto calc Square's max size
5179: {$IFDEF MSWINDOWS}
5180: Proc FreeUnusedOle;
5181: Func GetWindowsVersion:Str;
5182: Func LoadDLL(const LibName:Str): THandle;
5183: Func RegisterServer(const ModuleName:Str): Bool;
5184: Func UnregisterServer(const ModuleName:Str): Bool;
5185: {$ENDIF MSWINDOWS}
5186: { String routines }
5187: Func GetEnvVar(const VarName:Str):Str;
5188: Func AnsiUpperFirstChar(const S:str):str;//follow Delphi2009's ex. with "Ansi" prefix
5189: Func StringToPChar(var S:Str): PChar;
5190: Func StrPAlloc(const S:Str): PChar;
5191: Proc SplitCommandLine(const CmdLine:Str; var ExeName, Params:Str);
5192: Func DropT(const S:Str):Str;
5193: { Memory routines }
5194: Func AllocMemo(Size: Longint): Pointer;
5195: Func ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5196: Proc FreeMemo(var fpBlock: Pointer);
5197: Func GetMemoSize(fpBlock: Pointer): Longint;
5198: Func CompareMem(fpBlock1, fpBlock2: Pointer; Size:Card):Bool;
5199: { Manipulate huge pointers routines }
5200: Proc HugeInc(var HugePtr: Pointer; Amount: Longint);
5201: Proc HugeDec(var HugePtr: Pointer; Amount: Longint);
5202: Func HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5203: Proc HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5204: Proc HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5205: Func WindowClassName(Wnd: THandle):Str;
5206: Proc SwitchToWindow(Wnd: THandle; Restore:Bool);
5207: Proc ActivateWindow(Wnd: THandle);
5208: Proc ShowWinNoAnimate(Handle: THandle; CmdShow: Int);
5209: Proc KillMessage(Wnd: THandle; Msg:Card);
5210: { SetWindowTop put window to top without recreating window }
5211: Proc SetWindowTop(const Handle: THandle; const Top:Bool);
5212: Proc CenterWindow(Wnd: THandle);
5213: Func MakeVariant(const Values: array of Variant): Variant;
5214: { Convert dialog units to pixels and backwards }
5215: {$IFDEF MSWINDOWS}
5216: Func DialogUnitsToPixelsX(DlgUnits:Word):Word;
5217: Func DialogUnitsToPixelsY(DlgUnits:Word):Word;
5218: Func PixelsToDialogUnitsX(PixUnits:Word):Word;
5219: Func PixelsToDialogUnitsY(PixUnits:Word):Word;
5220: {$ENDIF MSWINDOWS}
5221: Func GetUniqueFileNameInDir(const Path, FileNameMask:Str):Str;
5222: {$IFDEF BCB}
5223: Func FindPrevInstance(const MainFormClass: ShortString;const ATitle:Str): THandle;
5224: Func ActivatePrevInstance(const MainFormClass: ShortString;const ATitle:str):Bool;
5225: {$ELSE}
5226: Func FindPrevInstance(const MainFormClass, ATitle:Str): THandle;
5227: Func ActivatePrevInstance(const MainFormClass, ATitle:Str):Bool;
5228: {$ENDIF BCB}
5229: {$IFDEF MSWINDOWS}
5230: { BrowseForFolderNative displays Browse For Folder dialog }
5231: Func BrowseForFolderNative(const Handle:THandle;const Title:str;var Folder:str):Bool;
5232: {$ENDIF MSWINDOWS}
5233: Proc AntiAlias(Clip: TBitmap);
5234: Proc AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Int);
5235: Proc CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5236:   ABitmap: TBitmap; const SourceRect: TRect);
5237: Func IsTrueType(const FontName:Str):Bool;
5238: // Removes all non-numeric characters from AValue and returns the resulting string
5239: Func TextToValText(const AValue:Str):Str;
5240: Func ExecRegExpr( const ARegExpr, AInputStr : RegExprString) :Bool
5241: Proc SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5242: Func ReplaceRegExpr(const ARegExpr,AInputStr,
      AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5243: Func QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5244: Func RegExprSubExpressions(const ARegExpr:str;ASubExprs:TStrings;AExtendedSyntax:bool):
5245:
5246: ****************************************************unit uPSI_JvTFUtils;
5247:  Func JExtractYear( ADate : TDateTime) : Word
5248:  Func JExtractMonth( ADate : TDateTime) : Word
5249:  Func JExtractDay( ADate : TDateTime) : Word
5250:  Func ExtractHours( ATime : TDateTime) : Word
5251:  Func ExtractMins( ATime : TDateTime) : Word
5252:  Func ExtractSecs( ATime : TDateTime) : Word
```

```
5253:  Func ExtractMSecs( ATime : TDateTime) : Word
5254:  Func FirstOfMonth( ADate : TDateTime) : TDateTime
5255:  Func GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5256:  Func GetWeeksInMonth( Year, Month : Word; StartOfWeek : Int) : Word
5257:  Proc IncBorlDOW( var BorlDOW : Int; N : Int)
5258:  Proc IncDOW( var DOW : TTFDayOfWeek; N : Int)
5259:  Proc IncDays( var ADate : TDateTime; N : Int)
5260:  Proc IncWeeks( var ADate : TDateTime; N : Int)
5261:  Proc IncMonths( var ADate : TDateTime; N : Int)
5262:  Proc IncYears( var ADate : TDateTime; N : Int)
5263:  Func EndOfMonth( ADate : TDateTime) : TDateTime
5264:  Func IsFirstOfMonth( ADate : TDateTime) :Bool
5265:  Func IsEndOfMonth( ADate : TDateTime) :Bool
5266:  Proc EnsureMonth( Month : Word)
5267:  Proc EnsureDOW( DOW : Word)
5268:  Func EqualDates( D1, D2 : TDateTime) :Bool
5269:  Func Lesser( N1, N2 : Int) : Int
5270:  Func Greater( N1, N2 : Int) : Int
5271:  Func GetDivLength( TotalLength, DivCount, DivNum : Int) : Int
5272:  Func GetDivNum( TotalLength, DivCount, X : Int) : Int
5273:  Func GetDivStart( TotalLength, DivCount, DivNum : Int) : Int
5274:  Func DOWToBorl( ADOW : TTFDayOfWeek) : Int
5275:  Func BorlToDOW( BorlDOW : Int) : TTFDayOfWeek
5276:  Func DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5277:  Proc CalcTextPos(HostRect:TRect;var TextLeft,TextTop:Int;var
      TextBounds:TRect;AFont:TFont;AAngle:Int;HAlign:TAlignment;VAlign:TJvTFVAlignment;ATxt:str)
5278:  Proc DrawAngleText(ACanvas:TCanvas;HostRect:TRect;var TextBounds:TRect;AAngle:Int;HAlign:
      TAlignment;VAlign:TJvTFVAlignment;ATxt:str)
5279:  Func JRectWidth( ARect : TRect) : Int
5280:  Func JRectHeight( ARect : TRect): Int
5281:  Func JEmptyRect : TRect
5282:  Func IsClassByName( Obj : TObject; ClassName:str):Boolean
5283:
5284: Proc SIRegister_MSysUtils(CL: TPSPascalCompiler);
5285: begin
5286:  Proc HideTaskBarButton( hWindow : HWND)
5287:  Func msLoadStr( ID : Int) :Str
5288:  Func msFormat( fmt :Str; params : array of const) :Str
5289:  Func msFileExists( const FileName :Str) :Bool
5290:  Func msIntToStr( Int : Int64) :Str
5291:  Func msStrPas( const Str : PChar) :Str
5292:  Func msRenameFile( const OldName, NewName :Str) :Bool
5293:  Func CutFileName( s :Str) :Str
5294:  Func GetVersionInfo( var VersionString :Str) : DWORD
5295:  Func FormatTime( t :Card) :Str
5296:  Func msCreateDir( const Dir :Str) :Bool
5297:  Func SetAutoRun( NeedAutoRun :Bool; AppName :Str) :Bool
5298:  Func SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5299:  Func msStrLen( Str : PChar) : Int
5300:  Func msDirectoryExists( const Directory :Str) :Bool
5301:  Func GetFolder( hWnd : hWnd; RootDir : Int; Caption :Str) :Str
5302:  Func SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5303:  Func EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT
5304:  Proc SetEditWndProc( hWnd : HWND; ptr : TObject)
5305:  Func GetTextFromFile( Filename :Str) :Str
5306:  Func IsTopMost(hWnd:HWND):Bool //'LWA_ALPHA','LongWord').SetUInt( $00000002);
5307:  Func msStrToIntDef( const s :Str; const i : Int) : Int
5308:  Func msStrToInt( s :Str) : Int
5309:  Func GetItemText( hDlg : THandle; ID : DWORD):str
5310: end;
5311:
5312: Proc SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5313: begin
5314:   //TDynFloatArray', 'array of Extended
5315:   TDynLWordArray', 'array of LongWord
5316:   TDynLIntArray', 'array of LongInt
5317:   TDynFloatMatrix', 'array of TDynFloatArray
5318:   TDynLWordMatrix', 'array of TDynLWordArray
5319:   TDynLIntMatrix', 'array of TDynLIntArray
5320:  Func SquareAll( const X : TDynFloatArray) : TDynFloatArray
5321:  Func InverseAll( const X : TDynFloatArray) : TDynFloatArray
5322:  Func LnAll( const X : TDynFloatArray) : TDynFloatArray
5323:  Func Log10All( const X : TDynFloatArray) : TDynFloatArray
5324:  Func LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5325:  Func AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5326:  Func SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5327:  Func MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5328:  Func DotProduct( const X, Y : TDynFloatArray) : Extended
5329:  Func MNorm( const X : TDynFloatArray) : Extended
5330:  Func MatrixIsRectangular( const X : TDynFloatMatrix) :Bool
5331:  Proc MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Bool;
5332:  Func MatrixIsSquare( const X : TDynFloatMatrix) :Bool
5333:  Func MatricesSameDimensions( const X, Y : TDynFloatMatrix) :Bool
5334:  Func AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5335:  Proc AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5336:  Func SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5337:  Proc SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5338:  Func MultiplyMatrixByConst( const X: TDynFloatMatrix; const K: Extended):TDynFloatMatrix
5339:  Proc MultiplyMatrixByConst2( var X: TDynFloatMatrix; const K: Extended);
```

```
5340:  Func MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5341:  Func TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5342:  Func GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5343: end;
5344:
5345: Proc SIRegister_ESBMaths(CL: TPSPascalCompiler);
5346: begin
5347:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5348:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5349:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5350:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5351:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5352:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5353:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5354:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5355:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5356:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5357:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5358:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5359:  'ESBSqrtPi','Extended').setExtended( 1.7724538509055160272 9);
5360:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5361:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5362:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5363:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5364:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5365:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5366:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5367:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5368:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5369:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5370:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5371:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5372:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5373:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5374:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5375:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5376:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5377:  'ESBLnPi','Extended').setExtended( 1.1447298858494001741 4);
5378:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5379:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5380:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5381:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5382:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5383:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5384:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5385:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5386:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5387:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5388:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5389:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5390:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5391:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5392:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5393:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5394:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5395:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5396:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5397:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5398:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5399:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5400:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5401:   //LongWord', 'Cardinal
5402:   TBitList', 'Word
5403:  Func UMul( const Num1, Num2 : LongWord) : LongWord
5404:  Func UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5405:  Func UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5406:  Func UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5407:  Func SameFloat( const X1, X2 : Extended) :Bool
5408:  Func FloatIsZero( const X : Extended) :Bool
5409:  Func FloatIsPositive( const X : Extended) :Bool
5410:  Func FloatIsNegative( const X : Extended) :Bool
5411:  Proc IncLim( var B : Byte; const Limit : Byte)
5412:  Proc IncLimSI( var B : ShortInt; const Limit : ShortInt)
5413:  Proc IncLimW( var B : Word; const Limit : Word)
5414:  Proc IncLimI( var B : Int; const Limit : Int)
5415:  Proc IncLimL( var B : LongInt; const Limit : LongInt)
5416:  Proc DecLim( var B : Byte; const Limit : Byte)
5417:  Proc DecLimSI( var B : ShortInt; const Limit : ShortInt)
5418:  Proc DecLimW( var B : Word; const Limit : Word)
5419:  Proc DecLimI( var B : Int; const Limit : Int)
5420:  Proc DecLimL( var B : LongInt; const Limit : LongInt)
5421:  Func MaxB( const B1, B2 : Byte) : Byte
5422:  Func MinB( const B1, B2 : Byte) : Byte
5423:  Func MaxSI(const B1, B2 : ShortInt) : ShortInt
5424:  Func MinSI(const B1, B2 : ShortInt) : ShortInt
5425:  Func MaxW(const B1, B2 : Word) : Word
5426:  Func MinW(const B1, B2 : Word) : Word
5427:  Func esbMaxI( const B1, B2 : Int) : Int
5428:  Func esbMinI( const B1, B2 : Int) : Int
```

```
Func MaxL(const B1, B2 : LongInt) : LongInt
Func MinL(const B1, B2 : LongInt) : LongInt
Proc SwapB( var B1, B2 : Byte)
Proc SwapSI( var B1, B2 : ShortInt)
Proc SwapW( var B1, B2 : Word)
Proc SwapI( var B1, B2 : SmallInt)
Proc SwapL( var B1, B2 : LongInt)
Proc SwapI32( var B1, B2 : Int)
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
```

```
5518:  Proc ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5519:  Func ESBArcCos( const X : Extended) : Extended
5520:  Func ESBArcSin( const X : Extended) : Extended
5521:  Func ESBArcSec( const X : Extended) : Extended
5522:  Func ESBArcCosec( const X : Extended) : Extended
5523:  Func ESBLog10( const X : Extended) : Extended
5524:  Func ESBLog2( const X : Extended) : Extended
5525:  Func ESBLogBase( const X, Base : Extended) : Extended
5526:  Func Pow2( const X : Extended) : Extended
5527:  Func IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5528:  Func ESBIntPower( const X : Extended; const N : LongInt) : Extended
5529:  Func XtoY( const X, Y : Extended) : Extended
5530:  Func esbTenToY( const Y : Extended) : Extended
5531:  Func esbTwoToY( const Y : Extended) : Extended
5532:  Func LogXtoBaseY( const X, Y : Extended) : Extended
5533:  Func esbISqrt( const I : LongWord) : Longword
5534:  Func ILog2( const I : LongWord) : LongWord
5535:  Func IGreatestPowerOf2( const N : LongWord) : LongWord
5536:  Func ESBArCosh(X:Extended):Extended
5537:  Func ESBArSinh(X:Extended):Extended
5538:  Func ESBArTanh(X:Extended):Extended
5539:  Func ESBCosh(X:Extended):Extended
5540:  Func ESBSinh(X:Extended):Extended
5541:  Func ESBTanh(X:Extended):Extended
5542:  Func InverseGamma(const X: Extended) : Extended
5543:  Func esbGamma(const X : Extended): Extended
5544:  Func esbLnGamma(const X: Extended): Extended
5545:  Func esbBeta(const X, Y: Extended): Extended
5546:  Func IncompleteBeta(X :Extended; P, Q: Extended): Extended
5547: end;
5548:
5549: ****************************Int Huge Cardinal Utils**************************
5550: Func Add_uint64_WithCarry( x, y : uint64; var Carry :Bool) : uint64
5551: Func Add_uint32_WithCarry( x, y : uint32; var Carry :Bool) : uint32
5552: Func Subtract_uint64_WithBorrow( x, y : uint64; var Borrow :Bool) : uint64
5553: Func Subtract_uint32_WithBorrow( x, y : uint32; var Borrow :Bool) : uint32
5554: Func BitCount_8( Value : byte) : Int
5555: Func BitCount_16( Value : uint16) : Int
5556: Func BitCount_32( Value : uint32) : Int
5557: Func BitCount_64( Value : uint64) : Int
5558: Func CountSetBits_64( Value : uint64) : Int TPrimalityTestNoticeProc',
5559: Proc ( CountPrimalityTests : Int)
5560: Func gcdhuge( a, b : THugeCardinal) : THugeCardinal
5561: Func lcmhuge( a, b : THugeCardinal) : THugeCardinal
5562: Func isCoPrime( a, b : THugeCardinal) :Bool
5563: Func isProbablyPrime(p:THugeCardinal;OnProgress:TProgress;var wasAborted:bool):boolean
5564: Func hasSmallFactor( p : THugeCardinal) :Bool
5565: Func GeneratePrime(NumBits:
      Int;OnProgress:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;PassCount:Int;Pool1:TMemoryStreamPool;var
      Prime:THugeCardinal;var NumbersTested:Int):bool
5566:  Func Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5567:  Const('StandardExponent','LongInt'( 65537);
5568:  Proc Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:Int;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPool;var
      Numbers
5569:  Func Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal:Bool')
5570:  Func AnsiBytesOf(const S:Str): TBytes;
5571:  Func GetBytes(value:Str): TBytes;
5572:  Func GetBytesInt(value: Smallint): TBytes;
5573:  Func HugeToBase10(const AHuge: THugeCardinal):Str;
5574:
5575: (*----------------------------------------------------------------------------*)
5576: Proc SIRegister_uTPLb_StreamUtils(CL: TPSPascalCompiler);
5577: begin
5578:  SIRegister_TDesalinationWriteStream(CL);
5579:  // CL.AddTypeS('TByteDynArray', 'array of Byte
5580:  Func CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
5581:  Proc CopyMemoryStream( Source, Destination : TMemoryStream)
5582:  Proc BurnMemoryStream( Destructo : TMemoryStream)
5583:  Proc BurnMemory( var Buff, BuffLen : integer)
5584:  Proc BurnMemoryString( var Buff:Str; BuffLen : integer)
5585:  Proc BurnMemoryInteger( var Buff: Integer; BuffLen : integer)
5586:  Proc BurnMemoryString2( Buff:Str; BuffLen : integer)
5587:  Proc BurnMemoryInteger2( Buff: Integer; BuffLen : integer)
5588:  (Proc BurnMemoryByteArray( var array of Byte; BuffLen : integer)
5589:  Proc BurnMemoryByteArray2( var buff: TByteDynArray; BuffLen : integer)
5590:  Proc BurnMemoryByteArray3( buff: TByteDynArray; BuffLen : integer)
5591:  Proc ZeroFillStream( Stream : TMemoryStream)
5592:  Proc RandomFillStream( Stream : TMemoryStream)
5593:  Proc XOR_Streams2( Dest, Srce : TMemoryStream)
5594:  Proc XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
5595:  Func CompareMemoryStreams( S1, S2 : TMemoryStream) :Bool
5596:  Func Stream_to_Base64( Source : TStream) : Ansistr
5597:  Func Stream_to_Base642( Source : TStream) : Ansistr
5598:  Proc Base64_to_stream2( const Base64 : Ansistr; Destin : TStream)
5599:  Func Stream_to_Ansistr( Source : TStream) : Ansistr
5600:  Func StreamtoAnsistr( Source : TStream) : Ansistr
5601:  Func StreamToString( Source : TStream) :Str
5602:  Func StreamToString2( Source : TStream) :Str
```

```
5603:  Func StreamToString3( Source : TStream) :Str
5604:  Func StreamToString4( Source : TStream) :Str
5605:   Ansistr_to_stream( const Value : Ansistr; Destin : TStream)
5606:  Func CompareFiles(const FN1,FN2:str;Breathe:TNotifyEvent;BreathingSender:TObject):bool;
5607:  Func FileSize( const FileName :Str) : int64
5608:  Func Stream_to_decimalbytes( Source : TStream) :Str
5609:  Func StreamToDecimalbytes( Source : TStream) :Str
5610:  Func StreamtoOrd( Source : TStream) :Str
5611:  Func StreamToByte( Source : TStream) :Str
5612:  Func DisplayStream( Stream : TStream) :Str
5613:  Proc Base64_to_streamBytes( const Base64: TBytes; Destin: TStream);
5614:  Func Stream_to_Base64Bytes(ASource: TStream; const ATransform: TBytes): TBytes;
5615:  Func Stream_To_Hex(ASource: TStream): Ansistr;
5616:  // from unit uPSI_uTPLb_PointerArithmetic;
5617:  Proc ClearMemory( Stream : TMemoryStream; Offset, CountBytes : integer)
5618:  Func ReadMem(Source:TStream;Destin:TMemoryStream;DestinOffset,CountBytes:integer):int);
5619:  Func WriteMem(Source:TMemoryStream;SourceOffset:int;Destin:TStream;CountBytes:int):int;
5620: end;
5621:
5622: Proc SIRegister_EwbCoreTools(CL: TPSPascalCompiler);
5623: begin
5624:  //CL.AddDelphiFunction('Func IsWinXPSP2OrLater( ) :Bool
5625:  Func ColorToHTML2( const Color : TColor) :Str
5626:  //Func WideStringToLPOLESTR( const Source : Widestring) : POleStr
5627:  //Func XPath4Node( node : IHTMLElement) :Str
5628:  //Func TaskAllocWideString( const S :Str) : PWChar
5629:  Func AnsiIndexStr2( const AText :Str; const AValues : array of string) : Integer
5630:  Func GetPos( const SubSt, Text :Str; StartPos : Integer) : Integer
5631:  Func _CharPos( const C : Char; const S :Str) : Integer
5632:  Func CutString(var Text:str; const Delimiter:str; const Remove:Boolean) :Str
5633:  Proc FormatPath( Path :Str)
5634:  Func GetWinText( WinHandle : THandle) :Str
5635:  Func GetWinClass( Handle : Hwnd) : WideString
5636:  Func GetParentWinByClass( ChildHandle : HWND; const ClassName :Str) : HWND
5637:  Func DirectoryExists2( const Directory :Str) :Bool
5638:  //CL.AddDelphiFunction('Func VarSupports(const V:Variant;const IID:TGUID; out Intf): Bool);
5639:  Func CharInSet3( C : Char; const CharSet : TSysCharSet) :Bool
5640:  Func AddBackSlash2( const S :Str) :Str
5641:  CL.AddConstantN('WM_SETWBFOCUS','LongWord').SetUInt( $0400 + $44);
5642: end;
5643:
5644: Proc SIRegister_EwbUrl(CL: TPSPascalCompiler);
5645: begin
5646:  'TEMP_SIZE_EWB','LongInt').SetInt( 1024);
5647:  'MAX_BUFFER_EWB','LongInt').SetInt( 256);
5648:  'WebDelim_EWB','String').SetString( '/');
5649:  'ProtocolDelim_EWB','String').SetString( '://');
5650:  'QueryDelim_EWB','String').SetString( '?');
5651:  'BookmarkDelim_EWB','String').SetString( '#');
5652:  'EqualDelim_EWB','String').SetString( '=');
5653:  'DriveDelim_EWB','String').SetString( ':');
5654:   CL.AddTypeS('TQueryOption_EWB', 'ULONG');
5655:   TOnError_EWB','Proc (Sender:TObject;ErrorCode:integer;ErrMessage:str);
5656:   SIRegister_TUrl(CL);
5657: end;
5658:
5659: Func IS_INTRESOURCE(lpszType: PChar): BOOL;  //as of windows.pas
5660: begin
5661:   Result:= ULONG_PTR(lpszType) shr 16 = 0;
5662: end;
5663:
5664: Func ResourceNameToString(lpszName: PChar):Str;
5665: Func ResourceTypeToString(lpszType: PChar):Str;
5666: Proc storeRCDATAResourcetofile(aresname, afilename:Str);
5667: Func InternalfindNextUrlToDownload(aNode: TPageNotYetDownloadedBinTreeNode; '+
5668:         alowDeepLevel: Integer): TPageNotYetDownloadedBinTreeNode;'); //ALWebSpider
5669:
5670: Proc SIRegister_xrtl_math_Int(CL: TPSPascalCompiler);
5671: begin
5672:   AddTypeS('TXRTLInt', 'array of Int
5673:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5674:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5675:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5676:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5677:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5678:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5679:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5680:   'BitsPerByte','LongInt'( 8);
5681:   BitsPerDigit','LongInt'( 32);
5682:   SignBitMask','LongWord( $80000000);
5683:  Func XRTLAdjustBits( const ABits : Int) : Int
5684:  Func XRTLLength( const AInt : TXRTLInt) : Int
5685:  Func XRTLDataBits( const AInt : TXRTLInt) : Int
5686:  Proc XRTLBitPosition( const BitIndex : Int; var Index, Mask : Int)
5687:  Proc XRTLBitSet( var AInt : TXRTLInt; const BitIndex : Int)
5688:  Proc XRTLBitReset( var AInt : TXRTLInt; const BitIndex : Int)
5689:  Func XRTLBitGet( const AInt : TXRTLInt; const BitIndex : Int) : Int
5690:  Func XRTLBitGetBool( const AInt : TXRTLInt; const BitIndex : Int) :Bool
5691:  Func XRTLExtend(const AInt:TXRTLInt;ADataBits:Int;Sign:Int;var AResult:TXRTLInt):Int;
```

```
 Func XRTLZeroExtend(const AInt:TXRTLInt;ADataBits:Int; var AResult:TXRTLInt):Int;
 Func XRTLSignExtend(const AInt:TXRTLInt; ADataBits:Int;var AResult:TXRTLInt):Int;
 Func XRTLSignStrip(const AInt:TXRTLInt;var AResult:TXRTLInt;const AMinDataBits:Int):Int;
 Proc XRTLNot( const AInt : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLOr( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLAnd( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLXor( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
 Func XRTLSign( const AInt : TXRTLInt) : Int
 Proc XRTLZero( var AInt : TXRTLInt)
 Proc XRTLOne( var AInt : TXRTLInt)
 Proc XRTLMOne( var AInt : TXRTLInt)
 Proc XRTLTwo( var AInt : TXRTLInt)
 Func XRTLNeg( const AInt : TXRTLInt; var AResult : TXRTLInt) : Int
 Func XRTLAbs( const AInt : TXRTLInt; var AResult : TXRTLInt) : Int
 Proc XRTLFullSum( const A, B, C : Int; var Sum, Carry : Int)
 Func XRTLAdd( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int;
 Func XRTLAdd1(const AInt1:TXRTLInt;const AInt2:Int64;var AResult:TXRTLInt):Int;
 Func XRTLSub( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int;
 Func XRTLSub1(const AInt1:TXRTLInt;const AInt2:Int64;var AResult:TXRTLInt):Int;
 Func XRTLCompare( const AInt1, AInt2 : TXRTLInt) : Int;
 Func XRTLCompare1( const AInt1 : TXRTLInt; const AInt2 : Int64) : Int;
 Func XRTLUMul( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int
 Func XRTLMulAdd(const AInt1,AInt2,AInt3:TXRTLInt; var AResult:TXRTLInt):Int
 Func XRTLMul( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int
 Proc XRTLDivMod( const AInt1, AInt2 : TXRTLInt; var QResult, RResult : TXRTLInt)
 Proc XRTLSqr( const AInt : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLSqrt( const AInt : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLRoot( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLRootApprox(const AInt1,AInt2:TXRTLInt;var ALowApproxResult,AHighApproxResult:TXRTLInt)
 Proc XRTLURootApprox(const AInt1,AInt2:TXRTLInt;var ALowApproxResult,AHighApproxResult:TXRTLInt);
 Proc XRTLExp( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLExpMod( const AInt1, AInt2, AInt3 : TXRTLInt;var AResult: TXRTLInt)
 Proc XRTLSLBL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
 Proc XRTLSABL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
 Proc XRTLRCBL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
 Proc XRTLSLDL(const AInt:TXRTLInt;const DigitCount:Int; var AResult:TXRTLInt)
 Proc XRTLSADL(const AInt: TXRTLInt; const DigitCount:Int;var AResult: TXRTLInt)
 Proc XRTLRCDL(const AInt:TXRTLInt; const DigitCount:Int; var AResult: TXRTLInt)
 Proc XRTLSLBR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
 Proc XRTLSABR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
 Proc XRTLRCBR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
 Proc XRTLSLDR(const AInt:TXRTLInt; const DigitCount:Int; var AResult: TXRTLInt)
 Proc XRTLSADR(const AInt: TXRTLInt; const DigitCount:Int;var AResult: TXRTLInt)
 Proc XRTLRCDR(const AInt: TXRTLInt;const DigitCount:Int;var AResult: TXRTLInt)
 Func XRTLToHex( const AInt : TXRTLInt; Digits : Int) :Str
 Func XRTLToBin( const AInt : TXRTLInt; Digits : Int) :Str
 Func XRTLToString(const AInt : TXRTLInt; Radix : Int; Digits : Int) :Str
 Proc XRTLFromHex(const Value :Str; var AResult : TXRTLInt)
 Proc XRTLFromBin(const Value :Str; var AResult : TXRTLInt)
 Proc XRTLFromString(const Value :Str; var AResult : TXRTLInt; Radix : Int)
 Proc XRTLAssign( const AInt : TXRTLInt; var AResult : TXRTLInt);
 Proc XRTLAssign1(const Value : Int; var AResult : TXRTLInt);
 Proc XRTLAssign2(const Value : Int64; var AResult : TXRTLInt);
 Proc XRTLAppend( const ALow, AHigh : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLSplit(const AInt: TXRTLInt; var ALow,AHigh: TXRTLInt;LowDigits: Int)
 Func XRTLGetMSBitIndex( const AInt : TXRTLInt) : Int
 Proc XRTLMinMax(const AInt1, AInt2:TXRTLInt;var AMinResult,AMaxResult: TXRTLInt)
 Proc XRTLMin( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt);
 Proc XRTLMin1(const AInt1: TXRTLInt;const AInt2:Int;var AResult : TXRTLInt);
 Proc XRTLMax( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt);
 Proc XRTLMax1(const AInt1:TXRTLInt; const AInt2:Int; var AResult:TXRTLInt);
 Proc XRTLGCD( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLSwap( var AInt1, AInt2 : TXRTLInt)
 Proc XRTLFactorial( const AInt : TXRTLInt; var AResult : TXRTLInt)
 Proc XRTLFactorialMod( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
end;

Proc SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
begin
 Func JvXPMethodsEqual( const Method1, Method2 : TMethod) :Bool
 Proc JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Int)
 Proc JvXPCreateGradientRect( const AWidth, AHeight : Int; const StartColor, EndColor : TColor;const
Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Bool;var Bitmap:TBitmap);
 Proc JvXPAdjustBoundRect(const BorderWidth:Byte;const ShowBoundLines:Boolean; const
BoundLines:TJvXPBoundLines; var Rect: TRect)
 Proc JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const AColor:TColor;Rect:TRect;
 Proc JvXPConvertToGray2( Bitmap : TBitmap)
 Proc JvXPRenderText(const AParent:TControl;const ACanvas:TCanvas;ACaption:TCaption;const
AFont:TFont;const AEnabled,AShowAccelChar:Bool;var ARect:TRect;AFlags:Int)
 Proc JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const Swapped:Bool;
 Proc JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
 Proc JvXPSetDrawFlags(const AAlignment:TAlignment;const AWordWrap:Bool;var Flags: Int)
 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)
end;

Proc SIRegister_uwinstr(CL: TPSPascalCompiler);
begin
```

```
5776:  Func StrDec(S :Str) :Str
5777:  Func uIsNumeric( var S :Str; var X : Float) :Bool
5778:  Func ReadNumFromEdit( Edit : TEdit) : Float
5779:  Proc WriteNumToFile( var F : Text; X : Float)
5780: end;
5781:
5782: Proc SIRegister_utexplot(CL: TPSPascalCompiler);
5783: begin
5784:  Func TeX_InitGraphics(FileName:str;PgWidth,PgHeight:Int; Header:Boolean):Boolean
5785:  Proc TeX_SetWindow( X1, X2, Y1, Y2 : Int; GraphBorder :Bool)
5786:  Proc TeX_LeaveGraphics( Footer :Bool)
5787:  Proc TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5788:  Proc TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5789:  Proc TeX_SetGraphTitle( Title :Str)
5790:  Proc TeX_SetOxTitle( Title :Str)
5791:  Proc TeX_SetOyTitle( Title :Str)
5792:  Proc TeX_PlotOxAxis
5793:  Proc TeX_PlotOyAxis
5794:  Proc TeX_PlotGrid( Grid : TGrid)
5795:  Proc TeX_WriteGraphTitle
5796:  Func TeX_SetMaxCurv( NCurv : Byte) :Bool
5797:  Proc TeX_SetPointParam( CurvIndex, Symbol, Size : Int)
5798:  Proc TeX_SetLineParam( CurvIndex, Style : Int; Width : Float; Smooth :Bool)
5799:  Proc TeX_SetCurvLegend( CurvIndex : Int; Legend :Str)
5800:  Proc TeX_SetCurvStep( CurvIndex, Step : Int)
5801:  Proc TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Int)
5802:  Proc TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Int)
5803:  Proc TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Int; CurvIndex : Int)
5804:  Proc TeX_WriteLegend( NCurv : Int; ShowPoints, ShowLines :Bool)
5805:  Proc TeX_ConRec( Nx, Ny, Nc : Int; X, Y, Z : TVector; F : TMatrix)
5806:  Func Xcm(X : Float) : Float
5807:  Func Ycm(Y : Float) : Float
5808: end;
5809:
5810: *----------------------------------------------------------------------------*)
5811: Proc SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5812: begin
5813:   TConstArray', 'array of TVarRec
5814:  Func CopyVarRec( const Item : TVarRec) : TVarRec
5815:  Func CreateConstArray( const Elements : array of const) : TConstArray
5816:  Proc FinalizeVarRec( var Item : TVarRec)
5817:  Proc FinalizeConstArray( var Arr : TConstArray)
5818: end;
5819:
5820: Proc SIRegister_StStrS(CL: TPSPascalCompiler);
5821: begin
5822:  Func HexBS( B : Byte) : ShortString
5823:  Func HexWS( W : Word) : ShortString
5824:  Func HexLS( L : LongInt) : ShortString
5825:  Func HexPtrS( P : Pointer) : ShortString
5826:  Func BinaryBS( B : Byte) : ShortString
5827:  Func BinaryWS( W : Word) : ShortString
5828:  Func BinaryLS( L : LongInt) : ShortString
5829:  Func OctalBS( B : Byte) : ShortString
5830:  Func OctalWS( W : Word) : ShortString
5831:  Func OctalLS( L : LongInt) : ShortString
5832:  Func Str2Int16S( const S : ShortString; var I : SmallInt) :Bool
5833:  Func Str2WordS(const S: ShortString; var I : Word) :Bool
5834:  Func Str2LongS(const S: ShortString; var I : LongInt) :Bool
5835:  Func Str2RealS(const S: ShortString; var R : Double) :Bool
5836:  Func Str2RealS(const S: ShortString; var R : Real) :Bool
5837:  Func Str2ExtS(const S : ShortString; var R : Extended) :Bool
5838:  Func Long2StrS( L : LongInt) : ShortString
5839:  Func Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5840:  Func Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5841:  Func ValPrepS( const S : ShortString) : ShortString
5842:  Func CharStrS( C : AnsiChar; Len :Card) : ShortString
5843:  Func PadChS( const S : ShortString; C : AnsiChar; Len :Card):ShortString
5844:  Func PadS( const S : ShortString; Len :Card) : ShortString
5845:  Func LeftPadChS( const S : ShortString; C : AnsiChar; Len :Card):ShortString
5846:  Func LeftPadS( const S : ShortString; Len :Card) : ShortString
5847:  Func TrimLeadS( const S : ShortString) : ShortString
5848:  Func TrimTrailS( const S : ShortString) : ShortString
5849:  Func TrimS( const S : ShortString) : ShortString
5850:  Func TrimSpacesS( const S : ShortString) : ShortString
5851:  Func CenterChS( const S : ShortString; C : AnsiChar; Len :Card): ShortString
5852:  Func CenterS( const S : ShortString; Len :Card) : ShortString
5853:  Func EntabS( const S : ShortString; TabSize : Byte) : ShortString
5854:  Func DetabS( const S : ShortString; TabSize : Byte) : ShortString
5855:  Func ScrambleS( const S, Key : ShortString) : ShortString
5856:  Func SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5857:  Func FilterS( const S, Filters : ShortString) : ShortString
5858:  Func CharExistsS( const S : ShortString; C : AnsiChar) :Bool
5859:  Func CharCountS( const S : ShortString; C : AnsiChar) : Byte
5860:  Func WordCountS( const S, WordDelims : ShortString) :Card
5861:  Func WordPositionS(N:Card;const S, WordDelims:ShortString;var Pos:Card):Bool
5862:  Func ExtractWordS( N :Card; const S, WordDelims : ShortString) : ShortString
5863:  Func AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) :Card
5864:  Func AsciiPositionS(N:Card;const S,WordDelims:ShortStr;Quote:AnsiChar;var Pos:Card):Bool
```

```
5865:  Func ExtractAsciiS(N:Card;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5866:  Proc WordWrapS(const InSt:ShortString;var OutSt,Overlap:ShortString; Margin:Card;PadToMargin:Bool)
5867:  Func CompStringS( const S1, S2 : ShortString) : Int
5868:  Func CompUCStringS( const S1, S2 : ShortString) : Int
5869:  Func SoundexS( const S : ShortString) : ShortString
5870:  Func MakeLetterSetS( const S : ShortString) : Longint
5871:  Proc BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5872:  Func BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var Pos:Card):Bool;
5873:  Func BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortStr;var Pos:Card):Bool;
5874:  Func DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5875:  Func ForceExtensionS( const Name, Ext : ShortString) : ShortString
5876:  Func JustFilenameS( const PathName : ShortString) : ShortString
5877:  Func JustNameS( const PathName : ShortString) : ShortString
5878:  Func JustExtensionS( const Name : ShortString) : ShortString
5879:  Func JustPathnameS( const PathName : ShortString) : ShortString
5880:  Func AddBackSlashS( const DirName : ShortString) : ShortString
5881:  Func CleanPathNameS( const PathName : ShortString) : ShortString
5882:  Func HasExtensionS( const Name : ShortString; var DotPos :Card):Bool
5883:  Func CommaizeS( L : LongInt) : ShortString
5884:  Func CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5885:  Func FloatFormS(const Mask:ShortStr;R:TstFloat;const LtCurr,RtCurr:SString;Sep,DecPt:Char):ShortStr;
5886:  Func LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5887:  Func StrChPosS( const P : ShortString; C : AnsiChar; var Pos :Card) :Bool
5888:  Func StrStPosS( const P, S : ShortString; var Pos :Card) :Bool
5889:  Func StrStCopyS( const S : ShortString; Pos, Count :Card) : ShortString
5890:  Func StrChInsertS(const S : ShortString; C : AnsiChar; Pos :Card):ShortString
5891:  Func StrStInsertS(const S1, S2 : ShortString; Pos :Card) : ShortString
5892:  Func StrChDeleteS(const S: ShortString; Pos :Card) : ShortString
5893:  Func StrStDeleteS(const S: ShortString; Pos, Count :Card) : ShortString
5894:  Func ContainsOnlyS(const S, Chars : ShortString; var BadPos :Card) : Bool
5895:  Func ContainsOtherThanS( const S, Chars : ShortString; var BadPos :Card): Bool
5896:  Func CopyLeftS( const S : ShortString; Len :Card) : ShortString
5897:  Func CopyMidS( const S : ShortString; First, Len :Card) : ShortString
5898:  Func CopyRightS( const S : ShortString; First :Card) : ShortString
5899:  Func CopyRightAbsS( const S : ShortString; NumChars :Card) : ShortString
5900:  Func CopyFromNthWordS(const S,WordDelims:str;const AWord:str;N:Card;var SubString:ShortString):Bool;
5901:  Func DeleteFromNthWordS(const S,WordDelims:str;AWord:ShortString;N:Card;var SubStr:ShortString):Bool;
5902:  Func CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5903:  Func DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5904:  Func CopyWithinS( const S, Delimiter : ShortString; Strip :Bool) : ShortString
5905:  Func DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5906:  Func ExtractTokensS(const S,Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Card
5907:  Func IsChAlphaS( C : Char) :Bool
5908:  Func IsChNumericS( C : Char; const Numbers : ShortString) :Bool
5909:  Func IsChAlphaNumericS( C : Char; const Numbers : ShortString) :Bool
5910:  Func IsStrAlphaS( const S : Shortstring) :Bool
5911:  Func IsStrNumericS( const S, Numbers : ShortString) :Bool
5912:  Func IsStrAlphaNumericS( const S, Numbers : ShortString) :Bool
5913:  Func LastWordS(const S,WordDelims, AWord:ShortString; var Position :Card) :Bool
5914:  Func LastWordAbsS( const S, WordDelims : ShortString; var Position :Card) :Bool
5915:  Func LastStringS( const S, AString : ShortString; var Position :Card) :Bool
5916:  Func LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5917:  Func KeepCharsS( const S, Chars : ShortString) : ShortString
5918:  Func RepeatStringS(const RepeatString:ShortStr;var Repetitions:Card;MaxLen:Card):ShortStr;
5919:  Func ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Card;var Replacements:Card):ShortString;
5920:  Func ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Card):ShortString;
5921:  Func ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Card;var Replacements:Card):ShortString
5922:  Func ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var Replacements:Card):ShortString
5923:  Func RightTrimCharsS( const S, Chars : ShortString) : ShortString
5924:  Func StrWithinS(const S,SearchStr: ShortString;Start:Card;var Position:Card):bool
5925:  Func TrimCharsS( const S, Chars : ShortString) : ShortString
5926:  Func WordPosS(const S,WordDelims,AWord:ShortString;N:Card;var Position:Card):Bool
5927: end;
5928:
5929: ********unit uPSI_StUtils; from Systools4****************************************
5930: Func SignL( L : LongInt) : Int
5931: Func SignF( F : Extended) : Int
5932: Func MinWord(A, B : Word) : Word
5933: Func MidWord(W1, W2, W3 : Word) : Word
5934: Func MaxWord(A, B : Word) : Word
5935: Func MinLong(A, B : LongInt) : LongInt
5936: Func MidLong(L1, L2, L3 : LongInt) : LongInt
5937: Func MaxLong(A, B : LongInt) : LongInt
5938: Func MinFloat(F1, F2 : Extended) : Extended
5939: Func MidFloat(F1, F2, F3 : Extended) : Extended
5940: Func MaxFloat(F1, F2 : Extended) : Extended
5941: Func MakeInt16( H, L : Byte) : SmallInt
5942: Func MakeWordS( H, L : Byte) : Word
5943: Func SwapNibble( B : Byte) : Byte
5944: Func SwapWord( L : LongInt) : LongInt
5945: Proc SetFlag( var Flags : Word; FlagMask : Word)
5946: Proc ClearFlag( var Flags : Word; FlagMask : Word)
5947: Func FlagIsSet( Flags, FlagMask : Word) :Bool
5948: Proc SetByteFlag( var Flags : Byte; FlagMask : Byte)
5949: Proc ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5950: Func ByteFlagIsSet( Flags, FlagMask : Byte) :Bool
```

```
Proc SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
Proc ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
Func LongFlagIsSet( Flags, FlagMask : LongInt) :Bool
Proc ExchangeBytes( var I, J : Byte)
Proc ExchangeWords( var I, J : Word)
Proc ExchangeLongInts( var I, J : LongInt)
Proc ExchangeStructs( var I, J, Size :Card)
Proc FillWord( var Dest, Count :Card; Filler : Word)
Proc FillStruct( var Dest, Count :Card; var Filler, FillerSize :Card)
Func AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
//******************uPSI_StFIN;***********************************************
Func AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis:TStBasis): Extended
Func AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,Par:Extended;Frequency:TStFrequency;
Basis : TStBasis): Extended
Func BondDuration( Settlement,Maturity:TStDate;Rate,
Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
Func BondPrice(Settlement,Maturity:TStDate;Rate,Yield,
Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):Extended
Func CumulativeInterest(Rate:Extended;NPeriods:Int;PV:Extended;StartPeriod,
EndPeriod:Int;Frequency:TStFrequency; Timing:TStPaymentTime):Extended
Func CumulativePrincipal(Rate: Extended;NPeriods:Int;PV:Extended;StartPeriod,
EndPeriod:Int;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
Func DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
Func DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Int) : Extended
Func DiscountRate(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis): Extended;
Func DollarToDecimal( FracDollar : Extended; Fraction : Int) : Extended
Func DollarToDecimalText( DecDollar : Extended) :Str
Func DollarToFraction( DecDollar : Extended; Fraction : Int) : Extended
Func DollarToFractionStr( FracDollar : Extended; Fraction : Int) :Str
Func EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
Func FutureValueS(Rate:Extended;NPeriods:Int;Pmt,PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
Func FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
Func FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Int) : Extended
Func InterestRateS(NPeriods:Int;Pmt,PV,
FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
Func InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
Func InternalRateOfReturn16( const Values, NValues : Int; Guess : Extended) : Extended
Func IsCardValid( const S :Str) :Bool
Func ModifiedDuration(Settlement,Maturity:TStDate;Rate,
Yield:Extend;Freq:TStFrequency;Basis:TStBasis):Extended;
Func ModifiedIRR(const Values:array of Double;FinanceRate,ReinvestRate:Extended): Extended
Func ModifiedIRR16(const Values,NValues:Int; FinanceRate,ReinvestRate: Extended): Extended
Func NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
Func NetPresentValue16( Rate : Extended; const Values, NValues : Int) : Extended
Func NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
Func NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
Func NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
Func Payment(Rate:Extended;NPeriods:Int;PV
FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
Func Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Int;
Func PresentValueS(Rate:Extended;NPeriods:Int;Pmt,
FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
Func ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
Func RoundToDecimal( Value : Extended; Places : Int; Bankers :Bool) : Extended
Func TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
Func TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
Func TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
Func VariableDecliningBalance(Cost,Salvage:Extended;Life:Int;StartPeriod,EndPeriod,
Factor:Extended;NoSwitch:bool):Extended
Func YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Exted;Basis:TStBasis):Extended;
Func YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
Func YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;

//*********************************************unit uPSI_StAstroP;
Proc PlanetsPos( JD : Double; var PA : TStPlanetsArray)
//*****unit unit uPSI_StStat; Statistic Package of SysTools*******************
Func AveDev( const Data : array of Double) : Double
Func AveDev16( const Data, NData : Int) : Double
Func Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
Func Correlation( const Data1, Data2 : array of Double) : Double
Func Correlation16( const Data1, Data2, NData : Int) : Double
Func Covariance( const Data1, Data2 : array of Double) : Double
Func Covariance16( const Data1, Data2, NData : Int) : Double
Func DevSq( const Data : array of Double) : Double
Func DevSq16( const Data, NData : Int) : Double
Proc Frequency(const Data:array of Double;const Bins:array of Double;var Counts:array of LongInt);
//Proc Frequency16( const Data, NData : Int; const Bins, NBins : Int; var Counts)
Func GeometricMeanS( const Data : array of Double) : Double
Func GeometricMean16( const Data, NData : Int) : Double
Func HarmonicMeanS( const Data : array of Double) : Double
Func HarmonicMean16( const Data, NData : Int) : Double
Func Largest( const Data : array of Double; K : Int) : Double
Func Largest16( const Data, NData : Int; K : Int) : Double
Func MedianS( const Data : array of Double) : Double
Func Median16( const Data, NData : Int) : Double
Func Mode( const Data : array of Double) : Double
Func Mode16( const Data, NData : Int) : Double
Func Percentile( const Data : array of Double; K : Double) : Double
```

```
Func Percentile16( const Data, NData : Int; K : Double) : Double
Func PercentRank( const Data : array of Double; X : Double) : Double
Func PercentRank16( const Data, NData : Int; X : Double) : Double
Func Permutations( Number, NumberChosen : Int) : Extended
Func Combinations( Number, NumberChosen : Int) : Extended
Func FactorialS( N : Int) : Extended
Func Rank( Number : Double; const Data : array of Double; Ascending: Bool) : Int
Func Rank16( Number : Double; const Data, NData : Int; Ascending : Bool) : Int
Func Smallest( const Data : array of Double; K : Int) : Double
Func Smallest16( const Data, NData : Int; K : Int) : Double
Func TrimMean( const Data : array of Double; Percent : Double) : Double
Func TrimMean16( const Data, NData : Int; Percent : Double) : Double
  AddTypeS('TStLinEst', 'record B0:Double; B1 : double; seB0 : double; seB'
   +1 : Double; R2: Double;sigma :Double;SSr:double;SSe: Double; F0:Double;df:Int;end
Proc LinEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TStLinEst;ErrorStats:Bool;
Proc LogEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TStLinEst;ErrorStats:Bool;
Func Forecast(X: Double;const KnownY: array of Double;const KnownX:array of Double):Double
Func ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double
Func Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
Func RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
Func Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
Func StandardErrorY(const KnownY: array of Double;const KnownX:array of Double):Double
Func BetaDist( X, Alpha, Beta, A, B : Single) : Single
Func BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
Func BinomDist(NumberS,Trials: Int; ProbabilityS : Single;Cumulative:Bool):Single
Func CritBinom( Trials : Int; ProbabilityS, Alpha : Single) : Int
Func ChiDist( X : Single; DegreesFreedom : Int) : Single
Func ChiInv( Probability : Single; DegreesFreedom : Int) : Single
Func ExponDist( X, Lambda : Single; Cumulative :Bool) : Single
Func FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Int) : Single
Func FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Int) : Single
Func LogNormDist( X, Mean, StandardDev : Single) : Single
Func LogInv( Probability, Mean, StandardDev : Single) : Single
Func NormDist( X, Mean, StandardDev : Single; Cumulative :Bool) : Single
Func NormInv( Probability, Mean, StandardDev : Single) : Single
Func NormSDist( Z : Single) : Single
Func NormSInv( Probability : Single) : Single
Func Poisson( X : Int; Mean : Single; Cumulative :Bool) : Single
Func TDist( X : Single; DegreesFreedom : Int; TwoTails :Bool) : Single
Func TInv( Probability : Single; DegreesFreedom : Int) : Single
Func Erfc( X : Single) : Single
Func GammaLn( X : Single) : Single
Func LargestSort( const Data : array of Double; K : Int) : Double
Func SmallestSort( const Data: array of double; K : Int) : Double

Proc SIRegister_TStSorter(CL: TPSPascalCompiler);
 Func OptimumHeapToUse( RecLen :Card; NumRecs : LongInt) : LongInt
 Func MinimumHeapToUse( RecLen :Card) : LongInt
 Func MergeInfo( MaxHeap : LongInt; RecLen :Card; NumRecs : LongInt) : TMergeInfo
 Func DefaultMergeName( MergeNum : Int) :Str
 Proc ArraySort( var A, RecLen, NumRecs :Card; Compare:TUntypedCompareFunc)
Proc SIRegister_StAstro(CL: TPSPascalCompiler);
Func AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
 Func FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
 Func SunPos( UT : TStDateTimeRec) : TStPosRec
 Func SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
 Func SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
 Func Twilight(LD:TStDate; Longitude,Latitude:Double;TwiType:TStTwilight):TStRiseSetRec
 Func LunarPhase( UT : TStDateTimeRec) : Double
 Func MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
 Func MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
 Func FirstQuarter( D : TStDate) : TStLunarRecord
 Func FullMoon( D : TStDate) : TStLunarRecord
 Func LastQuarter( D : TStDate) : TStLunarRecord
 Func NewMoon( D : TStDate) : TStLunarRecord
 Func NextFirstQuarter( D : TStDate) : TStDateTimeRec
 Func NextFullMoon( D : TStDate) : TStDateTimeRec
 Func NextLastQuarter( D : TStDate) : TStDateTimeRec
 Func NextNewMoon( D : TStDate) : TStDateTimeRec
 Func PrevFirstQuarter( D : TStDate) : TStDateTimeRec
 Func PrevFullMoon( D : TStDate) : TStDateTimeRec
 Func PrevLastQuarter( D : TStDate) : TStDateTimeRec
 Func PrevNewMoon( D : TStDate) : TStDateTimeRec
 Func SiderealTime( UT : TStDateTimeRec) : Double
 Func Solstice( Y, Epoch : Int; Summer :Bool) : TStDateTimeRec
 Func Equinox( Y, Epoch : Int; Vernal :Bool) : TStDateTimeRec
 Func SEaster( Y, Epoch : Int) : TStDate
 Func DateTimeToAJD( D : TDateTime) : Double
 Func HoursMin( RA : Double) : ShortString
 Func DegsMin( DC : Double) : ShortString
 Func AJDToDateTime( D : Double) : TDateTime

Proc SIRegister_StDate(CL: TPSPascalCompiler);
Func CurrentDate : TStDate
Func StValidDate( Day, Month, Year, Epoch : Int) :Bool
Func DMYtoStDate( Day, Month, Year, Epoch : Int) : TStDate
Proc StDateToDMY( Julian : TStDate; var Day, Month, Year : Int)
Func StIncDate( Julian : TStDate; Days, Months, Years : Int) : TStDate
Func IncDateTrunc( Julian : TStDate; Months, Years : Int) : TStDate
```

```
Proc StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Int)
Func BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
Func WeekOfYear( Julian : TStDate) : Byte
Func AstJulianDate( Julian : TStDate) : Double
Func AstJulianDatetoStDate( AstJulian : Double; Truncate :Bool) : TStDate
Func AstJulianDatePrim( Year, Month, Date : Int; UT : TStTime) : Double
Func StDayOfWeek( Julian : TStDate) : TStDayType
Func DayOfWeekDMY( Day, Month, Year, Epoch : Int) : TStDayType
Func StIsLeapYear( Year : Int) :Bool
Func StDaysInMonth( Month : Int; Year, Epoch : Int) : Int
Func ResolveEpoch( Year, Epoch : Int) : Int
Func ValidTime( Hours, Minutes, Seconds : Int) :Bool
Proc StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
Func HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
Func CurrentTime : TStTime
Proc TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
Func StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
Func DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
Func RoundToNearestHour( T : TStTime; Truncate :Bool) : TStTime
Func RoundToNearestMinute( const T : TStTime; Truncate :Bool) : TStTime
Proc DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
Proc IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Int;Secs:LongInt)
Func DateTimeToStDate(DT : TDateTime) : TStDate
Func DateTimeToStTime(DT : TDateTime) : TStTime
Func StDateToDateTime(D : TStDate) : TDateTime
Func StTimeToDateTime(T : TStTime) : TDateTime
Func Convert2ByteDate(TwoByteDate : Word) : TStDate
Func Convert4ByteDate(FourByteDate : TStDate) : Word

Proc SIRegister_StDateSt(CL: TPSPascalCompiler);
Func DateStringHMStoAstJD( const Picture, DS :Str; H, M, S, Epoch : Int) : Double
Func MonthToString( const Month : Int) :Str
Func DateStringToStDate( const Picture, S :Str; Epoch : Int) : TStDate
Func DateStringToDMY(const Picture,S:str; Epoch:Int; var D, M, Y : Int):Boolean
Func StDateToDateString( const Picture:str;const Julian:TStDate;Pack:Bool):str
Func DayOfWeekToString( const WeekDay : TStDayType) :Str
Func DMYtoDateString(const Picture:str;Day,Month,Year,Epoch:Int;Pack:Boolean):str);
Func CurrentDateString( const Picture :Str; Pack :Bool) :Str
Func CurrentTimeString( const Picture :Str; Pack :Bool) :Str
Func TimeStringToHMS( const Picture, St :Str; var H, M, S : Int) :Bool
Func TimeStringToStTime( const Picture, S :Str) : TStTime
Func StTimeToAmPmString(const Picture:str; const T:TStTime; Pack:Bool) :Str
Func StTimeToTimeString(const Picture:Str; const T:TStTime;Pack :Bool) :Str
Func DateStringIsBlank( const Picture, S :Str) :Bool
Func InternationalDate( ForceCentury :Bool) :Str
Func InternationalLongDate( ShortNames :Bool; ExcludeDOW :Bool) :Str
Func InternationalTime( ShowSeconds :Bool) :Str
Proc ResetInternationalInfo

Proc SIRegister_StBase(CL: TPSPascalCompiler);
 Func DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) :Bool
 Func AnsiUpperCaseShort32( const S :Str) :Str
 Func AnsiCompareTextShort32( const S1, S2 :Str) : Int
 Func AnsiCompareStrShort32( const S1, S2 :Str) : Int
 Func HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
 Func HugeDecompressRLE(const InBuffer,InLen:Longint;var OutBuffer,OutLen:LongInt):Longint
 Proc HugeFillChar( var Dest, Count : Longint; Value : Byte)
 Proc HugeFillStruc( var Dest, Count : Longint; const Value, ValSize :Card)
 Func Upcase( C : AnsiChar) : AnsiChar
 Func LoCase( C : AnsiChar) : AnsiChar
 Func CompareLetterSets( Set1, Set2 : LongInt) :Card
 Func CompStruct( const S1, S2, Size :Card) : Int
 Func Search(const Buffer,BufLength:Card;const Match,MatLength:Card;var Pos:Cardi):Bool;
 Func StSearch(const Buff,BufLength:Card;const Match,MatLength:Card;var Pos:Cardi):Bool
 Func SearchUC(const Buff,BufLength:Card;const Match,MatLength:Card;var Pos:Cardi):Bool
 Func IsOrInheritsFrom( Root, Candidate : TClass) :Bool
 Proc RaiseContainerError( Code : longint)
 Proc RaiseContainerErrorFmt( Code : Longint; Data : array of const)
 Func ProductOverflow( A, B : LongInt) :Bool
 Func StNewStr( S :Str) : PShortString
 Proc StDisposeStr( PS : PShortString)
 Proc ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : Int)
 Proc ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : Int)
 Proc ValWord( const S : ShortString; var Wd : word; var ErrorCode : Int)
 Proc RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
 Proc RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
 Proc RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info :Str)

Proc SIRegister_usvd(CL: TPSPascalCompiler);
begin
Proc SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Int; S : TVector; V : TMatrix)
Proc SV_SetZero( S : TVector; Lb, Ub : Int; Tol : Float)
Proc SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Int;X:TVector);
Proc SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Int; A : TMatrix)
Proc RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
end;

//**********unit unit ; StMath Package of SysTools*****************************
Func IntPowerS( Base : Extended; Exponent : Int) : Extended
```

```
6207: Func PowerS( Base, Exponent : Extended) : Extended
6208: Func StInvCos( X : Double) : Double
6209: Func StInvSin( Y : Double) : Double
6210: Func StInvTan2( X, Y : Double) : Double
6211: Func StTan( A : Double) : Double
6212: Proc DumpException;  //unit StExpEng;
6213: Func HexifyBlock( var Buffer, BufferSize : Int):Str
6214:
6215: //**********unit unit ; StCRC Package of SysTools*****************************
6216: Func Adler32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
6217: Func Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6218: Func Adler32OfFile( FileName : Ansistr) : LongInt
6219: Func Crc16Prim( var Data, DataSize, CurCrc :Card) :Card
6220: Func Crc16OfStream( Stream : TStream; CurCrc :Card) :Card
6221: Func Crc16OfFile( FileName : Ansistr) :Card
6222: Func Crc32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
6223: Func Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6224: Func Crc32OfFile( FileName : Ansistr) : LongInt
6225: Func InternetSumPrim( var Data, DataSize, CurCrc :Card) :Card
6226: Func InternetSumOfStream( Stream : TStream; CurCrc :Card) :Card
6227: Func InternetSumOfFile( FileName : Ansistr) :Card
6228: Func Kermit16Prim( var Data, DataSize, CurCrc :Card) :Card
6229: Func Kermit16OfStream( Stream : TStream; CurCrc :Card) :Card
6230: Func Kermit16OfFile( FileName : Ansistr) :Card
6231:
6232: //**********unit unit ; StBCD Package of SysTools*****************************************
6233: Func AddBcd( const B1, B2 : TbcdS) : TbcdS
6234: Func SubBcd( const B1, B2 : TbcdS) : TbcdS
6235: Func MulBcd( const B1, B2 : TbcdS) : TbcdS
6236: Func DivBcd( const B1, B2 : TbcdS) : TbcdS
6237: Func ModBcd( const B1, B2 : TbcdS) : TbcdS
6238: Func NegBcd( const B : TbcdS) : TbcdS
6239: Func AbsBcd( const B : TbcdS) : TbcdS
6240: Func FracBcd( const B : TbcdS) : TbcdS
6241: Func IntBcd( const B : TbcdS) : TbcdS
6242: Func RoundDigitsBcd( const B : TbcdS; Digits :Card): TbcdS
6243: Func RoundPlacesBcd( const B : TbcdS; Places :Card : TbcdS
6244: Func ValBcd( const S :Str) : TbcdS
6245: Func LongBcd( L : LongInt) : TbcdS
6246: Func ExtBcd( E : Extended) : TbcdS
6247: Func ExpBcd( const B : TbcdS) : TbcdS
6248: Func LnBcd( const B : TbcdS) : TbcdS
6249: Func IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6250: Func PowBcd( const B, E : TbcdS) : TbcdS
6251: Func SqrtBcd( const B : TbcdS) : TbcdS
6252: Func CmpBcd( const B1, B2 : TbcdS) : Int
6253: Func EqDigitsBcd( const B1, B2 : TbcdS; Digits :Card) :Bool
6254: Func EqPlacesBcd( const B1, B2 : TbcdS; Digits :Card) :Bool
6255: Func IsIntBcd( const B : TbcdS) :Bool
6256: Func TruncBcd( const B : TbcdS) : LongInt
6257: Func BcdExt( const B : TbcdS) : Extended
6258: Func RoundBcd( const B : TbcdS) : LongInt
6259: Func StrBcd( const B : TbcdS; Width, Places :Card) :Str
6260: Func StrExpBcd( const B : TbcdS; Width :Card) :Str
6261: Func FormatBcd( const Format :Str; const B : TbcdS) :Str
6262: Func StrGeneralBcd( const B : TbcdS) :Str
6263: Func FloatFormBcd(const Mask:str;B:TbcdS;const LtCurr,RtCurr:str;Sep,DecPt:AnsiChar):str
6264: Proc ConvertBcd( const SrcB, SrcSize: Byte; var DestB, DestSize : Byte)
6265:
6266: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6267: Proc StParseLine( const Data : Ansistr; Schema : TStTextDataSchema; Result : TStrings)
6268: Func StFieldTypeToStr( FieldType : TStSchemaFieldType) : Ansistr
6269: Func StStrToFieldType( const S : Ansistr) : TStSchemaFieldType
6270: Func StDeEscape( const EscStr : Ansistr) : Char
6271: Func StDoEscape( Delim : Char) : Ansistr
6272: Func StTrimTrailingChars( const S : Ansistr; Trailer : Char) : Ansistr
6273: Func AnsiHashText( const S :Str; Size : Int) : Int
6274: Func AnsiHashStr( const S :Str; Size : Int) : Int
6275: Func AnsiELFHashText( const S :Str; Size : Int) : Int
6276: Func AnsiELFHashStr( const S :Str; Size : Int) : Int
6277:
6278: //**********unit unit ; StNetCon Package of SysTools*****************************************
6279: with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6280: Constructor Create( AOwner : TComponent)
6281: Func Connect : DWord
6282: Func Disconnect : DWord
6283: RegisterProperty('Password', 'String', iptrw);
6284: Property(UserName', 'String', iptrw);
6285: Property(ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6286: Property(DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6287: Property(LocalDevice', 'String', iptrw);
6288: Property(ServerName', 'String', iptrw);
6289: Property(ShareName', 'String', iptrw);
6290: Property(OnConnect', 'TNotifyEvent', iptrw);
6291: Property(OnConnectFail', 'TOnConnectFailEvent', iptrw);
6292: Property(OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6293: Property(OnDisconnect', 'TNotifyEvent', iptrw);
6294: Property(OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6295: Property(OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
```

```
6296: end;
6297: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6298: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6299: Proc InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6300: Proc EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6301: Proc LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6302: Func InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6303: Func SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6304: Func TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6305: Proc DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6306: Func GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6307: Func SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6308: Func SuspendThread( hThread : THandle) : DWORD
6309: Func ResumeThread( hThread : THandle) : DWORD
6310: Func CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6311: Func GetCurrentThread : THandle
6312: Proc ExitThread( dwExitCode : DWORD)
6313: Func TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6314: Func GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6315: Proc EndThread(ExitCode: Int);
6316: Func WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6317: Func MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6318: Proc FreeProcInstance( Proc : FARPROC)
6319: Proc FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6320: Func DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6321: Proc ParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Pointer; ASafeSection:Bool);
6322: Proc ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection :Bool);
6323: Proc ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6324: Proc ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: bool;
6325: Func CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6326: Func CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection :Bool) : TParallelJob;
6327: Func CurrentParallelJobInfo : TParallelJobInfo
6328: Func ObtainParallelJobInfo : TParallelJobInfo
6329: Proc GetSystemInfo( var lpSystemInfo : TSystemInfo);
6330: Func IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL;
6331: Func SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL;
6332: Func
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL;
6333: Func SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL;
6334: Func DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL;dwOptions:DWORD):BOOL;
6335: Func GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL;
6336: Func SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL;
6337:
6338: ****************************************************unit uPSI_JclMime;
6339:  Func MimeEncodeString( const S : Ansistr) : Ansistr
6340:  Func MimeDecodeString( const S : Ansistr) : Ansistr
6341:  Proc MimeEncodeStream(const InputStream:TStream; const OutputStream : TStream)
6342:  Proc MimeDecodeStream(const InputStream:TStream; const OutputStream : TStream)
6343:  Func MimeEncodedSize( const I :Card) :Card
6344:  Func MimeDecodedSize( const I :Card) :Card
6345:  Proc MimeEncode(var InputBuffer:Str;const InputByteCount:Card;var OutputBuffer)
6346:  Func MimeDecode(var InputBuffer:str;const InputBytesCount:Card;var OutputBuffer):Cardi;
6347:  Func MimeDecodePartial(var InputBuffer:str;const InputBytesCount:Card;var OutputBuffer:str;var
      ByteBuffer:Card;var ByteBufferSpace:Card):Card
6348:  Func MimeDecodePartialEnd(var OutputBuf:str;const ByteBuf:Card;const ByteBufferSpace:Card):Cardi;
6349:
6350: ****************************************************unit uPSI_JclPrint;
6351:  Proc DirectPrint( const Printer, Data :Str)
6352:  Proc SetPrinterPixelsPerInch
6353:  Func GetPrinterResolution : TPoint
6354:  Func CharFitsWithinDots( const Text :Str; const Dots : Int) : Int
6355:  Proc PrintMemo( const Memo : TMemo; const Rect : TRect)
6356:
6357: //*******************************unit uPSI_ShLwApi;***************************************
6358:  Func StrChr( lpStart : PChar; wMatch : WORD) : PChar
6359:  Func StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6360:  Func StrCmpN( lpStr1, lpStr2 : PChar; nChar : Int) : Int
6361:  Func StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Int) : Int
6362:  Func StrCSpn( lpStr_, lpSet : PChar) : Int
6363:  Func StrCSpnI( lpStr1, lpSet : PChar) : Int
6364:  Func StrDup( lpSrch : PChar) : PChar
6365:  Func StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6366:  Func StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6367:  Func StrFromTimeInterval(pszOut:PChar cchMax:UINT;dwTimeMS:DWORD;digits:Int):Int
6368:  Func StrIsIntlEqual(fCaseSens: BOOL;lpString1,lpString2:PChar; nChar:Int): BOOL
6369:  Func StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
6370:  Func StrPBrk( psz, pszSet : PChar) : PChar
6371:  Func StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6372:  Func StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6373:  Func StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6374:  Func StrSpn( psz, pszSet : PChar) : Int
6375:  Func StrStr( lpFirst, lpSrch : PChar) : PChar
6376:  Func StrStrI( lpFirst, lpSrch : PChar) : PChar
6377:  Func StrToInt( lpSrch : PChar) : Int
6378:  Func StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Int) : BOOL
6379:  Func StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
```

```
Func ChrCmpI( w1, w2 : WORD) : BOOL
Func ChrCmpIA( w1, w2 : WORD) : BOOL
Func ChrCmpIW( w1, w2 : WORD) : BOOL
Func StrIntlEqN( s1, s2 : PChar; nChar : Int) : BOOL
Func StrIntlEqNI( s1, s2 : PChar; nChar : Int) : BOOL
Func StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Int) : PChar
Func StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
Func IntlStrEqWorker(fCaseSens:BOOL;lpString1,lpString2:PChar;nChar:Int):BOOL
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);
```

```
6469:  URL_SCHEME_WAIS','LongInt'( 8);
6470:  URL_SCHEME_FILE','LongInt'( 9);
6471:  URL_SCHEME_MK','LongInt'( 10);
6472:  URL_SCHEME_HTTPS','LongInt'( 11);
6473:  URL_SCHEME_SHELL','LongInt'( 12);
6474:  URL_SCHEME_SNEWS','LongInt'( 13);
6475:  URL_SCHEME_LOCAL','LongInt'( 14);
6476:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6477:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6478:  URL_SCHEME_ABOUT','LongInt'( 17);
6479:  URL_SCHEME_RES','LongInt'( 18);
6480:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6481:  URL_SCHEME', 'Int
6482:  URL_PART_NONE','LongInt'( 0);
6483:  URL_PART_SCHEME','LongInt'( 1);
6484:  URL_PART_HOSTNAME','LongInt'( 2);
6485:  URL_PART_USERNAME','LongInt'( 3);
6486:  URL_PART_PASSWORD','LongInt'( 4);
6487:  URL_PART_PORT','LongInt'( 5);
6488:  URL_PART_QUERY','LongInt'( 6);
6489:  URL_PART', 'DWORD
6490:  URLIS_URL','LongInt'( 0);
6491:  URLIS_OPAQUE','LongInt'( 1);
6492:  URLIS_NOHISTORY','LongInt'( 2);
6493:  URLIS_FILEURL','LongInt'( 3);
6494:  URLIS_APPLIABLE','LongInt'( 4);
6495:  URLIS_DIRECTORY','LongInt'( 5);
6496:  URLIS_HASQUERY','LongInt'( 6);
6497:  TUrlIs', 'DWORD
6498:  URL_UNESCAPE','LongWord( $10000000);
6499:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6500:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6501:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6502:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6503:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6504:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6505:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6506:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6507:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6508:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6509:  URL_INTERNAL_PATH','LongWord( $00800000);
6510:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6511:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6512:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6513:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6514:  URL_APPLY_DEFAULT','LongWord( $00000001);
6515:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6516:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6517:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6518:  Func UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Int
6519:  Func UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6520:  Func UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6521:  Func UrlIsOpaque( pszURL : PChar) : BOOL
6522:  Func UrlIsNoHistory( pszURL : PChar) : BOOL
6523:  Func UrlIsFileUrl( pszURL : PChar) : BOOL
6524:  Func UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6525:  Func UrlGetLocation( psz1 : PChar) : PChar
6526:  Func UrlUnescape(pszUrl,pszUnescaped:PChar;pcchUnescaped:DWORD; dwFlags: DWORD): HRESULT
6527:  Func UrlEscape(pszUrl:PChar; pszEscaped:PChar; pcchEscaped:DWORD;dwFlags:DWORD): HRESULT
6528:  Func UrlCreateFromPath(pszPath:PChar;pszUrl:PChar;pcchUrl: DWORD;dwFlags:DWORD): HRESULT
6529:  Func PathCreateFromUrl(pszUrl:PChar; pszPath:PChar;pcchPath:DWORD; dwFlags:DWORD):HRESULT
6530:  Func UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6531:  Func UrlGetPart(pszIn:PChar; pszOut:PChar;pcchOut:DWORD; dwPart,dwFlags: DWORD): HRESULT
6532:  Func UrlApplyScheme(pszIn:PChar; pszOut:PChar; pcchOut : DWORD; dwFlags: DWORD): HRESULT
6533:  Func HashData(pbData:BYTE;cbData:DWORD; pbHash:BYTE; cbHash: DWORD) : HRESULT
6534:  Func UrlEscapeSpaces(pszUrl:PChar;pszEscaped:PChar;pcchEscaped: WORD): HRESULT
6535:  Func UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6536:  Func SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6537:  Func SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6538:  Func SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6539:  Func SHEnumKeyEx(hKey:HKEY; dwIndex : DWORD; pszName:PChar;var pcchName:DWORD) : Longint
6540:  Func SHEnumValue(hKey:HKEY;dwIndex:DWORD;pszValueName:PChar;var pcchValueName:DWORD;pdwType:DWORD;pvData:
      ___Pointer; pcbData:DWORD):Longint
6541:  Func SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6542:  Func SHCopyKey(hkeySrc:HKEY;szSrcSubKey:PChar; hkeyDest:HKEY; fReserved:DWORD) : DWORD
6543:  Func SHRegGetPath(hKey:HKEY;pcszSubKey,pcszValue:PChar;pszPath:PChar;dwFlags:DWORD):DWORD
6544:  Func SHRegSetPath(hKey:HKEY;pcszSubKey,pcszValue,pcszPath:PChar;dwFlags:DWORD): DWORD
6545:  SHREGDEL_DEFAULT','LongWord( $00000000);
6546:  SHREGDEL_HKCU','LongWord( $00000001);
6547:  SHREGDEL_HKLM','LongWord( $00000010);
6548:  SHREGDEL_BOTH','LongWord( $00000011);
6549:  SHREGENUM_DEFAULT','LongWord( $00000000);
6550:  SHREGENUM_HKCU','LongWord( $00000001);
6551:  SHREGENUM_HKLM','LongWord( $00000010);
6552:  SHREGENUM_BOTH','LongWord( $00000011);
6553:  SHREGSET_HKCU','LongWord( $00000001);
6554:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6555:  SHREGSET_HKLM','LongWord( $00000004);
6556:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
```

```
TSHRegDelFlags', 'DWORD
TSHRegEnumFlags', 'DWORD
HUSKEY', 'THandle
ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
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
Func ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
Proc SHFreeNameMappings( hNameMappings : THandle)

DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
DLLVER_PLATFORM_NT','LongWord( $00000002);
DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
Func MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
Func SimpleXMLEncode( const S :Str) :Str
Proc SimpleXMLDecode( var S :Str; TrimBlanks :Bool)
Func XMLEncode( const S :Str) :Str
Func XMLDecode( const S :Str) :Str
Func EntityEncode( const S :Str) :Str
Func EntityDecode( const S :Str) :Str

Proc RIRegister_CPort_Routines(S: TPSExec);
 Proc EnumComPorts( Ports : TStrings)
 Proc ListComPorts( Ports : TStrings)
 Proc ComPorts( Ports : TStrings)  //Alias to Arduino
 Func GetComPorts: TStringlist;
 Func StrToBaudRate( Str :Str) : TBaudRate
 Func StrToStopBits( Str :Str) : TStopBits
 Func StrToDataBits( Str :Str) : TDataBits
 Func StrToParity( Str :Str) : TParityBits
 Func StrToFlowControl( Str :Str) : TFlowControl
 Func BaudRateToStr( BaudRate : TBaudRate) :Str
 Func StopBitsToStr( StopBits : TStopBits) :Str
 Func DataBitsToStr( DataBits : TDataBits) :Str
 Func ParityToStr( Parity : TParityBits) :Str
 Func FlowControlToStr( FlowControl : TFlowControl) :Str
 Func ComErrorsToStr( Errors : TComErrors) :Str
 Func GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
 Func DispatchMessage( const lpMsg : TMsg) : Longint
 Func TranslateMessage( const lpMsg : TMsg) : BOOL
 Func SetMessageQueue( cMessagesMax : Int) : BOOL
 Func PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
 Func GetMessagePos : DWORD
 Func GetMessageTime : Longint
 Func GetMessageExtraInfo : Longint
 Func GetSpecialFolderPath( const FolderName :Str; CanCreate :Bool) :Str
 Proc JAddToRecentDocs( const Filename :Str)
 Proc ClearRecentDocs
 Func ExtractIconFromFile( FileName :Str; Index : Int) : HICON
 Func CreateShellLink( const AppName, Desc :Str; Dest :Str) :Str
 Proc GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
 Proc SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
 Func RecycleFile( FileToRecycle :Str) :Bool
 Func JCopyFile( FromFile, ToDir :Str) :Bool
 Func ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
 Func ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
 Func QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel: DWORD; lpBuffer : LPBYTE; cbBufSize :
DWORD; var pcbBytesNeeded : DWORD) : BOOL
 Func QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel: DWORD; lpBuffer : LPBYTE; cbBufSize :
DWORD; var pcbBytesNeeded : DWORD) : BOOL
 Func QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel:DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD;
var pcbBytesNeeded : DWORD) : BOOL
 Func EnumServicesStatusExA(hSCManager:SC_HANDLE;InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
lpResumeHandle:DWORD;pszGroupName: LP
 Func EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
dwServiceState : DWORD; lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
lpResumeHandle:DWORD; pszGroupNam
 Func EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
dwServiceState:DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
lpResumeHandle:DWORD; pszGroupName
 Func ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL

****************************************** unit uPSI_JclPeImage;
 Func IsValidPeFile( const FileName : TFileName) :Bool
// Func PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders): Bool
 Func PeCreateNameHintTable( const FileName : TFileName) :Bool
 Func PeRebaseImage(const ImageName:TFileName; NewBase:DWORD; TimeStamp : DWORD; MaxNewSize:DWORD):
TJclRebaseImageInfo
 Func PeVerifyCheckSum( const FileName : TFileName) :Bool
 Func PeClearCheckSum( const FileName : TFileName) :Bool
 Func PeUpdateCheckSum( const FileName : TFileName) :Bool
 Func PeDoesExportFunction(const FileName:TFileName;const FuncName:str;Options:TJclSmartCompOptions):Bool;
 Func PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName:str; var
ForwardedName:str;Options:TJclSmartCompOptions):Boolean
```

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

```
6804:  Func IsFileSystem( const S : Ansistr) : Int
6805:  Func LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Int) : TStDateTimeRec
6806:  Func ReadVolumeLabel( var VolName : Ansistr; Drive : Char) :Card
6807:  Func SameFile( const FilePath1, FilePath2 : Ansistr; var ErrorCode : Int) :Bool
6808:  Func SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) :Card
6809:  Proc SplitPath( const APath : Ansistr; Parts : TStrings)
6810:  Func StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6811:  Func StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6812:  Func UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6813:  Func ValidDrive( Drive : Char) :Bool
6814:  Func WriteVolumeLabel( const VolName : Ansistr; Drive : Char) :Card
6815:
6816: //****************************unit uPSI_JclLANMan;*****************************************
6817: Func CreateAccount( const Server,Username,Fullname,Password,Description,Homedir,Script :str; const
      PasswordNeverExpires : Bool) : Bool
6818:  Func CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script :Str; const
      PasswordNeverExpires :Bool) :Bool
6819:  Func DeleteAccount( const Servername, Username :Str) :Bool
6820:  Func DeleteLocalAccount( Username :Str) :Bool
6821:  Func CreateLocalGroup( const Server, Groupname, Description :Str) :Bool
6822:  Func CreateGlobalGroup( const Server, Groupname, Description :Str) :Bool
6823:  Func DeleteLocalGroup( const Server, Groupname :Str) :Bool
6824:  Func GetLocalGroups( const Server :Str; const Groups : TStrings) :Bool
6825:  Func GetGlobalGroups( const Server :Str; const Groups : TStrings) :Bool
6826:  Func LocalGroupExists( const Group :Str) :Bool
6827:  Func GlobalGroupExists( const Server, Group :Str) :Bool
6828:  Func AddAccountToLocalGroup( const Accountname, Groupname :Str) :Bool
6829:  Func LookupGroupName( const Server :Str; const RID : TNetWellKnownRID) :Str
6830:  Proc ParseAccountName( const QualifiedName :Str; var Domain, UserName :Str)
6831:  Func IsLocalAccount( const AccountName :Str) :Bool
6832:  Func TimeStampInterval( StartStamp, EndStamp : TDateTime) : Int
6833:  Func GetRandomString( NumChar :Card) :Str
6834:
6835: //****************************unit uPSI_cUtils;*****************************************
6836: TypeS(TUnitType', ( utSrc, utHead, utRes, utPrj, utOther )
6837: Func cIsWinNT :Bool
6838:  Proc cFilesFromWildcard(Directory,Mask:Str;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Boolean;
6839:  Func cExecuteFile( const FileName, Params, DefaultDir :Str; ShowCmd : Int) : THandle
6840:  Func cRunAndGetOutput(Cmd,WorkDir:str; ErrFunc:TErrFunc;LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc:TCheckAbortFunc;ShowReturnValue:Bool):str
6841:  Func cGetShortName( FileName :Str) :Str
6842:  Proc cShowError( Msg :Str)
6843:  Func cCommaStrToStr( s :Str; formatstr :Str) :Str
6844:  Func cIncludeQuoteIfSpaces( s :Str) :Str
6845:  Func cIncludeQuoteIfNeeded( s :Str) :Str
6846:  Proc cLoadFilefromResource( const FileName :Str; ms : TMemoryStream)
6847:  Func cValidateFile(const FileName:str;const WorkPath:str;const CheckDirs:bool):str;
6848:  Func cBuildFilter( var value :Str; const FLTStyle : TFILTERSET) :Bool;
6849:  Func cBuildFilter1( var value :Str; const _filters : array of string) :Bool;
6850:  Func cCodeInstoStr( s :Str) :Str
6851:  Func cStrtoCodeIns( s :Str) :Str
6852:  Proc cStrtoAttr( var Attr : TSynHighlighterAttributes; Value :Str)
6853:  Func cAttrtoStr( const Attr : TSynHighlighterAttributes) :Str
6854:  Proc cStrtoPoint( var pt : TPoint; value :Str)
6855:  Func cPointtoStr( const pt : TPoint) :Str
6856:  Func cListtoStr( const List : TStrings) :Str
6857:  Func ListtoStr( const List : TStrings) :Str
6858:  Proc StrtoList( s :Str; const List : TStrings; const delimiter : char)
6859:  Proc cStrtoList( s :Str; const List : TStrings; const delimiter : char)
6860:  Func cGetFileTyp( const FileName :Str) : TUnitType
6861:  Func cGetExTyp( const FileName :Str) : TExUnitType
6862:  Proc cSetPath( Add :Str; const UseOriginal :Bool)
6863:  Func cExpandFileto( const FileName :Str; const BasePath :Str) :Str
6864:  Func cFileSamePath( const FileName :Str; const TestPath :Str) :Bool
6865:  Proc cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6866:  Func cGetLastPos( const SubStr :Str; const S :Str) : Int
6867:  Func cGenMakePath( FileName :Str) :Str;
6868:  Func cGenMakePath2( FileName :Str) :Str
6869:  Func cGenMakePath1(FileName:Str; EscapeSpaces,EncloseInQuotes:Bool):str;
6870:  Func cGetRealPath( BrokenFileName :Str; Directory :Str) :Str
6871:  Func cCalcMod( Count : Int) : Int
6872:  Func cGetVersionString( FileName :Str) :Str
6873:  Func cCheckChangeDir( var Dir :Str) :Bool
6874:  Func cGetAssociatedProgram(const Extension:str;var Filename,Description:str):bool
6875:  Func cIsNumeric( s :Str) :Bool
6876:  Proc StrtoAttr( var Attr : TSynHighlighterAttributes; Value :Str)
6877:  Func AttrtoStr( const Attr : TSynHighlighterAttributes) :Str
6878:  Func GetFileTyp( const FileName :Str) : TUnitType
6879:  Func Atoi(const aStr:Str): Int
6880:  Func Itoa(const aint: Int):Str
6881:  Func Atof(const aStr:Str): double;
6882:  Func Atol(const aStr:Str): longint;
6883:
6884: Proc SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6885: begin
6886:   FindClass('TOBJECT'),'EHTTP
6887:   FindClass('TOBJECT'),'EHTTPParser
6888:   //AnsiCharSet', 'set of AnsiChar
6889:   AnsistrArray', 'array of Ansistr
```

```
THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
 +'TTPProtocolEnum; CustomProtocol : Ansistr; CustomMajVersion : Int; '
 +'CustomMinVersion : Int; end
THTTPHeaderNameEnum','( hntCustom, hntHost, hntContentType, hnt'
 +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
 +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
 +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
 +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
 +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
 +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
 +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
 +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
 +'nection, hntOrigin, hntKeepAlive )
THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: Ansistr; end
THTTPCustomHeader', 'record FieldName : Ansistr; FieldValue :'
 +' Ansistr; end
//PHTTPCustomHeader', '^THTTPCustomHeader // will not work
THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
//PHTTPContentLength', '^THTTPContentLength // will not work
THTTPContentTypeMajor','( hctmCustom, hctmText, hctmImage )
THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
 +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
 +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
 +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
 +'ationCustom, hctAudioCustom, hctVideoCustom )
THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
 +ajor:Ansistr;CustomMinor:Ansistr;Parameters:AnsistrArray;CustomStr:Ansistr;end
THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
 +' Int; Day : Int; Month : Int; Year : Int; Hour : Int; '
 +'Min : Int; Sec : Int; TimeZoneGMT :Bool; CustomTimeZone : Ansi'
 +'String; DateTime : TDateTime; Custom : Ansistr; end
THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum;Custom:Ansistr;end
THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;Custom:Ansistr; end
THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:Ansistr; end
THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
 +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
 +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
 +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
 +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
THTTPContentEncoding','record Value:THTTPContentEncodingEnum;Custom:Ansistr;end;
THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
 +'FieldEnum; List : array of THTTPContentEncoding; end
THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
 +' Custom : Ansistr; Date : TDateTime; Seconds : Int64; end
THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
 +num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom: Ansistr; end
THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
THTTPSetCookieCustomField', 'record Name : Ansistr; Value : Ansistr; end
THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
 +'omain : Ansistr; Path : Ansistr; Expires : THTTPDateField; MaxAge : '
 +'Int64; HttpOnly :Bool; Secure :Bool; CustomFields : THTTPSetCookie'
 +'CustomFieldArray; Custom : Ansistr; end
//PHTTPSetCookieField', '^THTTPSetCookieField // will not work
THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
THTTPCookieFieldEntry', 'record Name : Ansistr; Value : Ansistr; end
//PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
 +' : THTTPCookieFieldEntryArray; Custom : Ansistr; end
THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
 +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
 +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
 +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
THTTPCustomHeaders', 'array of THTTPCustomHeader
 //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of Ansistr
THTTPFixedHeaders','array[0..42] of Ansistr
 THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
 +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
THTTPMethod', 'record Value : THTTPMethodEnum; Custom : Ansistr; end
THTTPRequestStartLine','record Method:THTTPMethod;URI:Ansistr;Version:THTTPVersion;end
THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
 +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
 +'kie:THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
```

```
6979:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6980:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6981:    +' : THTTPRequestHeader; HeaderComplete :Bool; HasContent :Bool; end
6982:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6983:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6984:    +'Int; Msg : THTTPResponseStartLineMessage; CustomMsg : Ansistr; end
6985:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6986:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6987:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6988:    +' THTTPDateField; Age : THTTPAgeField; end
6989:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6990:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6991:    +er : THTTPResponseHeader; HeaderComplete :Bool; HasContent :Bool; end
6992:  Func HTTPMessageHasContent( const H : THTTPCommonHeaders) :Bool
6993:  Proc InitHTTPRequest( var A : THTTPRequest)
6994:  Proc InitHTTPResponse( var A : THTTPResponse)
6995:  Proc ClearHTTPVersion( var A : THTTPVersion)
6996:  Proc ClearHTTPContentLength( var A : THTTPContentLength)
6997:  Proc ClearHTTPContentType( var A : THTTPContentType)
6998:  Proc ClearHTTPDateField( var A : THTTPDateField)
6999:  Proc ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
7000:  Proc ClearHTTPConnectionField( var A : THTTPConnectionField)
7001:  Proc ClearHTTPAgeField( var A : THTTPAgeField)
7002:  Proc ClearHTTPContentEncoding( var A : THTTPContentEncoding)
7003:  Proc ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
7004:  Proc ClearHTTPContentRangeField( var A : THTTPContentRangeField)
7005:  Proc ClearHTTPSetCookieField( var A : THTTPSetCookieField)
7006:  Proc ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
7007:  //Proc ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
7008:  Proc ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
7009:  Proc ClearHTTPCookieField( var A : THTTPCookieField)
7010:  Proc ClearHTTPMethod( var A : THTTPMethod)
7011:  Proc ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
7012:  Proc ClearHTTPRequestHeader( var A : THTTPRequestHeader)
7013:  Proc ClearHTTPRequest( var A : THTTPRequest)
7014:  Proc ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
7015:  Proc ClearHTTPResponseHeader( var A : THTTPResponseHeader)
7016:  Proc ClearHTTPResponse( var A : THTTPResponse)
7017:   THTTPStringOption', '( hsoNone )
7018:   THTTPStringOptions', 'set of THTTPStringOption
7019:   FindClass('TOBJECT'),'TAnsistrBuilder
7020:  Proc BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsistrBuilder; PTHTTPStringOptions;
7021:  Proc BuildStrHTTPContentLengthValue(const A:THTTPContentLength;B:TAnsistrBuilder;P:THTTPStringOptions)
7022:  Proc BuildStrHTTPContentLength(const A : THTTPContentLength; B:TAnsistrBuilder;PTHTTPStringOptions)
7023:  Proc BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsistrBuilder;const P:THTTPStringOptions)
7024:  Proc BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsistrBuilder; const P:THTTPStringOptions)
7025:  Proc BuildStrRFCDateTime(const DOW,Da,Mo,Ye,Ho,Mi,Se:Int;const TZ:Ansistr;const B:TAnsistrBuilder;const
       P:THTTPStringOptions)
7026:  Proc BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7027:  Proc BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsistrBuilder;const PTHTTPStringOptions);
7028:  Proc BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B : TAnsistrBuilder; const
       P : THTTPStringOptions)
7029:  Proc BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7030:  Proc BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsistrBuilder; const P
       : THTTPStringOptions)
7031:  Proc BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsistrBuilder; const P
       : THTTPStringOptions)
7032:  Proc BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7033:  Proc BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsistrBuilder;const P:THTTPStringOptions);
7034:  Proc BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7035:  Proc BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const B:TAnsistrBuilder;const
       P:THTTPStringOptions)
7036:  Proc BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsistrBuilder; const P
       : THTTPStringOptions)
7037:  Proc BuildStrHTTPCommonHeaders( const A: THTTPCommonHeaders; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7038:  Proc BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions)
7039:  Proc BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7040:  Proc BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7041:  Proc BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7042:  Proc BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const P:THTTPStringOptions);
7043:  Proc BuildStrHTTPMethod(const A:THTTPMethod;const B:TAnsistrBuilder;const P: THTTPStringOptions)
7044:  Proc BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsistrBuilder; const P :
       THTTPStringOptions)
7045:  Proc BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsistrBuilder;const
       P:THTTPStrOptions);
7046:  Proc BuildStrHTTPRequest(const A : THTTPRequest; const B: TAnsistrBuilder; const P : THTTPStringOptions)
7047:  Proc BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B : TAnsistrBuilder;
       const P : THTTPStringOptions)
7048:  Proc BuildStrHTTPResponseStartLine(const A:THTTPResponsStartLine;const B:TAnsiStrBldr;const P
       THTTPStrOptions);
```

```
7049:  Proc BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
7050:  Proc BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsistrBuilder; const P:THTTPStringOptions);
7051:  Func HTTPContentTypeValueToStr( const A : THTTPContentType) : Ansistr
7052:  Func HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : Ansistr
7053:  Func HTTPCookieFieldValueToStr( const A : THTTPCookieField) : Ansistr
7054:  Func HTTPMethodToStr( const A : THTTPMethod) : Ansistr
7055:  Func HTTPRequestToStr( const A : THTTPRequest) : Ansistr
7056:  Func HTTPResponseToStr( const A : THTTPResponse) : Ansistr
7057:  Proc PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const Domain:Ansistr;const
      Secure:Bool; THTTPParserHeaderParseFunc','Func (const HeaderName : THTT'
7058:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) :Bool
7059:   SIRegister_THTTPParser(CL);
7060:   FindClass('TOBJECT'),'THTTPContentDecoder
7061:   THTTPContentDecoderProc', 'Proc ( const Sender : THTTPContentDecoder)
7062:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
7063:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
7064:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
7065:   THTTPContentDecoderLogEvent,Procedure(const Sender:THTTPContentDecoder;const LogMsg:str)
7066:   SIRegister_THTTPContentDecoder(CL);
7067:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
7068:   FindClass('TOBJECT'),'THTTPContentReader
7069:   THTTPContentReaderProc','Proc ( const Sender : THTTPContentReader)
7070:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:str;const
      LogLevel:Int;
7071:   SIRegister_THTTPContentReader(CL);
7072:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
7073:   FindClass('TOBJECT'),'THTTPContentWriter
7074:   THTTPContentWriterLogEvent,Procedure(const Sender:THTTPContentWriter;const LogMsg:Ansistr);
7075:   SIRegister_THTTPContentWriter(CL);
7076:  Proc SelfTestcHTTPUtils
7077: end;
7078:
7079: (*----------------------------------------------------------------------------*)
7080: Proc SIRegister_cTLSUtils(CL: TPSPascalCompiler);
7081: begin
7082:  'TLSLibraryVersion','String '1.00
7083:  'TLSError_None','LongInt'( 0);
7084:  'TLSError_InvalidBuffer','LongInt'( 1);
7085:  'TLSError_InvalidParameter','LongInt'( 2);
7086:  'TLSError_InvalidCertificate','LongInt'( 3);
7087:  'TLSError_InvalidState','LongInt'( 4);
7088:  'TLSError_DecodeError','LongInt'( 5);
7089:  'TLSError_BadProtocol','LongInt'( 6);
7090:  Func TLSErrorMessage( const TLSError : Int) :Str
7091:   SIRegister_ETLSError(CL);
7092:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
7093:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
7094:  Proc InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
7095:  Proc InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
7096:  Proc InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
7097:  Proc InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
7098:  Func IsTLSProtocolVersion( const A, B : TTLSProtocolVersion):Bool
7099:  Func IsSSL2( const A : TTLSProtocolVersion) :Boolean
7100:  Func IsSSL3( const A : TTLSProtocolVersion) :Boolean
7101:  Func IsTLS10( const A : TTLSProtocolVersion):Boolean
7102:  Func IsTLS11( const A : TTLSProtocolVersion):Boolean
7103:  Func IsTLS12( const A : TTLSProtocolVersion):Boolean
7104:  Func IsTLS10OrLater( const A : TTLSProtocolVersion) :Bool
7105:  Func IsTLS11OrLater( const A : TTLSProtocolVersion) :Bool
7106:  Func IsTLS12OrLater( const A : TTLSProtocolVersion) :Bool
7107:  Func IsFutureTLSVersion( const A : TTLSProtocolVersion) :Bool
7108:  Func IsKnownTLSVersion( const A : TTLSProtocolVersion) :Bool
7109:  Func TLSProtocolVersionToStr( const A : TTLSProtocolVersion):Str
7110:  Func TLSProtocolVersionName( const A : TTLSProtocolVersion) :Str
7111:   PTLSRandom', '^TTLSRandom // will not work
7112:  Proc InitTLSRandom( var Random : TTLSRandom)
7113:  Func TLSRandomToStr( const Random : TTLSRandom) : Ansistr
7114:  'TLSSessionIDMaxLen','LongInt'( 32);
7115:  Proc InitTLSSessionID( var SessionID : TTLSSessionID; const A : Ansistr)
7116:  Func EncodeTLSSessionID(var Buff:str;const Size:Int;const SessionID:TTLSSessionID):Int;
7117:  Func DecodeTLSSessionID(const Buff:str;const Size:Int;var SessionID:TTLSSessionID):Int;
7118:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
7119:    +'; Signature : TTLSSignatureAlgorithm; end
7120:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
7121:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
7122:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
7123:    +DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
7124:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
7125:    +HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
7126:   TTLSMacAlgorithmInfo', 'record Name:Ansistr;DigestSize:Integer; Supported:Bool; end
7127:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
7128:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
7129:   TTLSPRFAlgorithm', '( tlspaSHA256 )
7130:  Func tlsP_MD5( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7131:  Func tlsP_SHA1( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7132:  Func tlsP_SHA256( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7133:  Func tlsP_SHA512( const Secret, Seed : Ansistr; const Size : Int) : Ansistr
7134:  Func tls10PRF( const Secret, ALabel, Seed : Ansistr; const Size : Int) : Ansistr
```

```
7135:  Func tls12PRF_SHA256(const Secret,ALabel,Seed:Ansistr; const Size : Int): Ansistr
7136:  Func tls12PRF_SHA512(const Secret,ALabel,Seed :Ansistr; const Size: Int): Ansistr
7137:  Func TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
7138:  Func tls10KeyBlock(const MasterSecret,ServerRandom,ClientRandom:Ansistr;const Size:Int):AnsiStr
7139:  Func tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: Ansistr;const Size:Int):Ansistr;
7140:  Func tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: Ansistr;const Size:Int):Ansistr;
7141:  Func TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion;const MasterSecret,ServerRandom,ClientRandom:
      Ansistr;const Size:Int):Ansistr
7142:  Func tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:Ansistr) :Ansistr;
7143:  Func tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiStr):AnsiStr;
7144:  Func tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiStr): AnsiStr;
7145:  Func TLSMasterSecret(const ProtocolVersion:TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:Ansistr):Ansistr
7146:    TTLSKeys', 'record KeyBlock : Ansistr; ClientMACKey : AnsiStr'
7147:     +'ing; ServerMACKey : Ansistr; ClientEncKey : Ansistr; ServerEncKey : '
7148:     +'Ansistr; ClientIV : Ansistr; ServerIV : Ansistr; end
7149:  Proc GenerateTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const MACKeyBits,CipherKeyBits,
      IVBits:Int;const MasterSecret,ServerRandom,ClientRandom:Ansistr;var TLSKeys: TTLSKeys)
7150:  Proc GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Bool;const
      ExpandedKeyBits:Int;const ServerRandom,ClientRandom:Ansistr;var TLSKeys:TTLSKeys)
7151:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
7152:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7153:  Proc SelfTestcTLSUtils
7154: end;
7155:
7156: (*----------------------------------------------------------------------------*)
7157: Proc SIRegister_Reversi(CL: TPSPascalCompiler);
7158: begin
7159:  sPosData','record
      corner:boolean;square2x2:boolean;edge:bool;stable:Int;internal:Int;disks:Int;mx:Int;my:Int; end
7160:  // pBoard', '^tBoard // will not work
7161:  Func rCalculateData( cc : byte; cx, cy : Int) : sPosData
7162:  Func rCheckMove( color : byte; cx, cy : Int) : Int
7163:  //Func rDoStep( data : pBoard) : word
7164:  Func winExecAndWait( const sAppPath :Str; wVisible : word) :Bool
7165: end;
7166:
7167: Proc SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7168: begin
7169: Func InEditMode( ADataset : TDataset) :Bool
7170: Func CheckDataSource( ADataSource : TDataSource) :Bool;
7171: Func CheckDataSource1(ADataSource:TDataSource;const AFieldName:str;var VField:TField):bool;
7172: Func GetFieldText( AField:TField) :Str
7173: end;
7174:
7175: Proc SIRegister_SortGrid(CL: TPSPascalCompiler);
7176: begin
7177:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7178:   TMyPrintRange', '( prAll, prSelected )
7179:   TSortStyle', '(ssAutomatic,ssNormal,ssNumeric,ssNumericExtended,ssDateTime,ssTime,ssCustom )
7180:   TSortDirection', '( sdAscending, sdDescending )
7181:   TSetChecked', 'Proc ( Sender : TObject; ACol, ARow : Int; State :Bool)
7182:   TGetComboBox', 'Proc ( Sender : TObject; ACol, ARow : integ'
7183:    +er; var Strs : TStringList; var Width, Height : Int; var Sorted :Bool)
7184:   TSetComboBox', 'Proc ( Sender : TObject; ACol, ARow : Int; Str :Str)
7185:   TSetEllipsis', 'Proc ( Sender : TObject; ACol, ARow : Int)
7186:   SIRegister_TSortOptions(CL);
7187:   SIRegister_TPrintOptions(CL);
7188:   TSortedListEntry', 'record Str :Str; RowNum : Int; SortOption : TSortOptions; end
7189:   SIRegister_TSortedList(CL);
7190:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7191:   TCellBevel', 'record Style: TCellBevelStyle;UpperLeftColor:TColor;LowerRightColor:TColor;end
7192:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7193:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7194:    +Horz : TAlignment;AlignmentVert:TVertAlignment;Bevel:TCellBevel;HideText:Boolean; end
7195:   SIRegister_TFontSetting(CL);
7196:   SIRegister_TFontList(CL);
7197:   AddTypeS(TFormatDrawCellEvent', 'Proc ( Sender : TObject; Col, Row :'
7198: +Int;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7199:   TSetFilterEvent', 'Proc ( ARows : TStrings; var Accept :Bool)
7200:   TSearchEvent', 'Proc ( ARows : TStrings; var Accept :Bool)
7201:   TUpdateGridEvent', 'Proc ( Sender : TObject; ARow : Int)
7202:   TSizeChangedEvent', 'Proc ( Sender : TObject; OldColCount, OldRowCount : Int)
7203:   TBeginSortEvent', 'Proc ( Sender : TObject; var Col : Int)
7204:   TEndSortEvent', 'Proc ( Sender : TObject; Col : Int)
7205:   TGetSortStyleEvent', 'Proc ( Sender : TObject; Col : integer; var SortStyle : TSortStyle)
7206:   TCellValidateEvent', 'Proc ( Sender : TObject; ACol, ARow :'
7207:    + Int; const OldValue :Str; var NewValue :Str; var Valid :Bool)
7208:   SIRegister_TSortGrid(CL);
7209:  Func ExtendedCompare( const Str1, Str2 :Str) : Int
7210:  Func NormalCompare( const Str1, Str2 :Str) : Int
7211:  Func DateTimeCompare( const Str1, Str2 :Str) : Int
7212:  Func NumericCompare( const Str1, Str2 :Str) : Int
7213:  Func TimeCompare( const Str1, Str2 :Str) : Int
7214:  //Func Compare( Item1, Item2 : Pointer) : Int
7215: end;
7216:
7217: ************************************ Proc Register_IB(CL: TPSPascalCompiler);
```

```
7218:  Proc IBAlloc( var P, OldSize, NewSize : Int)
7219:  Proc IBError( ErrMess : TIBClientError; const Args : array of const)
7220:  Proc IBDataBaseError
7221:  Func StatusVector : PISC_STATUS
7222:  Func StatusVectorArray : PStatusVector
7223:  Func CheckStatusVector( ErrorCodes : array of ISC_STATUS) :Bool
7224:  Func StatusVectorAsText :Str
7225:  Proc SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7226:  Func GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7227:
7228: //***********************unit uPSI_BoldUtils;**********************************
7229:  Func CharCount( c : char; const s :Str) : Int
7230:  Func BoldNamesEqual( const name1, name2 :Str) :Bool
7231:  Proc BoldAppendToStrings(strings:TStrings; const aString:str;const ForceNewLine:Bool)
7232:  Func BoldSeparateStringList(strings:TStringList;const Separatr,PreString,PostString:Str)Str
7233:  Func BoldSeparatedAppend( const S1, S2 :Str; const Separator :Str) :Str
7234:  Func BoldTrim( const S :Str) :Str
7235:  Func BoldIsPrefix( const S, Prefix :Str) :Bool
7236:  Func BoldStrEqual( P1, P2 : PChar; Len : Int) :Bool
7237:  Func BoldStrAnsiEqual( P1, P2 : PChar; Len : Int) :Bool
7238:  Func BoldAnsiEqual( const S1, S2 :Str) :Bool
7239:  Func BoldStrStringEqual( const S1 :Str; P2 : PChar; Len : Int) :Bool
7240:  Func BoldCaseIndependentPos( const Substr, S :Str) : Int
7241:  //Proc EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7242:  Func CapitalisedToSpaced( Capitalised :Str) :Str
7243:  Func SpacedToCapitalised( Spaced :Str) :Str
7244:  Func BooleanToString( BoolValue :Bool) :Str
7245:  Func StringToBoolean( StrValue :Str) :Bool
7246:  Func GetUpperLimitForMultiplicity( const Multiplicity :Str) : Int
7247:  Func GetLowerLimitForMultiplicity( const Multiplicity :Str) : Int
7248:  Func StringListToVarArray( List : TStringList) : variant
7249:  Func IsLocalMachine( const Machinename : WideString) :Bool
7250:  Func GetComputerNameStr :Str
7251:  Func TimeStampComp( const Time1, Time2 : TTimeStamp) : Int
7252:  Func BoldStrToDateFmt(const S:str;const DateFormat:str;const DateSeparatorChar:char):TDateTime
7253:  Func BoldDateToStrFmt(const aDate:TDateTime;DateFormat:str;const DateSeparatorChar:char):Str;
7254:  Func BoldParseFormattedDateList(const value:str;const formats:TStrings;var Date:TDateTime):Boolean;
7255:  Func BoldParseFormattedDate(const value:str;const formats:array of string; var Date:TDateTime):Boolean;
7256:  Proc EnsureTrailing( var Str :Str; ch : char)
7257:  Func BoldDirectoryExists( const Name :Str) :Bool
7258:  Func BoldForceDirectories( Dir :Str) :Bool
7259:  Func BoldRootRegistryKey :Str
7260:  Func GetModuleFileNameAsString( IncludePath :Bool) :Str
7261:  Func BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Int
7262:  Func LogicalAnd( A, B : Int) :Bool
7263:  record TByHandleFileInformation dwFileAttributes: DWORD; '
7264:    +ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7265:    +: TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7266:    +eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow:DWORD; end
7267:  Func GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7268:  Func IsFirstInstance :Bool
7269:  Proc ActivateFirst( AString : PChar)
7270:  Proc ActivateFirstCommandLine
7271:  Func MakeAckPkt(const BlockNumber: Word):Str;
7272:  Proc SendError(UDPBase:TIdUDPBase;APeerIP:str;const APort:Int;const ErrNumb:Word;ErrStr:str;
7273:  Proc SendError(UDPClient:TIdUDPClient;const ErrNumber:Word; ErrorString:str);overload;
7274:  Proc SendError(UDPBase:TIdUDPBase;APeerIP:str;const APort:Int; E:Exception);overload;
7275:  Proc SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7276:  Func IdStrToWord(const Value:Str): Word;
7277:  Func IdWordToStr(const Value: Word): WordStr;
7278:  Func HasInstructionSet( const InstructionSet : TCPUInstructionSet) :Bool
7279:  Func CPUFeatures : TCPUFeatures
7280:
7281: Proc SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7282: begin
7283:   AddTypeS('TXRTLBitIndex', 'Int
7284:  Func XRTLSwapBits( Data :Card; Bit1Index, Bit2Index : TXRTLBitIndex) :Card
7285:  Func XRTLBitTest( Data :Card; BitIndex : TXRTLBitIndex) :Bool
7286:  Func XRTLBitSet( Data :Card; BitIndex : TXRTLBitIndex) :Card
7287:  Func XRTLBitReset( Data :Card; BitIndex : TXRTLBitIndex) :Card
7288:  Func XRTLBitComplement( Data :Card; BitIndex : TXRTLBitIndex) :Card
7289:  Func XRTLSwapHiLo16( X : Word) : Word
7290:  Func XRTLSwapHiLo32( X :Card) :Card
7291:  Func XRTLSwapHiLo64( X : Int64) : Int64
7292:  Func XRTLROL32( A, S :Card) :Card
7293:  Func XRTLROR32( A, S :Card) :Card
7294:  Func XRTLROL16( A : Word; S :Card) : Word
7295:  Func XRTLROR16( A : Word; S :Card) : Word
7296:  Func XRTLROL8( A : Byte; S :Card) : Byte
7297:  Func XRTLROR8( A : Byte; S :Card) : Byte
7298:  //Proc XRTLXorBlock( I1, I2, O1 : PByteArray; Len : Int)
7299:  //Proc XRTLIncBlock( P : PByteArray; Len : Int)
7300:  Proc XRTLUMul64( const A, B : Int; var MulL, MulH : Int)
7301:  Func XRTLPopulation( A :Card):Card
7302: end;
7303: Func XRTLURLDecode( const ASrc : WideString) : WideString
7304: Func XRTLURLEncode( const ASrc : WideString) : WideString
7305: Func XRTLURINormalize( const AURI : WideString) : WideString
7306: Proc XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword:WideString)
```

```
7307: Func XRTLExtractLongPathName(APath:Str):Str;
7308:
7309: Proc SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7310: begin
7311:   Int8', 'ShortInt
7312:   AddTypeS('Int16', 'SmallInt
7313:   Int32', 'LongInt
7314:   UInt8', 'Byte
7315:   UInt16', 'Word
7316:   UInt32', 'LongWord
7317:   UInt64', 'Int64
7318:   Word8', 'UInt8
7319:   Word16', 'UInt16
7320:   Word32', 'UInt32
7321:   Word64', 'UInt64
7322:   LargeInt', 'Int64
7323:   NativeInt', 'Int
7324:   AddTypeS('NativeUInt', 'Cardinal
7325:   Const('BitsPerByte','LongInt'( 8);
7326:   Const('BitsPerWord','LongInt'( 16);
7327:   Const('BitsPerLongWord','LongInt'( 32);
7328:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7329:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7330:  Func MinI( const A,B Int) : Int
7331:  Func MaxI( const A,B:Int) : Int
7332:  Func MinC( const A,B:Card) :Card
7333:  Func MaxC( const A,B:Card) :Card
7334:  Func SumClipI( const A, I : Int) : Int
7335:  Func SumClipC( const A :Card; const I : Int) :Card
7336:  Func InByteRange( const A : Int64) :Bool
7337:  Func InWordRange( const A : Int64) :Bool
7338:  Func InLongWordRange( const A : Int64): Bool
7339:  Func InShortIntRange( const A : Int64): Bool
7340:  Func InSmallIntRange( const A : Int64 : Bool
7341:  Func InLongIntRange( const A : Int64) : Bool
7342:   AddTypeS('Bool8', 'ByteBool
7343:   AddTypeS('Bool16', 'WordBool
7344:   AddTypeS('Bool32', 'LongBool
7345:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7346:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7347:  Func ReverseCompareResult( const C : TCompareResult) : TCompareResult
7348:  Const('MinSingle','Single').setExtended( 1.5E-45);
7349:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7350:  Const('MinDouble','Double').setExtended( 5.0E-324);
7351:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7352:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7353:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7354:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7355:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7356:  Func MinF( const A, B : Float) : Float
7357:  Func MaxF( const A, B : Float) : Float
7358:  Func ClipF( const Value : Float; const Low, High : Float) : Float
7359:  Func InSingleRange( const A : Float) :Bool
7360:  Func InDoubleRange( const A : Float) :Bool
7361:  Func InCurrencyRange( const A : Float) :Bool;
7362:  Func InCurrencyRange1( const A : Int64) :Bool;
7363:  Func FloatExponentBase2( const A : Extended; var Exponent : Int) :Bool
7364:  Func FloatExponentBase10( const A : Extended; var Exponent : Int):Bool
7365:  Func FloatIsInfinity( const A : Extended) :Bool
7366:  Func FloatIsNaN( const A : Extended) :Bool
7367:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7368:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7369:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7370:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7371:  Func FloatZero( const A : Float; const CompareDelta : Float) :Bool
7372:  Func FloatOne( const A : Float; const CompareDelta : Float) :Bool
7373:  Func FloatsEqual( const A, B : Float; const CompareDelta : Float) :Bool
7374:  Func FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7375:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7376:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7377:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7378:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7379:  Func ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) :Bool
7380:  Func ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7381:  Func cClearBit( const Value, BitIndex : LongWord) : LongWord
7382:  Func cSetBit( const Value, BitIndex : LongWord) : LongWord
7383:  Func cIsBitSet( const Value, BitIndex : LongWord) :Bool
7384:  Func cToggleBit( const Value, BitIndex : LongWord) : LongWord
7385:  Func cIsHighBitSet( const Value : LongWord) :Bool
7386:  Func SetBitScanForward( const Value : LongWord) : Int;
7387:  Func SetBitScanForward1( const Value, BitIndex : LongWord) : Int;
7388:  Func SetBitScanReverse( const Value : LongWord) : Int;
7389:  Func SetBitScanReverse1( const Value, BitIndex : LongWord) : Int;
7390:  Func ClearBitScanForward( const Value : LongWord) : Int;
7391:  Func ClearBitScanForward1( const Value, BitIndex : LongWord) : Int;
7392:  Func ClearBitScanReverse( const Value : LongWord) : Int;
7393:  Func ClearBitScanReverse1( const Value, BitIndex : LongWord) : Int;
7394:  Func cReverseBits( const Value : LongWord) : LongWord;
7395:  Func cReverseBits1( const Value : LongWord; const BitCount : Int) : LongWord;
```

```
Func cSwapEndian( const Value : LongWord) : LongWord
Func cTwosComplement( const Value : LongWord) : LongWord
Func RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
Func RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
Func RotateRightBits16( const Value : Word; const Bits : Byte) : Word
Func RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
Func cBitCount( const Value : LongWord) : LongWord
Func cIsPowerOfTwo( const Value : LongWord) :Bool
Func LowBitMask( const HighBitIndex : LongWord) : LongWord
Func HighBitMask( const LowBitIndex : LongWord) : LongWord
Func RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
Func SetBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex: LongWord): LongWord
Func ClearBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord): LongWord
Func ToggleBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):LongWord
Func IsBitRangeSet(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):Bool
Func IsBitRangeClear(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):Boolean
//  AddTypeS('CharSet', 'set of AnsiChar
 AddTypeS('CharSet', 'set of Char       //!!!
 AddTypeS('AnsiCharSet', 'TCharSet
 AddTypeS('ByteSet', 'set of Byte
 AddTypeS('AnsiChar', 'Char
  // Func AsCharSet( const C : array of AnsiChar) : CharSet
Func AsByteSet( const C : array of Byte) : ByteSet
Proc ComplementChar( var C : CharSet; const Ch : Char)
Proc ClearCharSet( var C : CharSet)
Proc FillCharSet( var C : CharSet)
Proc FillCharSearchRec;  // with 0
Proc ComplementCharSet( var C : CharSet)
Proc AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
Proc Union( var DestSet : CharSet; const SourceSet : CharSet)
Proc Difference( var DestSet : CharSet; const SourceSet : CharSet)
Proc Intersection( var DestSet : CharSet; const SourceSet : CharSet)
Proc XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
Func IsSubSet( const A, B : CharSet) :Bool
//Func IsEqual( const A, B : CharSet) :Bool
Func IsEqualCharset( const A, B : CharSet) :Bool
Func IsEqual( const D1, D2 : TDateTime) :Bool;
Func IsEmpty( const C : CharSet) :Bool
Func IsEmptyCharset( const C : CharSet) :Bool
Func IsComplete( const C : CharSet) :Bool
Func cCharCount( const C : CharSet) : Int
Proc ConvertCaseInsensitive( var C : CharSet)
Func CaseInsensitiveCharSet( const C : CharSet) : CharSet
Func IntRangeLength( const Low, High : Int) : Int64
Func IntRangeAdjacent( const Low1, High1, Low2, High2 : Int) :Bool
Func IntRangeOverlap( const Low1, High1, Low2, High2 : Int) :Bool
Func IntRangeHasElement( const Low, High, Element : Int) :Bool
Func IntRangeIncludeElement( var Low, High : Int; const Element : Int) :Bool
Func IntRangeIncludeElementRange(var Low,High:Int;const LowElement,HighElement:Int):Boolean
Func CardRangeLength( const Low, High :Card) : Int64
Func CardRangeAdjacent( const Low1, High1, Low2, High2 :Card) :Bool
Func CardRangeOverlap( const Low1, High1, Low2, High2 :Card) :Bool
Func CardRangeHasElement( const Low, High, Element :Card) :Bool
Func CardRangeIncludeElement( var Low, High :Card; const Element :Card) :Bool
Func CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElemnt:Card):Bool;
 AddTypeS('UnicodeChar', 'WideChar
Func Compare( const I1, I2 :Bool) : TCompareResult;
Func Compare1( const I1, I2 : Int) : TCompareResult;
Func Compare2( const I1, I2 : Int64) : TCompareResult;
Func Compare3( const I1, I2 : Extended) :TCompareResult;
Func CompareA( const I1, I2 : Ansistr): TCompareResult
Func CompareW( const I1, I2 : WideString): TCompareResult
Func cSgn( const A : LongInt) : Int;
Func cSgn1( const A : Int64) : Int;
Func cSgn2( const A : Extended) : Int;
AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
Func AnsiCharToInt( const A : AnsiChar) : Int
Func WideCharToInt( const A : WideChar) : Int
Func CharToInt( const A : Char) : Int
Func IntToAnsiChar( const A : Int) : AnsiChar
Func IntToWideChar( const A : Int) : WideChar
Func IntToChar( const A : Int) : Char
Func IsHexAnsiChar( const Ch : AnsiChar) :Bool
Func IsHexWideChar( const Ch : WideChar) :Bool
Func IsHexChar( const Ch : Char) :Bool
Func HexAnsiCharToInt( const A : AnsiChar) : Int
Func HexWideCharToInt( const A : WideChar) : Int
Func HexCharToInt( const A : Char) : Int
Func IntToUpperHexAnsiChar( const A : Int) : AnsiChar
Func IntToUpperHexWideChar( const A : Int) : WideChar
Func IntToUpperHexChar( const A : Int) : Char
Func IntToLowerHexAnsiChar( const A : Int) : AnsiChar
Func IntToLowerHexWideChar( const A : Int) : WideChar
Func IntToLowerHexChar( const A : Int) : Char
Func IntToStringA( const A : Int64) : Ansistr
Func IntToStringW( const A : Int64) : WideString
Func IntToString( const A : Int64) :Str
Func UIntToStringA( const A : NativeUInt) : Ansistr
Func UIntToStringW( const A : NativeUInt) : WideString
```

```
7485:  Func UIntToString( const A : NativeUInt) :Str
7486:  Func LongWordToStrA( const A : LongWord; const Digits : Int) : Ansistr
7487:  Func LongWordToStrW( const A : LongWord; const Digits : Int) : WideString
7488:  Func LongWordToStrU( const A : LongWord; const Digits : Int) : UnicodeString
7489:  Func LongWordToStr( const A : LongWord; const Digits : Int) :Str
7490:  Func LongWordToHexA(const A:LongWord;const Digits:Int;const UpperCase:Bool):Ansistr;
7491:  Func LongWordToHexW(const A:LongWord;const Digits:Int;const UpperCase:Bool):WideString;
7492:  Func LongWordToHex( const A : LongWord; const Digits : Int;const UpperCase:Boolean):str
7493:  Func LongWordToOctA( const A : LongWord; const Digits : Int) : Ansistr
7494:  Func LongWordToOctW( const A : LongWord; const Digits : Int) : WideString
7495:  Func LongWordToOct( const A : LongWord; const Digits : Int) :Str
7496:  Func LongWordToBinA( const A : LongWord; const Digits : Int) : Ansistr
7497:  Func LongWordToBinW( const A : LongWord; const Digits : Int) : WideString
7498:  Func LongWordToBin( const A : LongWord; const Digits : Int) :Str
7499:  Func TryStringToInt64A( const S : Ansistr; out A : Int64) :Bool
7500:  Func TryStringToInt64W( const S : WideString; out A : Int64) :Bool
7501:  Func TryStringToInt64( const S :Str; out A : Int64) :Bool
7502:  Func StringToInt64DefA( const S : Ansistr; const Default : Int64): Int64
7503:  Func StringToInt64DefW( const S : WideString; const Default : Int64): Int64
7504:  Func StringToInt64Def( const S :Str; const Default : Int64) : Int64
7505:  Func StringToInt64A( const S : Ansistr) : Int64
7506:  Func StringToInt64W( const S : WideString) : Int64
7507:  Func StringToInt64( const S :Str) : Int64
7508:  Func TryStringToIntA( const S : Ansistr; out A : Int) :Bool
7509:  Func TryStringToIntW( const S : WideString; out A : Int) :Bool
7510:  Func TryStringToInt( const S :Str; out A : Int) :Bool
7511:  Func StringToIntDefA( const S : Ansistr; const Default : Int) : Int
7512:  Func StringToIntDefW( const S : WideString; const Default : Int) : Int
7513:  Func StringToIntDef( const S :Str; const Default : Int) : Int
7514:  Func StringToIntA( const S : Ansistr) : Int
7515:  Func StringToIntW( const S : WideString) : Int
7516:  Func StringToInt( const S :Str) : Int
7517:  Func TryStringToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7518:  Func TryStringToLongWordW( const S : WideString; out A : LongWord) :Bool
7519:  Func TryStringToLongWord( const S :Str; out A : LongWord) :Bool
7520:  Func StringToLongWordA( const S : Ansistr) : LongWord
7521:  Func StringToLongWordW( const S : WideString) : LongWord
7522:  Func StringToLongWord( const S :Str) : LongWord
7523:  Func HexToUIntA( const S : Ansistr) : NativeUInt
7524:  Func HexToUIntW( const S : WideString) : NativeUInt
7525:  Func HexToUInt( const S :Str) : NativeUInt
7526:  Func TryHexToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7527:  Func TryHexToLongWordW( const S : WideString; out A : LongWord) :Bool
7528:  Func TryHexToLongWord( const S :Str; out A : LongWord) :Bool
7529:  Func HexToLongWordA( const S : Ansistr) : LongWord
7530:  Func HexToLongWordW( const S : WideString) : LongWord
7531:  Func HexToLongWord( const S :Str) : LongWord
7532:  Func TryOctToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7533:  Func TryOctToLongWordW( const S : WideString; out A : LongWord) :Bool
7534:  Func TryOctToLongWord( const S :Str; out A : LongWord) :Bool
7535:  Func OctToLongWordA( const S : Ansistr) : LongWord
7536:  Func OctToLongWordW( const S : WideString) : LongWord
7537:  Func OctToLongWord( const S :Str) : LongWord
7538:  Func TryBinToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7539:  Func TryBinToLongWordW( const S : WideString; out A : LongWord) :Bool
7540:  Func TryBinToLongWord( const S :Str; out A : LongWord) :Bool
7541:  Func BinToLongWordA( const S : Ansistr) : LongWord
7542:  Func BinToLongWordW( const S : WideString) : LongWord
7543:  Func BinToLongWord( const S :Str) : LongWord
7544:  Func FloatToStringA( const A : Extended) : Ansistr
7545:  Func FloatToStringW( const A : Extended) : WideString
7546:  Func FloatToString( const A : Extended) :Str
7547:  Func FloatToStringA( const A : Extended) :Str
7548:  Func TryStringToFloatA( const A : Ansistr; out B : Extended) :Bool
7549:  Func TryStringToFloatW( const A : WideString; out B : Extended) :Bool
7550:  Func TryStringToFloat( const A :Str; out B : Extended) :Bool
7551:  Func StringToFloatA( const A : Ansistr) : Extended
7552:  Func StringToFloatW( const A : WideString) : Extended
7553:  Func StringToFloat( const A :Str) : Extended
7554:  Func StringToFloatDefA( const A : Ansistr; const Default : Extended) : Extended
7555:  Func StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7556:  Func StringToFloatDef( const A :Str; const Default : Extended) : Extended
7557:  Func EncodeBase64(const S,Alphabet:Ansistr;const Pad:Bool;const PadMultiple:Int;const
       PadChar:AnsiChar):Ansistr
7558:  Func DecodeBase64( const S, Alphabet:Ansistr;const PadSet:CharSet):Ansistr
7559:  unit uPSI_cFundamentUtils;
7560:  Const('b64_MIMEBase64',Str).String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7561:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7562:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7563:  Const('CCHARSET','Stringb64_XXEncode);
7564:  Const('CHEXSET','String'0123456789ABCDEF
7565:  Const('HEXDIGITS','String'0123456789ABCDEF
7566:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7567:  Const('DIGISET','String'0123456789
7568:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7569:  Const('DIGISET2','TCharset').SetSet('0123456789'
7570:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7571:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF;
7572:  Const('NUMBERSET','TCharset').SetSet('0123456789;
```

```
7573:  Const('NUMBERS','String'0123456789;
7574:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ;
7575:  Const('NUMBLETTS','String').SetString('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ;
7576:  Const('NUMBLETTSET','TCharset').SetSet('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ;
7577:  Const('ALPHANUMSET','TCharset').SetSet('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz);
7578:  tantN('ALPHANUMSET','TCharset').SetSet('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
7579:  Const('ALPHANUMSTR','String').SetString('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
7580:
7581:  Func CharSetToStr( const C : CharSet) : Ansistr
7582:  Func StrToCharSet( const S : Ansistr) : CharSet
7583:  Func MIMEBase64Decode( const S : Ansistr) : Ansistr
7584:  Func MIMEBase64Encode( const S : Ansistr) : Ansistr
7585:  Func UUDecode( const S: Ansistr) : Ansistr
7586:  Func XXDecode( const S  Ansistr) : Ansistr
7587:  Func BytesToHex( const P : array of Byte; const UpperCase :Bool) : Ansistr
7588:  Func InterfaceToStrA( const I : IInterface) : Ansistr
7589:  Func InterfaceToStrW( const I : IInterface) : WideString
7590:  Func InterfaceToStr( const I : IInterface) :Str
7591:  Func ObjectClassName( const O : TObject) :Str
7592:  Func ClassClassName( const C : TClass) :Str
7593:  Func ObjectToStr( const O : TObject) :Str
7594:  Func ObjectToString( const O : TObject) :Str
7595:  Func CharSetToStr( const C : CharSet) : Ansistr
7596:  Func StrToCharSet( const S : Ansistr) : CharSet
7597:  Func HashStrA(const S : Ansistr; const Index : Int; const Count: Int;const AsciiCaseSensitive :Bool;
      const Slots : LongWord) : LongWord
7598:  Func HashStrW(const S:WideString;const Index:Int;const Count:Int;const AsciiCaseSensitive:Bool;const
      Slots:LongWord):LongWord
7599:  Func HashStr(const S:str;const Index:Int;const Count:Int;const AsciiCaseSensitive:Bool;const Slots:
      LongWord):LongWord
7600:  Func HashInt( const I : Int; const Slots : LongWord) : LongWord
7601:  Func HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7602:  Const('Bytes1KB','LongInt'( 1024);
7603:   SIRegister_IInterface(CL);
7604:  Proc SelfTestCFundamentUtils
7605:
7606: Func CreateSchedule : IJclSchedule
7607: Func NullStamp : TTimeStamp
7608: Func CompareTimeStamps(const Stamp1, Stamp2 : TTimeStamp) : Int64
7609: Func EqualTimeStamps(const Stamp1, Stamp2 : TTimeStamp) :Bool
7610: Func IsNullTimeStamp(const Stamp : TTimeStamp) :Bool
7611:
7612: Proc SIRegister_uwinplot(CL: TPSPascalCompiler);
7613: begin
7614:  AddTypeS(TFunc', 'function(X : Float) : Float;
7615: Func InitGraphics( Width, Height : Int) :Bool
7616: Proc SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Int; GraphBorder :Bool)
7617: Proc SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7618: Proc SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7619: Proc GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7620: Proc GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7621: Proc SetGraphTitle( Title :Str)
7622: Proc SetOxTitle( Title :Str)
7623: Proc SetOyTitle( Title :Str)
7624: Func GetGraphTitle :Str
7625: Func GetOxTitle :Str
7626: Func GetOyTitle :Str
7627: Proc PlotOxAxis( Canvas : TCanvas)
7628: Proc PlotOyAxis( Canvas : TCanvas)
7629: Proc PlotGrid( Canvas : TCanvas; Grid : TGrid)
7630: Proc WriteGraphTitle( Canvas : TCanvas)
7631: Func SetMaxCurv( NCurv : Byte) :Bool
7632: Proc SetPointParam( CurvIndex, Symbol, Size : Int; Color : TColor)
7633: Proc SetLineParam(CurvIndex: Int; Style : TPenStyle; Width : Int; Color : TColor)
7634: Proc SetCurvLegend( CurvIndex : Int; Legend :Str)
7635: Proc SetCurvStep( CurvIndex, Step : Int)
7636: Func GetMaxCurv : Byte
7637: Proc GetPointParam( CurvIndex : Int; var Symbol, Size : Int; var Color : TColor)
7638: Proc GetLineParam(CurvIndex:Int;var Style:TPenStyle;var Width:Int;var Color:TColor);
7639: Func GetCurvLegend( CurvIndex : Int) :Str
7640: Func GetCurvStep( CurvIndex : Int) : Int
7641: Proc PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Int)
7642: Proc PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Int)
7643: Proc PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Int)
7644: Proc PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Int)
7645: Proc WriteLegend( Canvas : TCanvas; NCurv : Int; ShowPoints, ShowLines :Bool)
7646: Proc ConRec( Canvas : TCanvas; Nx, Ny, Nc : Int; X, Y, Z : TVector; F : TMatrix)
7647: Func Xpixel(X:Float) : Int
7648: Func Ypixel(Y:Float) : Int
7649: Func Xuser( X:Int) : Float
7650: Func Yuser( Y:Int) : Float
7651: end;
7652:
7653: Proc FFT( NumSamples : Int; InArray, OutArray : TCompVector)
7654: Proc IFFT( NumSamples : Int; InArray, OutArray : TCompVector)
7655: Proc FFT_Int( NumSamples : Int; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7656: Proc FFT_Int_Cleanup
7657: Proc CalcFrequency(NumSamples,FrequencyIndex: Int;InArray: TCompVector;var FT : Complex)
7658: //unit uPSI_JclStreams;
```

```
7659: Func StreamSeek(Stream: TStream; const Offset: Int64; const Origin : TSeekOrigin) : Int64
7660: Func StreamCopy( Source : TStream; Dest : TStream; BufferSize : Int) : Int64
7661: Func CompareStreams( A, B : TStream; BufferSize : Int) :Bool
7662: Func JCompareFiles( const FileA, FileB : TFileName; BufferSize : Int) :Bool
7663:
7664: Proc SIRegister_FmxUtils(CL: TPSPascalCompiler);
7665: begin
7666:  FindClass('TOBJECT'),'EInvalidDest
7667:  FindClass('TOBJECT'),'EFCantMove
7668:  Proc fmxCopyFile( const FileName, DestName :Str)
7669:  Proc fmxMoveFile( const FileName, DestName :Str)
7670:  Func fmxGetFileSize( const FileName :Str) : LongInt
7671:  Func fmxFileDateTime( const FileName :Str) : TDateTime
7672:  Func fmxHasAttr( const FileName :Str; Attr : Word) :Bool
7673:  Func fmxExecuteFile( const FileName, Params, DefaultDir:Str; ShowCmd : Int):THandle;
7674: end;
7675:
7676: Proc SIRegister_FindFileIter(CL: TPSPascalCompiler);
7677: begin
7678:  SIRegister_IFindFileIterator(CL);
7679:  Func CreateFindFile(const Path:str; IncludeAttr:Int;out iffi:IFindFileIterator):Bool;
7680: end;
7681:
7682: Proc SIRegister_PCharUtils(CL: TPSPascalCompiler);
7683: begin
7684:  Func SkipWhite( cp : PChar) : PChar
7685:  Func ReadStringDoubleQuotedMaybe( cp : PChar; var AStr :Str) : PChar
7686:  Func ReadStringSingleQuotedMaybe( cp : PChar; var AStr :Str) : PChar
7687:  Func ReadIdent( cp : PChar; var ident :Str) : PChar
7688: end;
7689:
7690: Proc SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7691: begin
7692:   SIRegister_TStringHashMapTraits(CL);
7693:  Func CaseSensitiveTraits : TStringHashMapTraits
7694:  Func CaseInsensitiveTraits : TStringHashMapTraits
7695:   THashNode','record Str:str;Ptr:Pointer; Left:PHashNode;Right:PHashNode;end
7696:   //PHashArray', '^THashArray // will not work
7697:   SIRegister_TStringHashMap(CL);
7698:  THashValue', 'Cardinal
7699: Func StrHash( const s :Str) : THashValue
7700: Func TextHash( const s :Str) : THashValue
7701: Func DataHash( var AValue, ASize :Card) : THashValue
7702: Func Iterate_FreeObjects(AUserData:Pointer;const AStr:str; var AData:Pointer):Boolean
7703: Func Iterate_Dispose(AUserData: Pointer;const AStr:str; var AData: Pointer):Bool
7704: Func Iterate_FreeMem(AUserData: Pointer;const AStr:str; var AData  Pointer):Bool
7705:  SIRegister_TCaseSensitiveTraits(CL);
7706:  SIRegister_TCaseInsensitiveTraits(CL);
7707: //*************************************************************unit uPSI_umath;
7708:  Func uExpo( X : Float) : Float
7709:  Func uExp2( X : Float) : Float
7710:  Func uExp10( X : Float) : Float
7711:  Func uLog( X : Float) : Float
7712:  Func uLog2( X : Float) : Float
7713:  Func uLog10( X : Float) : Float
7714:  Func uLogA( X, A : Float) : Float
7715:  Func uIntPower( X : Float; N : Int): Float
7716:  Func uPower( X, Y : Float) : Float
7717:  Func SgnGamma( X : Float) : Int
7718:  Func Stirling( X : Float) : Float
7719:  Func StirLog( X : Float) : Float
7720:  Func Gamma( X : Float) : Float
7721:  Func LnGamma( X : Float) : Float
7722:  Func DiGamma( X : Float) : Float
7723:  Func TriGamma( X : Float) : Float
7724:  Func IGamma( X : Float) : Float
7725:  Func JGamma( X : Float) : Float
7726:  Func InvGamma( X : Float) : Float
7727:  Func Erf( X : Float) : Float
7728:  Func Erfc( X : Float) : Float
7729: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float;
7730: { Correlation coefficient between samples X and Y }
7731: Func DBeta(A, B, X : Float) : Float;
7732: { Density of Beta distribution with parameters A and B }
7733: Func LambertW( X : Float; UBranch, Offset :Bool) : Float
7734: Func Beta(X, Y : Float) : Float
7735: Func Binomial( N, K : Int) : Float
7736: Func PBinom( N : Int; P : Float; K : Int) : Float
7737: Proc Cholesky( A, L : TMatrix; Lb, Ub : Int)
7738: Proc LU_Decomp( A : TMatrix; Lb, Ub : Int)
7739: Proc LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Int; X : TVector)
7740: Func DNorm( X : Float) : Float
7741: Func DGamma(A, B, X : Float) : Float;
7742: { Density of Gamma distribution with parameters A and B }
7743: Func DKhi2(Nu : Int; X : Float) : Float;
7744: { Density of Khi-2 distribution with Nu d.o.f. }
7745: Func DStudent(Nu : Int; X : Float) : Float;
7746: { Density of Student distribution with Nu d.o.f. }
7747: Func DSnedecor(Nu1, Nu2 : Int; X : Float) : Float;
```

```
7748: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7749: Func IBeta(A, B, X : Float) : Float;
7750: { Incomplete Beta function}
7751: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float;
7752:
7753: Proc SIRegister_unlfit(CL: TPSPascalCompiler);
7754: begin Proc SetOptAlgo( Algo : TOptAlgo)
7755: Proc SetOptAlgo(Algo : TOptAlgo);
7756: { ----------------------------------------------------------------
7757:   Sets the optimization algorithm according to Algo, which must be
7758:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7759:  Func GetOptAlgo : TOptAlgo
7760:  Proc SetMaxParam( N : Byte)
7761:  Func GetMaxParam : Byte
7762:  Proc SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7763:  Proc GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7764:  Func NullParam( B : TVector; Lb, Ub : Int) :Bool
7765:  Proc NLFit(RegFunc:TRegFunc;DerivProc:TDerivProc;X,Y:TVector;Lb Ub:Int;MaxIter:Int;Tol:
      Float;B:TVector;FirstPar,LastPar:Int;V:TMatrix)
7766:  Proc WNLFit(RegFunc:TRegFunc;DerivProc:TDerivProc;X,Y,S:TVector;Lb,Ub:Int;MaxIter:Int;Tol
      :Float;B:TVector;FirstPar,LastPar:Int;V:TMatrix)
7767:  Proc SetMCFile( FileName :Str)
7768:  Proc SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Int;B:TVector;FirstPar,LastPar:Int;V:TMatrix;
7769:  Proc WSimFit(RegFunc:TRegFunc;X,Y,S:TVector;Lb,Ub:Int;B:TVector;FirstPar,LastPar:Int;V:TMatrix);
7770: end;
7771:
7772: (*----------------------------------------------------------------------------*)
7773: Proc SIRegister_usimplex(CL: TPSPascalCompiler);
7774: begin
7775:  Proc SaveSimplex( FileName :Str)
7776:  Proc Simplex(Func:TFuncNVar;X:TVector;Lb,Ub:Int; MaxIter:Int;Tol:Float;var F_min:Float);
7777: end;
7778: (*----------------------------------------------------------------------------*)
7779: Proc SIRegister_uregtest(CL: TPSPascalCompiler);
7780: begin
7781:  Proc RegTest(Y,Ycalc:TVector;LbY,UbY:Int;V:TMatrix;LbV,UbV:Int;var Test:TRegTest)
7782:  Proc WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Int;V:TMatrix;LbV,UbV:Int;var Test:TRegTest);
7783: end;
7784:
7785: Proc SIRegister_ustrings(CL: TPSPascalCompiler);
7786: begin
7787:  Func LTrim( S:Str) :Str
7788:  Func RTrim( S:Str) :Str
7789:  Func uTrim( S:Str) :Str
7790:  Func StrChar(N : Byte; C : Char) :Str
7791:  Func RFill( S:Str; L : Byte) :Str
7792:  Func LFill( S:Str; L : Byte) :Str
7793:  Func CFill( S:Str; L : Byte) :Str
7794:  Func Replace(S :Str; C1, C2 : Char) :Str
7795:  Func Extract(S :Str; var Index : Byte; Delim : Char) :Str
7796:  Proc Parse( S :Str; Delim : Char; Field : TStrVector; var N : Byte)
7797:  Proc SetFormat( NumLength, MaxDec : Int; FloatPoint, NSZero :Bool)
7798:  Func FloatStr( X : Float) :Str
7799:  Func IntStr( N : LongInt) :Str
7800:  Func uCompStr( Z : Complex) :Str
7801: end;
7802:
7803: Proc SIRegister_uhyper(CL: TPSPascalCompiler);
7804: begin
7805:  Func uSinh( X : Float) : Float
7806:  Func uCosh( X : Float) : Float
7807:  Func uTanh( X : Float) : Float
7808:  Func uArcSinh( X : Float) : Float
7809:  Func uArcCosh( X : Float) : Float
7810:  Func ArcTanh( X : Float) : Float
7811:  Proc SinhCosh( X : Float; var SinhX, CoshX : Float)
7812: end;
7813:
7814: Proc SIRegister_urandom(CL: TPSPascalCompiler);
7815: begin
7816: type RNG_Type =
7817:   (RNG_MWC,       { Multiply-With-Carry }
7818:    RNG_MT,        { Mersenne Twister }
7819:    RNG_UVAG);     { Universal Virtual Array Generator }
7820:  Proc SetRNG( RNG : RNG_Type)
7821:  Proc InitGen( Seed : RNG_IntType)
7822:  Proc SRand( Seed : RNG_IntType)
7823:  Func IRanGen : RNG_IntType
7824:  Func IRanGen31 : RNG_IntType
7825:  Func RanGen1 : Float
7826:  Func RanGen2 : Float
7827:  Func RanGen3 : Float
7828:  Func RanGen53 : Float
7829: end;
7830:
7831: //  Optimization by Simulated Annealing
7832: Proc SIRegister_usimann(CL: TPSPascalCompiler);
7833: begin
7834:  Proc InitSAParams( NT, NS, NCycles : Int; RT : Float)
```

```
7835:  Proc SA_CreateLogFile( FileName :Str)
7836:  Proc SimAnn(Func: TFuncNVar; X,Xmin,Xmax: TVector; Lb, Ub : Int; var F_min : Float);
7837: end;
7838:
7839: Proc SIRegister_uranuvag(CL: TPSPascalCompiler);
7840: begin
7841:  Proc InitUVAGbyString( KeyPhrase :Str)
7842:  Proc InitUVAG( Seed : RNG_IntType)
7843:  Func IRanUVAG : RNG_IntType
7844: end;
7845:
7846: Proc SIRegister_ugenalg(CL: TPSPascalCompiler);
7847: begin
7848:  Proc InitGAParams( NP, NG : Int; SR, MR, HR : Float)
7849:  Proc GA_CreateLogFile( LogFileName :Str)
7850:  Proc GenAlg(Func: TFuncNVar; X,Xmin,Xmax : TVector; Lb, Ub : Int; var F_min : Float);
7851: end;
7852:
7853:  TVector', 'array of Float
7854: Proc SIRegister_uqsort(CL: TPSPascalCompiler);
7855: begin
7856:  Proc QSort( X : TVector; Lb, Ub : Int)
7857:  Proc DQSort( X : TVector; Lb, Ub : Int)
7858: end;
7859:
7860: Proc SIRegister_uinterv(CL: TPSPascalCompiler);
7861: begin
7862:  Proc Interval( X1, X2 : Float; MinDiv, MaxDiv : Int; var Min, Max, Step : Float)
7863:  Proc AutoScale(X: TVector; Lb, Ub : Int; Scale : TScale; var XMin, XMax, XStep:Float)
7864: end;
7865:
7866: Proc SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7867: begin
7868:   FT_Result', 'Int
7869:   //TDWordptr', '^DWord // will not work
7870:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7871:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7872:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7873:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7874:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7875:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7876:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7877:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7878:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7879:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7880:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs :  '
7881:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7882:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7883:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7884:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7885:    yte; end
7886: end;
7887:
7888: //************************************** PaintFX***************************
7889: Proc SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7890: begin
7891: //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7892: with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7893:   Proc Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Int)
7894:   Proc Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Int)
7895:   Proc Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7896:   Proc Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount: Single)
7897:   Proc ExtractColor( const Dst : TBitmap; AColor : TColor)
7898:   Proc ExcludeColor( const Dst : TBitmap; AColor : TColor)
7899:   Proc Turn( Src, Dst : TBitmap)
7900:   Proc TurnRight( Src, Dst : TBitmap)
7901:   Proc HeightMap( const Dst : TBitmap; Amount : Int)
7902:   Proc TexturizeTile( const Dst : TBitmap; Amount : Int)
7903:   Proc TexturizeOverlap( const Dst : TBitmap; Amount : Int)
7904:   Proc RippleRandom( const Dst : TBitmap; Amount : Int)
7905:   Proc RippleTooth( const Dst : TBitmap; Amount : Int)
7906:   Proc RippleTriangle( const Dst : TBitmap; Amount : Int)
7907:   Proc Triangles( const Dst : TBitmap; Amount : Int)
7908:   Proc DrawMandelJulia(const Dst:TBitmap;x0,y0,x1,y1:Single;Niter:Int;Mandel:Bool)
7909:   Proc FilterXBlue( const Dst : TBitmap; Min, Max : Int)
7910:   Proc FilterXGreen( const Dst : TBitmap; Min, Max : Int)
7911:   Proc FilterXRed( const Dst : TBitmap; Min, Max : Int)
7912:   Proc FilterBlue( const Dst : TBitmap; Min, Max : Int)
7913:   Proc FilterGreen( const Dst : TBitmap; Min, Max : Int)
7914:   Proc FilterRed( const Dst : TBitmap; Min, Max : Int)
7915:   Proc Emboss( var Bmp : TBitmap)
7916:   Proc Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7917:   Proc Shake( Src, Dst : TBitmap; Factor : Single)
7918:   Proc ShakeDown( Src, Dst : TBitmap; Factor : Single)
7919:   Proc KeepBlue( const Dst : TBitmap; Factor : Single)
7920:   Proc KeepGreen( const Dst : TBitmap; Factor : Single)
7921:   Proc KeepRed( const Dst : TBitmap; Factor : Single)
7922:   Proc Mandelbrot( const Dst : TBitmap; Factor : Int)
7923:   Proc MaskMandelbrot( const Dst : TBitmap; Factor : Int)
```

```
7924:   Proc FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7925:   Proc QuartoOpaque( Src, Dst : TBitmap)
7926:   Proc SemiOpaque( Src, Dst : TBitmap)
7927:   Proc ShadowDownLeft( const Dst : TBitmap)
7928:   Proc ShadowDownRight( const Dst : TBitmap)
7929:   Proc ShadowUpLeft( const Dst : TBitmap)
7930:   Proc ShadowUpRight( const Dst : TBitmap)
7931:   Proc Darkness( const Dst : TBitmap; Amount : Int)
7932:   Proc Trace( const Dst : TBitmap; Intensity : Int)
7933:   Proc FlipRight( const Dst : TBitmap)
7934:   Proc FlipDown( const Dst : TBitmap)
7935:   Proc SpotLight( const Dst : TBitmap; Amount : Int; Spot : TRect)
7936:   Proc SplitLight( const Dst : TBitmap; Amount : Int)
7937:   Proc MakeSeamlessClip( var Dst : TBitmap; Seam : Int)
7938:   Proc Wave( const Dst : TBitmap; Amount, Inference, Style : Int)
7939:   Proc Mosaic( const Bm : TBitmap; Size : Int)
7940:   Proc SmoothRotate( var Src, Dst : TBitmap; CX, CY : Int; Angle : Single)
7941:   Proc SmoothResize( var Src, Dst : TBitmap)
7942:   Proc Twist( var Bmp, Dst : TBitmap; Amount : Int)
7943:   Proc SplitBlur( const Dst : TBitmap; Amount : Int)
7944:   Proc GaussianBlur( const Dst : TBitmap; Amount : Int)
7945:   Proc Smooth( const Dst : TBitmap; Weight : Int)
7946:   Proc GrayScale( const Dst : TBitmap)
7947:   Proc AddColorNoise( const Dst : TBitmap; Amount : Int)
7948:   Proc AddMonoNoise( const Dst : TBitmap; Amount : Int)
7949:   Proc Contrast( const Dst : TBitmap; Amount : Int)
7950:   Proc Lightness( const Dst : TBitmap; Amount : Int)
7951:   Proc Saturation( const Dst : TBitmap; Amount : Int)
7952:   Proc Spray( const Dst : TBitmap; Amount : Int)
7953:   Proc AntiAlias( const Dst : TBitmap)
7954:   Proc AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Int)
7955:   Proc SmoothPoint( const Dst : TBitmap; XK, YK : Int)
7956:   Proc FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7957:   Proc Marble(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7958:   Proc Marble2(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7959:   Proc Marble3(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7960:   Proc Marble4(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7961:   Proc Marble5(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7962:   Proc Marble6(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7963:   Proc Marble7(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7964:   Proc Marble8(const Src:TBitmap;var Dst:TBitmap;Scale:Single;Turbulence:Int)
7965:   Proc SqueezeHor( Src, Dst : TBitmap; Amount : Int; Style : TLightBrush)
7966:   Proc SplitRound( Src, Dst : TBitmap; Amount : Int; Style : TLightBrush)
7967:   Proc Tile( Src, Dst : TBitmap; Amount : Int)
7968:   Proc Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7969:   Proc Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Int)
7970:   Proc Invert( Src : TBitmap)
7971:   Proc MirrorRight( Src : TBitmap)
7972:   Proc MirrorDown( Src : TBitmap)
7973: end;
7974: end;
7975:
7976: (*----------------------------------------------------------------------------*)
7977: Proc SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7978: begin
7979:   AddTypeS('TLightBrush','( lbBrightness, lbContrast, lbSaturation, lbFishe'
7980:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7981:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7982:   SIRegister_TJvPaintFX(CL);
7983:  Func SplineFilter( Value : Single) : Single
7984:  Func BellFilter( Value : Single) : Single
7985:  Func TriangleFilter( Value : Single) : Single
7986:  Func BoxFilter( Value : Single) : Single
7987:  Func HermiteFilter( Value : Single) : Single
7988:  Func Lanczos3Filter( Value : Single) : Single
7989:  Func MitchellFilter( Value : Single) : Single
7990: end;
7991:
7992: (*----------------------------------------------------------------------------*)
7993: Proc SIRegister_Chart(CL: TPSPascalCompiler);
7994: begin
7995:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7996:  TeeMsg_DefaultSeriesName','String 'Series
7997:  TeeMsg_DefaultToolName','String 'ChartTool
7998:  ChartComponentPalette','String 'TeeChart
7999:  TeeMaxLegendColumns',LongInt'( 2);
8000:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
8001:  TeeTitleFootDistance,LongInt( 5);
8002:   SIRegister_TCustomChartWall(CL);
8003:   SIRegister_TChartWall(CL);
8004:   SIRegister_TChartLegendGradient(CL);
8005:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
8006:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
8007:   FindClass('TOBJECT'),'LegendException
8008:   TOnGetLegendText', 'Proc ( Sender : TCustomAxisPanel; Legen'
8009:    +'dStyle : TLegendStyle; Index : Int; var LegendText :Str)
8010:   FindClass('TOBJECT'),'TCustomChartLegend
8011:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
8012:   TLegendSymbolPosition', '( spLeft, spRight )
```

```
8013:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Int;R:TRect);
8014:   TSymbolCalcHeight', 'Func  : Int
8015:   SIRegister_TLegendSymbol(CL);
8016:   SIRegister_TTeeCustomShapePosition(CL);
8017:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
8018:   SIRegister_TLegendTitle(CL);
8019:   SIRegister_TLegendItem(CL);
8020:   SIRegister_TLegendItems(CL);
8021:   TLegendCalcSize', 'Proc ( Sender : TCustomChartLegend; var ASize : Int)
8022:   FindClass('TOBJECT'),'TCustomChart
8023:   SIRegister_TCustomChartLegend(CL);
8024:   SIRegister_TChartLegend(CL);
8025:   SIRegister_TChartTitle(CL);
8026:   SIRegister_TChartFootTitle(CL);
8027:   TChartClick', 'Procedure(ender:TCustomChart;Button:TMouseButton;Shift:TShiftState;X,Y:Int)
8028:   TChartClickAxis', 'Proc ( Sender : TCustomChart; Axis : TCh'
8029:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Int)
8030:   TChartClickSeries', 'Proc ( Sender : TCustomChart; Series :'
8031:    +TChartSeries; ValueIndex : Int; Button: TMouseButton; Shift:TShiftState;X,Y:Int)
8032:   TChartClickTitle', 'Proc ( Sender : TCustomChart; ATitle : '
8033:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Int)
8034:   TOnGetLegendPos', 'Proc (Sender: TCustomChart; Index: Int; var X,Y,XColor:Int)
8035:   TOnGetLegendRect', 'Proc ( Sender : TCustomChart; var Rect : TRect)
8036:   TAxisSavedScales', 'record Auto:Boolean;AutoMin:Bool;AutoMax:Bool;Min:Double;Max:Double;end
8037:   TAllAxisSavedScales', 'array of TAxisSavedScales
8038:   SIRegister_TChartBackWall(CL);
8039:   SIRegister_TChartRightWall(CL);
8040:   SIRegister_TChartBottomWall(CL);
8041:   SIRegister_TChartLeftWall(CL);
8042:   SIRegister_TChartWalls(CL);
8043:   TChartAllowScrollEvent,Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Bool);
8044:   SIRegister_TCustomChart(CL);
8045:   SIRegister_TChart(CL);
8046:   SIRegister_TTeeSeriesTypes(CL);
8047:   SIRegister_TTeeToolTypes(CL);
8048:   SIRegister_TTeeDragObject(CL);
8049:   SIRegister_TColorPalettes(CL);
8050:  Proc RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Int;
8051:  Proc RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
8052:  Proc RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Int;
8053:  Proc RegisterTeeBasicFunction(AFunctionClass: TTeeFunctionClass; ADescription : PString)
8054:  Proc RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription,AGalleryPage:PString;ANumGallerySeries:Int;ASubIndex : Int)
8055:  Proc UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
8056:  Proc UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
8057:  Proc AssignSeries( var OldSeries, NewSeries : TChartSeries)
8058:  Func CreateNewTeeFunction(ASeries:TChartSeries;AClass:TTeeFunctionClass): TTeeFunction
8059:  Func CreateNewSeries( AOwner: TComponent; AChar :TCustomAxisPanel;AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
8060:  Func CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
8061:  Func CloneChartSeries1(ASeries: TChartSeries; AChart: TCustomAxisPanel): TChartSeries;
8062:  Func CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
8063:  Func CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
8064:  Func ChangeSeriesType(var ASeries:TChartSeries;NewType:TChartSeriesClass) : TChartSeries
8065:  Proc ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
8066:  Func GetNewSeriesName( AOwner : TComponent) : TComponentName
8067:  Proc RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
8068:  Proc UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
8069:  Func GetGallerySeriesName( ASeries : TChartSeries) :Str
8070:  Proc PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const R:TRect;ReferenceChart:TCustomChart);
8071:   SIRegister_TChartTheme(CL); //TChartThemeClass', 'class of TChartTheme
8072:   //TCanvasClass', 'class of TCanvas3D
8073:  Func SeriesNameOrIndex( ASeries : TCustomChartSeries) :Str
8074:  Func SeriesTitleOrName( ASeries : TCustomChartSeries) :Str
8075:  Proc FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles :Bool)
8076:  Proc ShowMessageUser( const S :Str)
8077:  Func HasNoMandatoryValues( ASeries : TChartSeries) :Bool
8078:  Func HasLabels( ASeries : TChartSeries):Bool
8079:  Func HasColors( ASeries : TChartSeries):Bool
8080:  Func SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
8081:  Proc TeeDrawBitmapEditor(Canvas: TCanvas;Element: TCustomChartElement;Left,Top:Int)
8082: end;
8083:
8084: Proc SIRegister_TeeProcs(CL: TPSPascalCompiler);
8085: begin
8086:  //'TeeFormBorderStyle',' bsNone);
8087:   SIRegister_TMetafile(CL);
8088:  'TeeDefVerticalMargin','LongInt'( 4);
8089:  'TeeDefHorizMargin','LongInt'( 3);
8090:  'crTeeHand','LongInt'( TCursor ( 2020 ));
8091:  'TeeMsg_TeeHand','String 'crTeeHand
8092:  'TeeNormalPrintDetail','LongInt'( 0);
8093:  'TeeHighPrintDetail','LongInt'( - 100);
8094:  'TeeDefault_PrintMargin','LongInt'( 15);
8095:  'MaxDefaultColors','LongInt'( 19);
8096:  'TeeTabDelimiter',Char #9);
8097:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
```

```
8098:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
8099:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
8100:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
8101:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
8102:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
8103:   SIRegister_TCustomPanelNoCaption(CL);
8104:   FindClass('TOBJECT'),'TCustomTeePanel
8105:   SIRegister_TZoomPanning(CL);
8106:   SIRegister_TTeeEvent(CL);
8107:   //SIRegister_TTeeEventListeners(CL);
8108:   TTeeMouseEventKind', '( meDown, meUp, meMove )
8109:   SIRegister_TTeeMouseEvent(CL);
8110:   SIRegister_TCustomTeePanel(CL);
8111:   //TChartGradient', 'TTeeGradient
8112:   //TChartGradientClass', 'class of TChartGradient
8113:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
8114:   SIRegister_TTeeZoomPen(CL);
8115:   SIRegister_TTeeZoomBrush(CL);
8116:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
8117:   SIRegister_TTeeZoom(CL);
8118:   FindClass('TOBJECT'),'TCustomTeePanelExtended
8119:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
8120:   SIRegister_TBackImage(CL);
8121:   SIRegister_TCustomTeePanelExtended(CL);
8122:   //TChartBrushClass', 'class of TChartBrush
8123:   SIRegister_TTeeCustomShapeBrushPen(CL);
8124:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
8125:   TTextFormat', '( ttfNormal, ttfHtml )
8126:   SIRegister_TTeeCustomShape(CL);
8127:   SIRegister_TTeeShape(CL);
8128:   SIRegister_TTeeExportData(CL);
8129:  Func TeeStr( const Num : Int) :Str
8130:  Func DateTimeDefaultFormat( const AStep : Double) :Str
8131:  Func TEEDaysInMonth( Year, Month : Word) : Word
8132:  Func FindDateTimeStep( const StepValue : Double) : TDateTimeStep
8133:  Func NextDateTimeStep( const AStep : Double) : Double
8134:  Func PointInLine( const P : TPoint; const px, py, qx, qy : Int):Bool;
8135:  Func PointInLine1( const P, FromPoint, ToPoint : TPoint) :Bool;
8136:  Func PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Int):Boo;
8137:  Func PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Int):Bool;
8138:  Func PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Int):Bool;
8139:  Func PointInPolygon( const P : TPoint; const Poly : array of TPoint) :Bool
8140:  Func PointInTriangle( const P, P0, P1, P2 : TPoint) :Bool;
8141:  Func PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Int) :Bool;
8142:  Func PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Int) :Bool
8143:  Func PointInEllipse( const P : TPoint; const Rect : TRect) :Bool;
8144:  Func PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Int) :Bool;
8145:  Func DelphiToLocalFormat( const Format :Str) :Str
8146:  Func LocalToDelphiFormat( const Format :Str) :Str
8147:  Proc TEEEnableControls(Enable:Bool; const ControlArray : array of TControl)
8148:  Func TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8149:  Proc TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const AnIncrement:Double;
     tmpWhichDateTime:TDateTimeStep)
8150:   TTeeSortCompare', 'Func ( a, b : Int) : Int
8151:   TTeeSortSwap', 'Proc ( a, b : Int)
8152:  Proc TeeSort(StartIndex,EndIndex:Int;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8153:  Func TeeGetUniqueName( AOwner : TComponent; const AStartName :Str) :Str
8154:  Func TeeExtractField( St :Str; Index : Int) :Str;
8155:  Func TeeExtractField1( St :Str; Index : Int; const Separator :Str) :Str;
8156:  Func TeeNumFields( St :Str) : Int;
8157:  Func TeeNumFields1( const St, Separator :Str) : Int;
8158:  Proc TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8159:  Proc TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 :Str)
8160:  // TColorArray', 'array of TColor
8161:  Func GetDefaultColor( const Index : Int) : TColor
8162:  Proc SetDefaultColorPalette;
8163:  Proc SetDefaultColorPalette1( const Palette : array of TColor);
8164:  'TeeCheckBoxSize','LongInt'( 11);
8165:  Proc TeeDrawCheckBox(x,y:Int;Canvas:TCanvas;Checked:Bool;ABackColor:TColor;CheckBox:Bool);
8166:  Func TEEStrToFloatDef( const S :Str; const Default : Extended) : Extended
8167:  Func TryStrToFloat( const S :Str; var Value : Double) :Bool
8168:  Func CrossingLines(const X1,Y1,X2,Y2,X3,Y3,X4,Y4: Double; out x, y : Double) :Bool
8169:  Proc TeeTranslateControl( AControl : TControl);
8170:  Proc TeeTranslateControl1(AControl: TControl;const ExcludeChilds: array of TControl);
8171:  Func ReplaceChar(const AString:str;const Search: Char; const Replace : Char) :Str
8172:  //Proc RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8173:  Func TeeAntiAlias( Panel : TCustomTeePanel; ChartRect :Bool) : TBitmap
8174:  //Proc DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Int;Round:Int);
8175:  //Func ScreenRatio( ACanvas : TCanvas3D) : Double
8176:  Func TeeReadBoolOption( const AKey :Str; DefaultValue :Bool) :Bool
8177:  Proc TeeSaveBoolOption( const AKey :Str; Value :Bool)
8178:  Func TeeReadIntOption( const AKey :Str; DefaultValue : Int) : Int
8179:  Proc TeeSaveIntOption( const AKey :Str; Value : Int)
8180:  Func TeeReadStringOption( const AKey, DefaultValue :Str) :Str
8181:  Proc TeeSaveStringOption( const AKey, Value :Str)
8182:  Func TeeDefaultXMLEncoding :Str
8183:  Proc ConvertTextToXML(Stream: TStream; XMLHeader :Bool)
8184:   TeeWindowHandle', 'Int
8185:  Proc TeeGotoURL( Handle : TeeWindowHandle; const URL :Str)
```

```
8186:  Proc HtmlTextOut( ACanvas : TCanvas; x, y : Int; Text :Str)
8187:  Func HtmlTextExtent( ACanvas : TCanvas; const Text :Str) : TSize
8188: end;
8189:
8190: fix and add teEngine:  4.7.6.20
8191: TChartAxisTitle=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8192: TChartSeriesList=class {$IFDEF CLR}sealed{$ENDIF} (TCustomSeriesList)
8193: TSeriesGroup=class {$IFDEF CLR}sealed{$ENDIF} (TCollectionItem)
8194:   Destructor Destroy; override;
8195: TSeriesGroups=class {$IFDEF CLR}sealed{$ENDIF} (TOwnedCollection)
8196:  TChartCustomAxes=class {$IFDEF CLR}sealed{$ENDIF} (TOwnedCollection)
8197:  TChartChangePage=class {$IFDEF CLR}sealed{$ENDIF} (TTeeEvent);
8198:  TSeriesMarksPositions=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8199:  TMarksItem=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8200:    property Bevel;
8201:     property BevelWidth;
8202:     property Color default ChartMarkColor;
8203:     property Font;
8204:     property Gradient;
8205:     property Picture;
8206:     property Shadow;
8207:     property ShapeStyle;
8208:     property Text:TStrings read GetText write SetText;
8209:     property Transparency;
8210:     property Transparent;
8211:   TMarksItems=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8212:   TSeriesMarksGradient=class {$IFDEF CLR}sealed{$ENDIF} (TChartGradient)
8213:   property Direction default gdRightLeft;
8214:     property EndColor default clWhite;
8215:     property StartColor default clSilver;
8216:     TSeriesMarksSymbol=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8217:       TChartValueLists=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8218:    Procedure Clear; override;
8219:  TEEProcs
8220:  TTeeMouseEvent=class {$IFDEF CLR}sealed{$ENDIF} (TTeeEvent)
8221:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TTeeMouseEvent') do begin
8222:  CNN CreateAscentImages   - unit neuraldatasetsv;
8223:
8224:
8225: using mXBDEUtils
8226: ******************************************************************************
8227:  Proc SetAlias( aAlias, aDirectory :str)
8228:  Proc CheckRegistryEntry(Reg:TRegistry;const Path,Value:Str;const Default,Desired:Variant;Size:Byte);
8229:  Func GetFileVersionNumber( const FileName :Str) : TVersionNo
8230:  Proc SetBDE( aPath, aNode, aValue :Str)
8231:  Func RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Int): Int; stdcall;
8232:  Func GetSystemDirectory( lpBuffer :Str; uSize : UINT) : UINT
8233:  Func GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8234:  Func GetTempPath( nBufferLength : DWORD; lpBuffer :Str) : DWORD
8235:  Func GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer :Str) : DWORD
8236:  Func GetTempFileName(lpPathName,lpPrefixString:str;uUnique:UINT;lpTempFileName:str):UINT;
8237:
8238: Proc SIRegister_cDateTime(CL: TPSPascalCompiler);
8239: begin
8240: AddClassN(FindClass('TOBJECT'),'EDateTime
8241: Func DatePart( const D : TDateTime) : Int
8242: Func TimePart( const D : TDateTime) : Double
8243: Func Century( const D : TDateTime) : Word
8244: Func Year( const D : TDateTime) : Word
8245: Func Month( const D : TDateTime) : Word
8246: Func Day( const D : TDateTime) : Word
8247: Func Hour( const D : TDateTime) : Word
8248: Func Minute( const D : TDateTime) : Word
8249: Func Second( const D : TDateTime) : Word
8250: Func Millisecond( const D : TDateTime) : Word
8251: ('OneDay','Extended').setExtended( 1.0);
8252: ('OneHour','Extended').SetExtended( OneDay / 24);
8253: ('OneMinute','Extended').SetExtended( OneHour / 60);
8254: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8255: ('OneMillisecond','Extended').SetExtended( OneSecond/ 1000);
8256: ('OneWeek','Extended').SetExtended( OneDay * 7);
8257: ('HoursPerDay','Extended').SetExtended( 24);
8258: ('MinutesPerHour','Extended').SetExtended( 60);
8259: ('SecondsPerMinute','Extended').SetExtended( 60);
8260: Proc SetYear( var D : TDateTime; const Year : Word)
8261: Proc SetMonth( var D : TDateTime; const Month : Word)
8262: Proc SetDay( var D : TDateTime; const Day : Word)
8263: Proc SetHour( var D : TDateTime; const Hour : Word)
8264: Proc SetMinute( var D : TDateTime; const Minute : Word)
8265: Proc SetSecond( var D : TDateTime; const Second : Word)
8266: Proc SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8267: Func IsEqual( const D1, D2 : TDateTime) :Bool;
8268: Func IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8269: Func IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8270: Func IsAM( const D : TDateTime) :Bool
8271: Func IsPM( const D : TDateTime) :Bool
8272: Func IsMidnight( const D : TDateTime) :Bool
8273: Func IsNoon( const D : TDateTime) :Bool
8274: Func IsSunday( const D : TDateTime) :Bool
```

```
8275: Func IsMonday( const D : TDateTime) :Bool
8276: Func IsTuesday( const D : TDateTime) :Bool
8277: Func IsWedneday( const D : TDateTime) :Bool
8278: Func IsThursday( const D : TDateTime) :Bool
8279: Func IsFriday( const D : TDateTime) :Bool
8280: Func IsSaturday( const D : TDateTime) :Bool
8281: Func IsWeekend( const D : TDateTime) :Bool
8282: Func Noon( const D : TDateTime) : TDateTime
8283: Func Midnight( const D : TDateTime) : TDateTime
8284: Func FirstDayOfMonth( const D : TDateTime) : TDateTime
8285: Func LastDayOfMonth( const D : TDateTime) : TDateTime
8286: Func NextWorkday( const D : TDateTime) : TDateTime
8287: Func PreviousWorkday( const D : TDateTime) : TDateTime
8288: Func FirstDayOfYear( const D : TDateTime) : TDateTime
8289: Func LastDayOfYear( const D : TDateTime) : TDateTime
8290: Func EasterSunday( const Year : Word) : TDateTime
8291: Func GoodFriday( const Year : Word) : TDateTime
8292: Func AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8293: Func AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8294: Func AddMinutes( const D : TDateTime; const N : Int) : TDateTime
8295: Func AddHours( const D : TDateTime; const N : Int) : TDateTime
8296: Func AddDays( const D : TDateTime; const N : Int) : TDateTime
8297: Func AddWeeks( const D : TDateTime; const N : Int) : TDateTime
8298: Func AddMonths( const D : TDateTime; const N : Int) : TDateTime
8299: Func AddYears( const D : TDateTime; const N : Int) : TDateTime
8300: Func DayOfYear( const Ye, Mo, Da : Word) : Int
8301: Func DayOfYear( const D : TDateTime) : Int
8302: Func DaysInMonth( const Ye, Mo : Word) : Int
8303: Func DaysInMonth( const D : TDateTime) : Int
8304: Func DaysInYear( const Ye : Word) : Int
8305: Func DaysInYearDate( const D : TDateTime) : Int
8306: Func WeekNumber( const D : TDateTime) : Int
8307: Func ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8308: Proc ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8309: Func DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8310: Func DiffSeconds( const D1, D2 : TDateTime) : Int
8311: Func DiffMinutes( const D1, D2 : TDateTime) : Int
8312: Func DiffHours( const D1, D2 : TDateTime) : Int
8313: Func DiffDays( const D1, D2 : TDateTime) : Int
8314: Func DiffWeeks( const D1, D2 : TDateTime) : Int
8315: Func DiffMonths( const D1, D2 : TDateTime) : Int
8316: Func DiffYears( const D1, D2 : TDateTime) : Int
8317: Func GMTBias : Int
8318: Func GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8319: Func LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8320: Func NowAsGMTTime : TDateTime
8321: Func DateTimeToISO8601String( const D : TDateTime) : Ansistr
8322: Func ISO8601StringToTime( const D : Ansistr) : TDateTime
8323: Func ISO8601StringAsDateTime( const D : Ansistr) : TDateTime
8324: Func DateTimeToANSI( const D : TDateTime) : Int
8325: Func ANSIToDateTime( const Julian : Int) : TDateTime
8326: Func DateTimeToISOInt( const D : TDateTime) : Int
8327: Func DateTimeToISOString( const D : TDateTime) : Ansistr
8328: Func ISOIntToDateTime( const ISOInt : Int) : TDateTime
8329: Func TwoDigitRadix2000YearToYear( const Y : Int) : Int
8330: Func DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):Ansistr
8331: Func UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8332: Func DateTimeToUnixTime( const D : TDateTime) : LongWord
8333: Func EnglishShortDayOfWeekStrA( const DayOfWeek : Int) : Ansistr
8334: Func EnglishShortDayOfWeekStrU( const DayOfWeek : Int) : UnicodeString
8335: Func EnglishLongDayOfWeekStrA( const DayOfWeek : Int) : Ansistr
8336: Func EnglishLongDayOfWeekStrU( const DayOfWeek : Int) : UnicodeString
8337: Func EnglishShortMonthStrA( const Month : Int) : Ansistr
8338: Func EnglishShortMonthStrU( const Month : Int) : UnicodeString
8339: Func EnglishLongMonthStrA( const Month : Int) : Ansistr
8340: Func EnglishLongMonthStrU( const Month : Int) : UnicodeString
8341: Func EnglishShortDayOfWeekA( const S : Ansistr) : Int
8342: Func EnglishShortDayOfWeekU( const S : UnicodeString) : Int
8343: Func EnglishLongDayOfWeekA( const S : Ansistr) : Int
8344: Func EnglishLongDayOfWeekU( const S : UnicodeString) : Int
8345: Func EnglishShortMonthA( const S : Ansistr) : Int
8346: Func EnglishShortMonthU( const S : UnicodeString) : Int
8347: Func EnglishLongMonthA( const S : Ansistr) : Int
8348: Func EnglishLongMonthU( const S : UnicodeString) : Int
8349: Func RFC850DayOfWeekA( const S : Ansistr) : Int
8350: Func RFC850DayOfWeekU( const S : UnicodeString) : Int
8351: Func RFC1123DayOfWeekA( const S : Ansistr) : Int
8352: Func RFC1123DayOfWeekU( const S : UnicodeString) : Int
8353: Func RFCMonthA( const S : Ansistr) : Word
8354: Func RFCMonthU( const S : UnicodeString) : Word
8355: Func GMTTimeToRFC1123TimeA( const D : TDateTime;const IncludeSeconds:Boolean) : Ansistr
8356: Func GMTTimeToRFC1123TimeU( const D : TDateTime;const IncludeSeconds:Boolean):UnicodeString
8357: Func GMTDateTimeToRFC1123DateTimeA(const D:TDateTime;const IncludeDayOfWeek:Bool):AnsiStr;
8358: Func GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodStr;
8359: Func DateTimeToRFCDateTimeA( const D : TDateTime) : Ansistr
8360: Func DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8361: Func NowAsRFCDateTimeA : Ansistr
8362: Func NowAsRFCDateTimeU : UnicodeString
8363: Func RFCDateTimeToGMTDateTime( const S : Ansistr) : TDateTime
```

```
8364: Func RFCDateTimeToDateTime( const S : Ansistr) : TDateTime
8365: Func RFCTimeZoneToGMTBias( const Zone : Ansistr) : Int
8366: Func TimePeriodStr( const D : TDateTime) : Ansistr
8367: Proc SelfTestCDate
8368: end;
8369: //*******************************************CFileUtils
8370: Func PathHasDriveLetterA( const Path : Ansistr) :Bool
8371: Func PathHasDriveLetter( const Path :Str) :Bool
8372: Func PathIsDriveLetterA( const Path : Ansistr) :Bool
8373: Func PathIsDriveLetter( const Path :Str) :Bool
8374: Func PathIsDriveRootA( const Path : Ansistr) :Bool
8375: Func PathIsDriveRoot( const Path :Str) :Bool
8376: Func PathIsRootA( const Path : Ansistr) :Bool
8377: Func PathIsRoot( const Path :Str) :Bool
8378: Func PathIsUNCPathA( const Path : Ansistr) :Bool
8379: Func PathIsUNCPath( const Path :Str) :Bool
8380: Func PathIsAbsoluteA( const Path : Ansistr) :Bool
8381: Func PathIsAbsolute( const Path :Str) :Bool
8382: Func PathIsDirectoryA( const Path : Ansistr) :Bool
8383: Func PathIsDirectory( const Path :Str) :Bool
8384: Func PathInclSuffixA( const Path : Ansistr; const PathSep : Char) : Ansistr
8385: Func PathInclSuffix( const Path :Str; const PathSep : Char) :Str
8386: Func PathExclSuffixA( const Path : Ansistr; const PathSep : Char) : Ansistr
8387: Func PathExclSuffix( const Path :Str; const PathSep : Char) :Str
8388: Proc PathEnsureSuffixA( var Path : Ansistr; const PathSep : Char)
8389: Proc PathEnsureSuffix( var Path :Str; const PathSep : Char)
8390: Proc PathEnsureNoSuffixA( var Path : Ansistr; const PathSep : Char)
8391: Proc PathEnsureNoSuffix( var Path :Str; const PathSep : Char)
8392: //Func PathCanonicalA( const Path : Ansistr; const PathSep : Char) : Ansistr
8393: Func PathCanonical( const Path :Str; const PathSep : Char) :Str
8394: Func PathExpandA(const Path:Ansistr;const BasePath:AnsiStr;const PathSep:Char):AnsiStr
8395: Func PathExpand(const Path:str; const BasePath :Str; const PathSep : Char) :Str
8396: Func PathLeftElementA( const Path : Ansistr; const PathSep : Char) : Ansistr
8397: Func PathLeftElement( const Path :Str; const PathSep : Char) :Str
8398: Proc PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8399: Proc PathSplitLeftElement(const Path:str; var LeftElement,RightPath:Str;const PathSep:Char);
8400: Proc DecodeFilePathA(const FilePath:AnsiStr;var Path,FileName:AnsiStr;const PathSep:Char;
8401: Proc DecodeFilePath(const FilePath:str; var Path,FileName:str;const PathSep: Char)
8402: Func FileNameValidA( const FileName : Ansistr) : Ansistr
8403: Func FileNameValid( const FileName :Str) :Str
8404: Func FilePathA(const FileName,Path:Ansistr;const BasePath:AnsiStr;const PathSep:Char):Ansistr;
8405: Func FilePath(const FileName,Path:str;const BasePath:Str;const PathSep:Char):Str
8406: Func DirectoryExpandA(const Path:Ansistr;const BasePath:Ansistr;const PathSep:Char):Ansistr
8407: Func DirectoryExpand(const Path:str;const BasePath:str;const PathSep:Char):str
8408: Func UnixPathToWinPath( const Path : Ansistr) : Ansistr
8409: Func WinPathToUnixPath( const Path : Ansistr) : Ansistr
8410: Proc CCopyFile( const FileName, DestName :Str)
8411: Proc CMoveFile( const FileName, DestName :Str)
8412: Func CDeleteFiles( const FileMask :Str) :Bool
8413: Func FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64;const FilePos:TFileSeekPos):Int64;
8414: Proc FileCloseEx( const FileHandle : TFileHandle)
8415: Func FileExistsA( const FileName : Ansistr) :Bool
8416:  Func CFileExists( const FileName :Str) :Bool
8417:  Func CFileGetSize( const FileName :Str) : Int64
8418: Func FileGetDateTime( const FileName :Str) : TDateTime
8419: Func FileGetDateTime2( const FileName :Str) : TDateTime
8420: Func FileIsReadOnly( const FileName :Str) :Bool
8421: Proc FileDeleteEx( const FileName :Str)
8422: Proc FileRenameEx( const OldFileName, NewFileName :Str)
8423: Func ReadFileStrA( const FileName:Ansistr; const FileSharing : TFileSharing; const FileCreationMode :
      TFileCreationMode; const FileOpenWait : PFileOpenWait) : Ansistr
8424: Func DirectoryEntryExists( const Name :Str) :Bool
8425: Func DirectoryEntrySize( const Name :Str) : Int64
8426:  Func CDirectoryExists( const DirectoryName :Str) :Bool
8427: Func DirectoryGetDateTime( const DirectoryName :Str) : TDateTime
8428: Proc CDirectoryCreate( const DirectoryName :Str)
8429: Func GetFirstFileNameMatching( const FileMask :Str) :Str
8430: Func DirEntryGetAttr( const FileName : Ansistr) : Int
8431: Func DirEntryIsDirectory( const FileName : Ansistr) :Bool
8432: Func FileHasAttr( const FileName :Str; const Attr : Word) :Bool
8433:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8434:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8435: Func DriveIsValid( const Drive : Char) :Bool
8436: Func DriveGetType( const Path : Ansistr) : TLogicalDriveType
8437: Func DriveFreeSpace( const Path : Ansistr) : Int64
8438:
8439: Proc SIRegister_cTimers(CL: TPSPascalCompiler);
8440: begin
8441:  AddClassN(FindClass('TOBJECT'),'ETimers
8442:  Const('TickFrequency','LongInt'( 1000);Func GetTick : LongWord
8443: Func TickDelta( const D1, D2 : LongWord) : Int
8444: Func TickDeltaW( const D1, D2 : LongWord) : LongWord
8445:   AddTypeS('THPTimer', 'Int64
8446: Proc StartTimer( var Timer : THPTimer)
8447: Proc StopTimer( var Timer : THPTimer)
8448: Proc ResumeTimer( var StoppedTimer : THPTimer)
8449: Proc InitStoppedTimer( var Timer : THPTimer)
8450: Proc InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Int)
8451: Func MillisecondsElapsed( const Timer: THPTimer; const TimerRunning :Bool) : Int
```

```
8452: Func MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning :Bool) : Int64
8453: Proc WaitMicroseconds( const MicroSeconds : Int)
8454: Func GetHighPrecisionFrequency : Int64
8455: Func GetHighPrecisionTimerOverhead : Int64
8456: Proc AdjustTimerForOverhead(var StoppedTimer:THPTimer; const Overhead:Int64)
8457: Proc SelfTestCTimer
8458: end;
8459:
8460: Proc SIRegister_cRandom(CL: TPSPascalCompiler);
8461: begin
8462:  Func RandomSeed : LongWord
8463:  Proc AddEntropy( const Value : LongWord)
8464:  Func RandomUniform : LongWord;
8465:  Func RandomUniform1( const N : Int) : Int;
8466:  Func RandomBoolean :Bool
8467:  Func RandomByte : Byte
8468:  Func RandomByteNonZero : Byte
8469:  Func RandomWord : Word
8470:  Func RandomInt64 : Int64;
8471:  Func RandomInt641( const N : Int64) : Int64;
8472:  Func RandomHex( const Digits : Int) :Str
8473:  Func RandomFloat : Extended
8474:  Func RandomAlphaStr( const Length : Int) : Ansistr
8475:  Func RandomPseudoWord( const Length : Int) : Ansistr
8476:  Func RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiStr;
8477:  Func mwcRandomLongWord : LongWord
8478:  Func urnRandomLongWord : LongWord
8479:  Func moaRandomFloat : Extended
8480:  Func mwcRandomFloat : Extended
8481:  Func RandomNormalF : Extended
8482:  Proc SelfTestCRandom
8483: end;
8484:
8485: Proc SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8486: begin
8487:  // PIntArray', '^TIntArray // will not work
8488:  Addtypes('TConvertTabsProc','function(const Line:Ansistr; TabWidth: Int):Ansistr
8489:  TConvertTabsProcEx,function(const Line:Ansistr;TabWidth:Int;var HasTabs:boolean):Ansistr
8490:  Func synMax( x, y : Int) : Int
8491:  Func synMin( x, y : Int) : Int
8492:  Func synMinMax( x, mi, ma : Int) : Int
8493:  Proc synSwapInt( var l, r : Int)
8494:  Func synMaxPoint( const P1, P2 : TPoint) : TPoint
8495:  Func synMinPoint( const P1, P2 : TPoint) : TPoint
8496:  //Func synGetIntArray( Count :Card; InitialValue : Int) : PIntArray
8497:  Proc synInternalFillRect( dc : HDC; const rcPaint : TRect)
8498:  Func synGetBestConvertTabsProc( TabWidth : Int) : TConvertTabsProc
8499:  Func synConvertTabs( const Line : Ansistr; TabWidth : Int) : Ansistr
8500:  Func synGetBestConvertTabsProcEx( TabWidth : Int) : TConvertTabsProcEx
8501:  Func synConvertTabsEx(const Line:Ansistr;TabWidth:Int;var HasTabs:bool):Ansistr;
8502:  Func synGetExpandedLength( const aStr :Str; aTabWidth : Int) : Int
8503:  Func synCharIndex2CaretPos( Index, TabWidth : Int; const Line :Str) : Int
8504:  Func synCaretPos2CharIndex(Positin,TabWidth:int;const Line:str;var InsideTabChar:bool):int;
8505:  Func synStrScanForCharInSet(const Line:str;Start:Int;AChars:TSynIdentChars):int;
8506:  Func synStrRScanForCharInSet(const Line:str;Start:Int;AChars:TSynIdentChars):Int;
8507:   TStringType', '(stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8508:  +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida)
8509:  ('C3_NONSPACING','LongInt'( 1);
8510:  'C3_DIACRITIC','LongInt'( 2);
8511:  'C3_VOWELMARK','LongInt'( 4);
8512:  ('C3_SYMBOL','LongInt'( 8);
8513:  ('C3_KATAKANA',LongWord( $0010);
8514:  ('C3_HIRAGANA',LongWord( $0020);
8515:  ('C3_HALFWIDTH',LongWord( $0040);
8516:  ('C3_FULLWIDTH',LongWord( $0080);
8517:  ('C3_IDEOGRAPH',LongWord( $0100);
8518:  ('C3_KASHIDA',LongWord( $0200);
8519:  ('C3_LEXICAL',LongWord( $0400);
8520:  ('C3_ALPHA',LongWord( $8000);
8521:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8522:  Func synStrScanForMultiByteChar( const Line :Str; Start : Int) : Int
8523:  Func synStrRScanForMultiByteChar( const Line :Str; Start : Int) : Int
8524:  Func synIsStringType( Value : Word) : TStringType
8525:  Func synGetEOL( Line : PChar) : PChar
8526:  Func synEncodeString( s :Str) :Str
8527:  Func synDecodeString( s :Str) :Str
8528:  Proc synFreeAndNil( var Obj: TObject)
8529:  Proc synAssert( Expr :Bool)
8530:  Func synLastDelimiter( const Delimiters, S :Str) : Int
8531:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8532:   TReplaceFlags', 'set of TReplaceFlag )
8533:  Func synStringReplace(const S, OldPattern,NewPattern:str;Flags:TReplaceFlags):str
8534:  Func synGetRValue( RGBValue : TColor) : byte
8535:  Func synGetGValue( RGBValue : TColor) : byte
8536:  Func synGetBValue( RGBValue : TColor) : byte
8537:  Func synRGB( r, g, b : Byte) :Card
8538: //THighlighterAttriProc', 'Func ( Highlighter : TSynCustomHigh'
8539: //+'lighter;Attri:TSynHighlighterAttributes;UniqueAttriName:str;Params array of Pointr):Bool;
8540:  //Func synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates :Bool;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) :Bool
```

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

```
8629: Func Kermit16OfFile( FileName : Ansistr) :Card
8630: end;
8631:
8632: Proc SIRegister_ComObj(cl: TPSPascalCompiler);
8633: begin
8634:  Func CreateOleObject(const ClassName:Str): IDispatch;
8635:  Func GetActiveOleObject(const ClassName:Str): IDispatch;
8636:  Func ProgIDToClassID(const ProgID:Str): TGUID;
8637:  Func ClassIDToProgID(const ClassID: TGUID):Str;
8638:  Func CreateClassID:Str;
8639:  Func CreateGUIDString:Str;
8640:  Func CreateGUIDID:Str;
8641:  Proc OleError(ErrorCode: longint)
8642:  Proc OleCheck(Result: HResult);
8643: end;
8644:
8645:  Func xCreateOleObject( const ClassName :Str) : Variant //or IDispatch
8646:  Func xGetActiveOleObject( const ClassName :Str) : Variant
8647:  //Func DllGetClassObject(const CLSID:TCLSID;const IID TIID;var Obj):HResult
8648:  Func DllCanUnloadNow : HResult
8649:  Func DllRegisterServer : HResult
8650:  Func DllUnregisterServer : HResult
8651:  Func VarFromInterface( Unknown : IUnknown) : Variant
8652:  Func VarToInterface( const V : Variant) : IDispatch
8653:  Func VarToAutoObject( const V : Variant) : TAutoObject
8654:  //Proc DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8655:  //Proc DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8656:  Proc OleError( ErrorCode : HResult)
8657:  Proc OleCheck( Result : HResult)
8658:  Func StringToClassID( const S :Str) : TCLSID
8659:  Func ClassIDToString( const ClassID : TCLSID) :Str
8660:  Func xProgIDToClassID( const ProgID :Str) : TCLSID
8661:  Func xClassIDToProgID( const ClassID : TCLSID) :Str
8662:  Func xWideCompareStr( const S1, S2 : WideString) : Int
8663:  Func xWideSameStr( const S1, S2 : WideString) :Bool
8664:  Func xGUIDToString( const ClassID : TGUID) :Str
8665:  Func xStringToGUID( const S :Str) : TGUID
8666:  Func xGetModuleName( Module : HMODULE) :Str
8667:  Func xAcquireExceptionObject : TObject
8668:  Func xIfThen(AValue :Bool; const ATrue: Int; const AFalse: Int): Int
8669:  Func xUtf8Encode( const WS : WideString) : UTF8String
8670:  Func xUtf8Decode( const S : UTF8String) : WideString
8671:  Func xExcludeTrailingPathDelimiter( const S :Str) :Str
8672:  Func xIncludeTrailingPathDelimiter( const S :Str) :Str
8673:  Func XRTLHandleCOMException : HResult
8674:  Proc XRTLCheckArgument( Flag :Bool)
8675:  //Proc XRTLCheckOutArgument( out Arg)
8676:  Proc XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8677:  Proc XRTLInterfaceDisconnect(const Source:IUnknown;const IID:TIID;var Connection:Longint)
8678:  Func XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8679:  Func XRTLUnRegisterActiveObject( var RegisterCookie : Int) : HResult
8680:  //Func XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8681:  Proc XRTLEnumActiveObjects( Strings : TStrings)
8682: Func XRTLDefaultCategoryManager: IUnknown;
8683: Func XRTLIsCategoryEmpty(CatID:TGUID;const CategoryManager:IUnknown=nil):Bool;
8684: // ICatRegister helper functions
8685: Func  XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8686:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8687:                                const CategoryManager: IUnknown = nil): HResult;
8688: Func  XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8689:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8690:                                const CategoryManager: IUnknown = nil): HResult;
8691: Func  XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8692:                                const CategoryManager: IUnknown = nil): HResult;
8693: Func  XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8694:                                const CategoryManager: IUnknown = nil): HResult;
8695: // ICatInformation helper functions
8696: Func  XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8697:                                     LocaleID: TLCID = LOCALE_USER_DEFAULT;
8698:                                const CategoryManager: IUnknown = nil): HResult;
8699: Func  XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8700:                                const CategoryManager: IUnknown = nil): HResult;
8701: Func  XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8702:                                const CategoryManager: IUnknown = nil): HResult;
8703: Func  XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8704:                                const CategoryManager: IUnknown = nil): HResult;
8705: Func XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8706:                   const ADelete:Bool = True): WideString;
8707: Func XRTLRPos(const ASub, AIn: WideString; AStart: Int = -1): Int;
8708: Func XRTLGetVariantAsString( const Value : Variant) :Str
8709: Func XRTLDateTimeToTimeZoneTime(DT:TDateTime;TimeZone:TXRTLTimeZone):TDateTime
8710: Func XRTLGetTimeZones : TXRTLTimeZones
8711: Func XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8712: Func DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8713: Func GMTNow : TDateTime
8714: Func GMTToLocalTime( GMT : TDateTime) : TDateTime
8715: Func LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
```

```
8716: Proc XRTLNotImplemented
8717: Proc XRTLRaiseError( E : Exception)
8718: Proc XRTLRaise( E : Exception);
8719: Proc XRaise( E : Exception);
8720: Proc XRTLInvalidOperation(ClassName:str;OperationName:str;Description:Str)
8721:
8722: Proc SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8723: begin
8724:   SIRegister_IXRTLValue(CL);
8725:   SIRegister_TXRTLValue(CL);
8726:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8727:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8728: Func XRTLValue( const AValue :Card) : IXRTLValue;
8729: Func XRTLSetValue(const IValue:IXRTLValue; const AValue:Card) :Card;
8730: Func XRTLGetAsCardinal( const IValue : IXRTLValue) :Card
8731: Func XRTLGetAsCardinalDef(const IValue:IXRTLValue;const DefValue:Card): Card
8732: Func XRTLValue1( const AValue : Int) : IXRTLValue;
8733: Func XRTLSetValue1( const IValue : IXRTLValue; const AValue : Int) : Int;
8734: Func XRTLGetAsInt( const IValue : IXRTLValue) : Int
8735: Func XRTLGetAsIntDef( const IValue : IXRTLValue; const DefValue : Int) : Int
8736: Func XRTLValue2( const AValue : Int64) : IXRTLValue;
8737: Func XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8738: Func XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8739: Func XRTLGetAsInt64Def(const IValue:IXRTLValue;const DefValue:Int64) : Int64
8740: Func XRTLValue3( const AValue : Single) : IXRTLValue;
8741: Func XRTLSetValue3( const IValue : IXRTLValue; const AValue:Single) : Single;
8742: Func XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8743: Func XRTLGetAsSingleDef(const IValue: RTLValue;const DefValue: Single):Single
8744: Func XRTLValue4( const AValue : Double) : IXRTLValue;
8745: Func XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8746: Func XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8747: Func XRTLGetAsDoubleDef(const IValue:IXRTLValue;const DefValue:Double):Double
8748: Func XRTLValue5( const AValue : Extended) : IXRTLValue;
8749: Func XRTLSetValue5(const IValue:IXRTLValue;const AValue:Extended): Extended;
8750: Func XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8751: Func XRTLGetAsExtendedDef(const IValue:IXRTLValue;const DefValue:Extended): Extended
8752: Func XRTLValue6( const AValue : IInterface) : IXRTLValue;
8753: Func XRTLSetValue6(const IValue:IXRTLValue;const AValue:IInterface):IInterface;
8754: Func XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8755:  //Func XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8756: Func XRTLGetAsInterfaceDef(const IValue:IXRTLValue;const DefValue:IInterface):IInterface;
8757: Func XRTLValue7( const AValue : WideString) : IXRTLValue;
8758: Func XRTLSetValue7(const IValue:IXRTLValue;const AValue:WideString):WideString;
8759: Func XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8760: Func XRTLGetAsWideStringDef(const IValue:IXRTLValue;const DefValue:WideString): WideString
8761: Func XRTLValue8(const AValue:TObject; const AOwnValue:Boolean): IXRTLValue;
8762: Func XRTLSetValue8(const IValue:IXRTLValue;const AValue:TObject) : TObject;
8763: Func XRTLGetAsObject(const IValue:IXRTLValue;const ADetachOwnership:Bool): TObject;
8764: Func XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8765: //Func XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8766: //Func XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8767: //Func XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8768: //Func XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8769: Func XRTLValueV( const AValue : Variant) : IXRTLValue;
8770: Func XRTLSetValueV(const IValue:IXRTLValue; const AValue : Variant): Variant;
8771: Func XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8772: Func XRTLGetAsVariantDef(const IValue:IXRTLValue;const DefValue:Variant): Variant
8773: Func XRTLValue10( const AValue : Currency) : IXRTLValue;
8774: Func XRTLSetValue10(const IValue:IXRTLValue; const AValue:Currency):Currency;
8775: Func XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8776: Func XRTLGetAsCurrencyDef(const IValue:IXRTLValue;const DefValue:Currency): Currency
8777: Func XRTLValue11( const AValue : Comp) : IXRTLValue;
8778: Func XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8779: Func XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8780: Func XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8781: Func XRTLValue12( const AValue : TClass) : IXRTLValue;
8782: Func XRTLSetValue12( const IValue:IXRTLValue; const AValue : TClass) : TClass;
8783: Func XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8784: Func XRTLGetAsClassDef(const IValue: IXRTLValue;const DefValue:TClass):TClass
8785: Func XRTLValue13( const AValue : TGUID) : IXRTLValue;
8786: Func XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8787: Func XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8788: Func XRTLGetAsGUIDDef(const IValue:IXRTLValue; const DefValue : TGUID) : TGUID
8789: Func XRTLValue14( const AValue :Bool) : IXRTLValue;
8790: Func XRTLSetValue14( const IValue:IXRTLValue;const AValue:Boolean):Boolean;
8791: Func XRTLGetAsBoolean( const IValue : IXRTLValue) :Bool
8792: Func XRTLGetAsBooleanDef(const IValue:IXRTLValue;const DefValue:Bool):Bool
8793: end;
8794:
8795: //*****************************unit uPSI_GR32;*******************************
8796:
8797:  Func Color32( WinColor : TColor) : TColor32;
8798:  Func Color321( R, G, B : Byte; A : Byte) : TColor32;
8799:  Func Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8800:  Func Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8801:  Func WinColor( Color32 : TColor32) : TColor
8802:  Func ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8803:  Proc Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
```

```
8804:  Proc Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8805:  Func Color32Components( R, G, B, A :Bool) : TColor32Components
8806:  Func RedComponent( Color32 : TColor32) : Int
8807:  Func GreenComponent( Color32 : TColor32) : Int
8808:  Func BlueComponent( Color32 : TColor32) : Int
8809:  Func AlphaComponent( Color32 : TColor32) : Int
8810:  Func Intensity( Color32 : TColor32) : Int
8811:  Func SetAlpha( Color32 : TColor32; NewAlpha : Int) : TColor32
8812:  Func HSLtoRGB( H, S, L : Single) : TColor32;
8813:  Proc RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8814:  Func HSLtoRGB1( H, S, L : Int) : TColor32;
8815:  Proc RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8816:  Func WinPalette( const P : TPalette32) : HPALETTE
8817:  Func FloatPoint( X, Y : Single) : TFloatPoint;
8818:  Func FloatPoint1( const P : TPoint) : TFloatPoint;
8819:  Func FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8820:  Func FixedPoint( X, Y : Int) : TFixedPoint;
8821:  Func FixedPoint1( X, Y : Single) : TFixedPoint;
8822:  Func FixedPoint2( const P : TPoint) : TFixedPoint;
8823:  Func FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8824:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8825:  Func MakeRect( const L, T, R, B : Int) : TRect;
8826:  Func MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8827:  Func MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8828:  Func GFixedRect( const L, T, R, B : TFixed) : TRect;
8829:  Func FixedRect1( const ARect : TRect) : TRect;
8830:  Func FixedRect2( const FR : TFloatRect) : TRect;
8831:  Func GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8832:  Func FloatRect1( const ARect : TRect) : TFloatRect;
8833:  Func FloatRect2( const FXR : TRect) : TFloatRect;
8834:  Func GIntersectRect( out Dst : TRect; const R1, R2 : TRect) :Bool;
8835:  Func IntersectRect1(out Dst:TFloatRect;const FR1,FR2:TFloatRect):Boolean;
8836:  Func GUnionRect( out Rect : TRect; const R1, R2 : TRect) :Bool;
8837:  Func UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect):Boolean;
8838:  Func GEqualRect( const R1, R2 : TRect) :Bool;
8839:  Func EqualRect1( const R1, R2 : TFloatRect) :Bool;
8840:  Proc GInflateRect( var R : TRect; Dx, Dy : Int);
8841:  Proc InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8842:  Proc GOffsetRect( var R : TRect; Dx, Dy : Int);
8843:  Proc OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8844:  Func IsRectEmpty( const R : TRect) :Bool;
8845:  Func IsRectEmpty1( const FR : TFloatRect) :Bool;
8846:  Func GPtInRect( const R : TRect; const P : TPoint) :Bool;
8847:  Func PtInRect1( const R : TFloatRect; const P : TPoint) :Bool;
8848:  Func PtInRect2( const R : TRect; const P : TFloatPoint) :Bool;
8849:  Func PtInRect3( const R : TFloatRect; const P : TFloatPoint) :Bool;
8850:  Func EqualRectSize( const R1, R2 : TRect) :Bool;
8851:  Func EqualRectSize1( const R1, R2 : TFloatRect) :Bool;
8852:  Func MessageBeep( uType : UINT) : BOOL
8853:  Func ShowCursor( bShow : BOOL) : Int
8854:  Func SetCursorPos( X, Y : Int) : BOOL
8855:  Func SetCursor( hCursor : HICON) : HCURSOR
8856:  Func GetCursorPos( var lpPoint : TPoint) : BOOL
8857:  //Func ClipCursor( lpRect : PRect) : BOOL
8858:  Func GetClipCursor( var lpRect : TRect) : BOOL
8859:  Func GetCursor : HCURSOR
8860:  Func CreateCaret(hWnd: HWND;hBitmap:HBITMAP;nWidth, nHeight:Int):BOOL
8861:  Func GetCaretBlinkTime : UINT
8862:  Func SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8863:  Func DestroyCaret : BOOL
8864:  Func HideCaret( hWnd : HWND) : BOOL
8865:  Func ShowCaret( hWnd : HWND) : BOOL
8866:  Func SetCaretPos( X, Y : Int) : BOOL
8867:  Func GetCaretPos( var lpPoint : TPoint) : BOOL
8868:  Func ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8869:  Func ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8870:  Func MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints: UINT):Int
8871:  Func WindowFromPoint( Point : TPoint) : HWND
8872:  Func ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8873:
8874: Proc SIRegister_GR32_Math(CL: TPSPascalCompiler);
8875: begin
8876:  Func FixedFloor( A : TFixed) : Int
8877:  Func FixedCeil( A : TFixed) : Int
8878:  Func FixedMul( A, B : TFixed) : TFixed
8879:  Func FixedDiv( A, B : TFixed) : TFixed
8880:  Func OneOver( Value : TFixed) : TFixed
8881:  Func FixedRound( A : TFixed) : Int
8882:  Func FixedSqr( Value : TFixed) : TFixed
8883:  Func FixedSqrtLP( Value : TFixed) : TFixed
8884:  Func FixedSqrtHP( Value : TFixed) : TFixed
8885:  Func FixedCombine( W, X, Y : TFixed) : TFixed
8886:  Proc GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8887:  Proc GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8888:  Func GRHypot( const X, Y : TFloat) : TFloat;
8889:  Func Hypot1( const X, Y : Int) : Int;
8890:  Func FastSqrt( const Value : TFloat) : TFloat
8891:  Func FastSqrtBab1( const Value : TFloat) : TFloat
8892:  Func FastSqrtBab2( const Value : TFloat) : TFloat
```

```
8893:  Func FastInvSqrt( const Value : Single) : Single;
8894:  Func MulDiv( Multiplicand, Multiplier, Divisor : Int) : Int
8895:  Func GRIsPowerOf2( Value : Int) :Bool
8896:  Func PrevPowerOf2( Value : Int) : Int
8897:  Func NextPowerOf2( Value : Int) : Int
8898:  Func Average( A, B : Int) : Int
8899:  Func GRSign( Value : Int) : Int
8900:  Func FloatMod( x, y : Double) : Double
8901: end;
8902:
8903: Proc SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8904: begin
8905:  Func Clamp( const Value : Int) : Int;
8906:  Proc GRFillWord( var X, Count :Card; Value : Longword)
8907:  Func StackAlloc( Size : Int) : Pointer
8908:  Proc StackFree( P : Pointer)
8909:  Proc Swap( var A, B : Pointer);
8910:  Proc Swap1( var A, B : Int);
8911:  Proc Swap2( var A, B : TFixed);
8912:  Proc Swap3( var A, B : TColor32);
8913:  Proc TestSwap( var A, B : Int);
8914:  Proc TestSwap1( var A, B : TFixed);
8915:  Func TestClip( var A, B : Int; const Size : Int) :Bool;
8916:  Func TestClip1( var A, B : Int; const Start, Stop : Int) :Bool;
8917:  Func GRConstrain( const Value, Lo, Hi : Int) : Int;
8918:  Func Constrain1( const Value, Lo, Hi : Single) : Single;
8919:  Func SwapConstrain( const Value:Int; Constrain1,Constrain2:Int) : Int
8920:  Func GRMin( const A, B, C : Int) : Int;
8921:  Func GRMax( const A, B, C : Int) : Int;
8922:  Func Clamp( Value, Max : Int) : Int;
8923:  Func Clamp1( Value, Min, Max : Int) : Int;
8924:  Func Wrap( Value, Max : Int) : Int;
8925:  Func Wrap1( Value, Min, Max : Int) : Int;
8926:  Func Wrap3( Value, Max : Single) : Single;;
8927:  Func WrapPow2( Value, Max : Int) : Int;
8928:  Func WrapPow21( Value, Min, Max : Int) : Int;
8929:  Func Mirror( Value, Max : Int) : Int;
8930:  Func Mirror1( Value, Min, Max : Int) : Int;
8931:  Func MirrorPow2( Value, Max : Int) : Int;
8932:  Func MirrorPow21( Value, Min, Max : Int) : Int;
8933:  Func GetOptimalWrap( Max : Int) : TWrapProc;
8934:  Func GetOptimalWrap1( Min, Max : Int) : TWrapProcEx;
8935:  Func GetOptimalMirror( Max : Int) : TWrapProc;
8936:  Func GetOptimalMirror1( Min, Max : Int) : TWrapProcEx;
8937:  Func GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8938:  Func GetWrapProc1( WrapMode : TWrapMode; Max : Int) : TWrapProc;
8939:  Func GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8940:  Func GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Int):TWrapProcEx;
8941:  Func Div255( Value :Card) :Card
8942:  Func SAR_4( Value : Int) : Int
8943:  Func SAR_8( Value : Int) : Int
8944:  Func SAR_9( Value : Int) : Int
8945:  Func SAR_11( Value : Int) : Int
8946:  Func SAR_12( Value : Int) : Int
8947:  Func SAR_13( Value : Int) : Int
8948:  Func SAR_14( Value : Int) : Int
8949:  Func SAR_15( Value : Int) : Int
8950:  Func SAR_16( Value : Int) : Int
8951:  Func ColorSwap( WinColor : TColor) : TColor32
8952: end;
8953:
8954: Proc SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8955: begin
8956:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8957:  Proc CopyComponents(Dst,Src:TCustomBitmap32;Components:TColor32Components);
8958:  Proc CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8959:  Proc AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8960:  Proc ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha :Bool)
8961:  Proc IntensityToAlpha( Dst, Src : TCustomBitmap32)
8962:  Proc Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8963:  Proc InvertRGB( Dst, Src : TCustomBitmap32)
8964:  Proc ApplyLUT(Dst,Src:TCustomBitmap32;const LUT:TLUT8;PreserveAlpha:Boolean)
8965:  Proc ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8966:  Func CreateBitmask( Components : TColor32Components) : TColor32
8967:  Proc ApplyBitmask(Dst:TCustomBitmap32;DstX,DstY:Int;Src:TCustomBitmap32;SrcRect:TRect;
      Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8968:  Proc ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8969:  Proc CheckParams( Dst, Src : TCustomBitmap32; ResizeDst :Bool)
8970: end;
8971:
8972: Proc SIRegister_JclNTFS(CL: TPSPascalCompiler);
8973: begin
8974:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8975:   AddTypeS('TFileCompressionState','(fcNoCompression,fcDefaultCompression,fcLZNT1Compression )
8976:  Func NtfsGetCompression const FileName:str; var State : Short):Bool;
8977:  Func NtfsGetCompression1( const FileName :Str) : TFileCompressionState;
8978:  Func NtfsSetCompression const FileName:str; const State:Short) :Bool
8979:  Proc NtfsSetFileCompression(const FileName:Str; const State: TFileCompressionState)
```

```
8980:  Proc NtfsSetDirectoryTreeCompression(const Directory:Str; const State : TFileCompressionState)
8981:  Proc NtfsSetDefaultFileCompression(const Directory:Str; const State:TFileCompressionState)
8982:  Proc NtfsSetPathCompression(const Path:str;const State:TFileCompressionState;Recursive:Bool;
8983:  //AddTypeS('TNtfsAllocRanges', 'record Entries : Int; Data : PFileAlloca'
8984:  //+'tedRangeBuffer; MoreData :Bool; end
8985:  Func NtfsSetSparse( const FileName :Str) :Bool
8986:  Func NtfsZeroDataByHandle(const Handle:THandle;const First,Last:Int64): Bool
8987:  Func NtfsZeroDataByName(const FileName:str;const First,Last:Int64): Bool
8988:  //Func NtfsQueryAllocRanges(const FileName:str;Offset,Count:Int64;var Ranges:TNtfsAllocRanges):Boolean;
8989:  //Func NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges; Index:Int):TFileAllocatedRangeBuffer
8990:  Func NtfsSparseStreamsSupported( const Volume :Str) :Bool
8991:  Func NtfsGetSparse( const FileName :Str) :Bool
8992:  Func NtfsDeleteReparsePoint(const FileName:str; ReparseTag:DWORD):Boolean
8993:  Func NtfsSetReparsePoint(const FileName:str;var ReparseData,Size:Longword): Bool
8994: //Func NtfsGetReparsePoint(const FileName:str;var ReparseData:TReparseGuidDataBuffer):Bool
8995:  Func NtfsGetReparseTag( const Path :Str; var Tag : DWORD) :Bool
8996:  Func NtfsReparsePointsSupported( const Volume :Str) :Bool
8997:  Func NtfsFileHasReparsePoint( const Path :Str) :Bool
8998:  Func NtfsIsFolderMountPoint( const Path :Str) :Bool
8999:  Func NtfsMountDeviceAsDrive( const Device :Str; Drive : Char):Boolean
9000:  Func NtfsMountVolume(const Volume:Char; const MountPoint:str):Bool
9001:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
9002:  Func NtfsOpLockAckClosePending(Handle:THandle;Overlapped:TOverlapped):boolean
9003:  Func NtfsOpLockBreakAckNo2(Handle:THandle;Overlapped:TOverlapped):Bool
9004:  Func NtfsOpLockBreakAcknowledge(Handl:THandle;Overlapped:TOverlapped):Boolean
9005:  Func NtfsOpLockBreakNotify(Handle:THandle;Overlapped:TOverlapped) :Bool
9006:  Func NtfsRequestOpLock(Handle:THandle;Kind:TOpLock;Overlapped:TOverlapped): Bool
9007:  Func NtfsCreateJunctionPoint( const Source, Destination :Str) :Bool
9008:  Func NtfsDeleteJunctionPoint( const Source :Str) :Bool
9009:  Func NtfsGetJunctionPointDestination(const Source:Str;var Destination:str):Bool
9010:   AddTypeS('TStreamId','( siInvalid, siStandard, siExtendedAttribute, siSec'
9011:    +'urity, siAlternate,siHardLink,siProperty,siObjectIdentifier,siReparsePoints,siSparseFile )
9012:   AddTypeS('TStreamIds', 'set of TStreamId
9013:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
9014:    +': ___Pointer; StreamIds : TStreamIds; end
9015:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
9016:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
9017:  Func NtfsFindFirstStream(const FileName:str;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
9018:  Func NtfsFindNextStream( var Data : TFindStreamData) :Bool
9019:  Func NtfsFindStreamClose( var Data : TFindStreamData) :Bool
9020:  Func NtfsCreateHardLink(const LinkFileName,ExistingFileName:str):Boolean
9021:   AddTypeS('TNtfsHardLinkInfo','record LinkCount:Card;FileIndex:Int64; end
9022:  Func NtfsGetHardLinkInfo( const FileName :Str; var Info : TNtfsHardLinkInfo) :Bool
9023:  Func NtfsFindHardLinks(const Path:str;const FileIndexHigh,FileIndexLow:Card;const List:TStrings):Bool;
9024:  Func NtfsDeleteHardLinks( const FileName :Str) :Bool
9025:  Func JclAppInstances : TJclAppInstances;
9026:  Func JclAppInstances1( const UniqueAppIdGuidStr :Str) : TJclAppInstances;
9027:  Func ReadMessageCheck( var Message:TMessage;const IgnoredOriginatorWnd:HWND): TJclAppInstDataKind
9028:  Proc ReadMessageData(const Message:TMessage;var Data:___Pointer;var Size:Int)
9029:  Proc ReadMessageString( const Message : TMessage; var S :Str)
9030:  Proc ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
9031:
9032: (*----------------------------------------------------------------------------*)
9033: Proc SIRegister_JclGraphics(CL: TPSPascalCompiler);
9034: begin
9035:   FindClass('TOBJECT'),'EJclGraphicsError
9036:   TDynDynIntArrayArray', 'array of TDynIntArray
9037:   TDynPointArray', 'array of TPoint
9038:   TDynDynPointArrayArray', 'array of TDynPointArray
9039:   TPointF', 'record X : Single; Y : Single; end
9040:   TDynPointArrayF', 'array of TPointF
9041:   TDrawMode2', '( dmOpaque, dmBlend )
9042:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
9043:   TConversionKind', '(ckRed,ckGreen,ckBlue,ckAlpha ckUniformRGB, ckWeightedRGB )
9044:   TResamplingFilter','(rfBox,rfTriangle,rfHermite,rfBell,rfSpline,rfLanczos3,rfMitchell )
9045:   TMatrix3d', 'record array[0..2,0..2] of extended end
9046:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
9047:   TScanLine', 'array of Int
9048:   TScanLines', 'array of TScanLine
9049:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
9050:   TGradientDirection', '( gdVertical, gdHorizontal )
9051:   TPolyFillMode', '( fmAlternate, fmWinding )
9052:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
9053:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
9054:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
9055:   SIRegister_TJclDesktopCanvas(CL);
9056:   FindClass('TOBJECT'),'TJclRegion
9057:   SIRegister_TJclRegionInfo(CL);
9058:   SIRegister_TJclRegion(CL);
9059:   SIRegister_TJclThreadPersistent(CL);
9060:   SIRegister_TJclCustomMap(CL);
9061:   SIRegister_TJclBitmap32(CL);
9062:   SIRegister_TJclByteMap(CL);
9063:   SIRegister_TJclTransformation(CL);
9064:   SIRegister_TJclLinearTransformation(CL);
9065:  Proc Stretch(NewWidth,
       NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
9066:  Proc Stretch1(NewWidth,NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
9067:  Proc DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Int)
```

```
 Func GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
 Proc BitmapToJPeg( const FileName :Str)
 Proc JPegToBitmap( const FileName :Str)
 Func ExtractIconCount( const FileName :Str) : Int
 Func BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Int) : HICON
 Func IconToBitmapJ( Icon : HICON) : HBITMAP
 Proc BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
 Proc StretchTransfer(Dst:TJclBitmap32;DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter;CombineOp:TDrawMode)
 Proc Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
 Proc SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
 Func FillGradient(DC:HDC;ARect:TRect;ColorCount:Int;StartColor,EndColor:TColor;ADirection:TGradientDirection):Bool;
 Func CreateRegionFromBitmap(Bitmap:TBitmap;RegionColor:TColor;RegionBitmapMode:TJclRegionBitmapMode): HRGN
 Proc ScreenShot(bm:TBitmap; Left, Top, Width, Height : Int; Window : HWND);
 Proc ScreenShot1( bm : TBitmap);
 Proc PolyLineTS(Bitmap TJclBitmap32;const Points:TDynPointArray;Color: TColor32)
 Proc PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
 Proc PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
 Proc PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
 Proc PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
 Proc PolygonFS(Bitmap:TJclBitmap32;const Points:TDynPointArrayF;Color: TColor32)
 Proc PolyPolygonTS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
 Proc PolyPolygonAS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
 Proc PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Colr:TColr32);
 Proc AlphaToGrayscale( Dst, Src : TJclBitmap32)
 Proc IntensityToAlpha( Dst, Src : TJclBitmap32)
 Proc Invert( Dst, Src : TJclBitmap32)
 Proc InvertRGB( Dst, Src : TJclBitmap32)
 Proc ColorToGrayscale( Dst, Src : TJclBitmap32)
 Proc ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
 Proc SetGamma( Gamma : Single)
end;

(*------------------------------------------------------------------------------*)
Proc SIRegister_JclSynch(CL: TPSPascalCompiler);
begin
 Func LockedAdd( var Target : Int; Value : Int) : Int
 Func LockedCompareExchange( var Target : Int; Exch, Comp : Int) : Int;
 Func LockedCompareExchange1(var Target: ___Pointer;Exch,Comp:___Pointer): Pointer;
 Func LockedDec( var Target : Int) : Int
 Func LockedExchange( var Target : Int; Value : Int) : Int
 Func LockedExchangeAdd( var Target : Int; Value : Int) : Int
 Func LockedExchangeDec( var Target : Int) : Int
 Func LockedExchangeInc( var Target : Int) : Int
 Func LockedExchangeSub( var Target : Int; Value : Int) : Int
 Func LockedInc( var Target : Int) : Int
 Func LockedSub( var Target : Int; Value : Int) : Int
  TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
  SIRegister_TJclDispatcherObject(CL);
 Func WaitForMultipleObjects(const Objects:array of TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardin):Card;
 Func WaitAlertableForMultipleObjects(const Objects:array of TJclDispatcherObject;WaitAll:Bool;TimeOut:Card):Card
  SIRegister_TJclCriticalSection(CL);
  SIRegister_TJclCriticalSectionEx(CL);
  SIRegister_TJclEvent(CL);
  SIRegister_TJclWaitableTimer(CL);
  SIRegister_TJclSemaphore(CL);
  SIRegister_TJclMutex(CL);
  POptexSharedInfo', '^TOptexSharedInfo // will not work
  TOptexSharedInfo,record SpinCount:Int;LockCount:Int;ThreadId:Longword;RecursionCount:Int;end
  SIRegister_TJclOptex(CL);
  TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
  TMrewThreadInfo','record ThreadId:Longword;RecursionCount:Int;Reader:Bool; end
  TMrewThreadInfoArray', 'array of TMrewThreadInfo
  SIRegister_TJclMultiReadExclusiveWrite(CL);
  PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
  TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
   +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
  PMeteredSection', '^TMeteredSection // will not work
  TMeteredSection', 'record Event:THandle; FileMap:THandle; SharedInfo:PMetSectSharedInfo; end
  SIRegister_TJclMeteredSection(CL);
  TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
  TMutexInfo', 'record SignalState:Longint; Owned:Boolean; Abandoned:Bool; end
  TSemaphoreCounts', 'record CurrentCount: Longint; MaximumCount: Longint; end
  TTimerInfo', 'record Remaining : TLargeInt; Signaled : LongBool; end
 Func QueryCriticalSection(CS TJclCriticalSection;var Info:TRTLCriticalSection):Boolean
 Func QueryEvent( Handle : THandle; var Info : TEventInfo) :Bool
 Func QueryMutex( Handle : THandle; var Info : TMutexInfo) :Bool
 Func QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) :Bool
 Func QueryTimer( Handle : THandle; var Info : TTimerInfo) :Bool
  FindClass('TOBJECT'),'EJclWin32HandleObjectError
  FindClass('TOBJECT'),'EJclDispatcherObjectError
  FindClass('TOBJECT'),'EJclCriticalSectionError
  FindClass('TOBJECT'),'EJclEventError
  FindClass('TOBJECT'),'EJclWaitableTimerError
  FindClass('TOBJECT'),'EJclSemaphoreError
  FindClass('TOBJECT'),'EJclMutexError
```

```
9153:   FindClass('TOBJECT'),'EJclMeteredSectionError
9154: end;
9155:
9156: //***************************unit uPSI_mORMotReport;
9157: Proc SetCurrentPrinterAsDefault
9158: Func CurrentPrinterName :Str
9159: Func mCurrentPrinterPaperSize :Str
9160: Proc UseDefaultPrinter
9161:
9162: Proc SIRegisterTSTREAM(Cl: TPSPascalCompiler);
9163: begin
9164: with FindClass('TOBJECT'), 'TStream') do begin
9165:   IsAbstract := True;
9166:  //RegisterMethod('Func Read( var Buffer, Count : Longint) : Longint
9167:  // RegisterMethod('Func Write( const Buffer, Count : Longint) : Longint
9168:   Func Read(Buffer:str;Count:LongInt):LongInt
9169:   Func Write(Buffer:str;Count:LongInt):LongInt
9170:   Func ReadString(Buffer:str;Count:LongInt):LongInt     //FileStream
9171:   Func WriteString(Buffer:str;Count:LongInt):LongInt
9172:   Func ReadInt(Buffer:Int;Count:LongInt):LongInt
9173:   Func WriteInt(Buffer:Int;Count:LongInt):LongInt
9174:   Func ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt;
9175:   Func WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt;
9176:
9177:   Proc ReadAB(Buffer: TByteArray;Count:LongInt)
9178:   Proc WriteAB(Buffer: TByteArray;Count:LongInt)
9179:   Proc ReadABD(Buffer: TByteDynArray;Count:LongInt)
9180:   Proc WriteABD(Buffer: TByteDynArray;Count:LongInt)
9181:   Proc ReadAC(Buffer: TCharArray;Count:LongInt)
9182:   Proc WriteAC(Buffer: TCharArray;Count:LongInt)
9183:   Proc ReadACD(Buffer: TCharDynArray;Count:LongInt)
9184:   Proc WriteACD(Buffer: TCharDynArray;Count:LongInt)
9185:
9186:   Func Seek(Offset:LongInt;Origin:Word):LongInt
9187:   Proc ReadBuffer(Buffer:str;Count:LongInt)
9188:   Proc WriteBuffer(Buffer:str;Count:LongInt)
9189:   Proc ReadBufferInt(Buffer:Int;Count:LongInt);
9190:   Proc WriteBufferInt(Buffer:Int;Count:LongInt);
9191:   Proc ReadBufferFloat(Buffer:Double;Count:LongInt);
9192:   Proc WriteBufferFloat(Buffer:Double;Count:LongInt);
9193:
9194:   Proc ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9195:   Proc WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9196:   Proc ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9197:   Proc WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9198:   Proc ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9199:   Proc WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9200:   Proc ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9201:   Proc WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9202:   Proc ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9203:   Proc WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9204:   Proc ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9205:   Proc WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9206:   Proc ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9207:   Proc WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9208:
9209:   Proc ReadBufferP(Buffer: PChar;Count:LongInt)
9210:   Proc WriteBufferP(Buffer: PChar;Count:LongInt)
9211:   Proc ReadBufferO(Buffer: TObject;Count:LongInt);
9212:   Proc WriteBufferO(Buffer: TObject;Count:LongInt);
9213:   //READBUFFERAC
9214:   Func InstanceSize: Longint
9215:   Proc FixupResourceHeader( FixupInfo : Int)
9216:   Proc ReadResHeader
9217:   {$IFDEF DELPHI4UP}
9218:   Func CopyFrom(Source:TStream;Count:Int64):LongInt
9219:   {$ELSE}
9220:   Func CopyFrom(Source:TStream;Count:Int):LongInt
9221:   {$ENDIF}
9222:   RegisterProperty('Position', 'LongInt', iptrw);
9223:   RegisterProperty('Size', 'LongInt', iptrw);
9224: end;
9225: end;
9226:
9227: { ******************************************************************
9228:   Unit DMATH - Interface for DMATH.DLL
9229:   ****************************************************************** }
9230: // see more docs/dmath_manual.pdf
9231:
9232: Func InitEval : Int
9233: Proc SetVariable( VarName : Char; Value : Float)
9234: Proc SetFunction( FuncName :Str; Wrapper : TWrapper)
9235: Func Eval( ExpressionString :Str) : Float
9236:
9237: unit dmath; //types are in built, others are external in DLL
9238: interface
9239: {$IFDEF DELPHI}
9240: uses
9241:   StdCtrls, Graphics;
```

```
9242: {$ENDIF}
9243: { ----------------------------------------------------------------
9244:   Types and constants
9245:   ---------------------------------------------------------------- }
9246: {$i types.inc}
9247: { ----------------------------------------------------------------
9248:   Error handling
9249:   ---------------------------------------------------------------- }
9250: Proc SetErrCode(ErrCode : Int); external 'dmath';
9251: { Sets the error code }
9252: Func DefaultVal(ErrCode : Int; DefVal : Float) : Float; external 'dmath';
9253: { Sets error code and default Func value }
9254: Func MathErr : Int; external 'dmath';
9255: { Returns the error code }
9256: { ----------------------------------------------------------------
9257:   Dynamic arrays
9258:   ---------------------------------------------------------------- }
9259: Proc SetAutoInit(AutoInit :Bool); external 'dmath';
9260: { Sets the auto-initialization of arrays }
9261: Proc DimVector(var V : TVector; Ub : Int); external 'dmath';
9262: { Creates floating point vector V[0..Ub] }
9263: Proc DimIntVector(var V : TIntVector; Ub : Int); external 'dmath';
9264: { Creates Int vector V[0..Ub] }
9265: Proc DimCompVector(var V : TCompVector; Ub : Int);external 'dmath';
9266: { Creates complex vector V[0..Ub] }
9267: Proc DimBoolVector(var V : TBoolVector; Ub : Int);external 'dmath';
9268: { Creates boolean vector V[0..Ub] }
9269: Proc DimStrVector(var V : TStrVector; Ub : Int); external 'dmath';
9270: { Creates string vector V[0..Ub] }
9271: Proc DimMatrix(var A : TMatrix; Ub1, Ub2 : Int); external 'dmath';
9272: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9273: Proc DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Int); external 'dmath';
9274: { Creates Int matrix A[0..Ub1, 0..Ub2] }
9275: Proc DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Int);external 'dmath';
9276: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9277: Proc DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Int);external 'dmath';
9278: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9279: Proc DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Int); external 'dmath';
9280: { Creates string matrix A[0..Ub1, 0..Ub2] }
9281: { ----------------------------------------------------------------
9282:   Minimum, maximum, sign and exchange
9283:   ---------------------------------------------------------------- }
9284: Func FMin(X, Y : Float) : Float; external 'dmath';
9285: { Minimum of 2 reals }
9286: Func FMax(X, Y : Float) : Float; external 'dmath';
9287: { Maximum of 2 reals }
9288: Func IMin(X, Y : Int) : Int; external 'dmath';
9289: { Minimum of 2 Ints }
9290: Func IMax(X, Y : Int) : Int; external 'dmath';
9291: { Maximum of 2 Ints }
9292: Func Sgn(X : Float) : Int; external 'dmath';
9293: { Sign (returns 1 if X = 0) }
9294: Func Sgn0(X : Float) : Int; external 'dmath';
9295: { Sign (returns 0 if X = 0) }
9296: Func DSgn(A, B : Float) : Float; external 'dmath';
9297: { Sgn(B) * |A| }
9298: Proc FSwap(var X, Y : Float); external 'dmath';
9299: { Exchange 2 reals }
9300: Proc ISwap(var X, Y : Int); external 'dmath';
9301: { Exchange 2 Ints }
9302: { ----------------------------------------------------------------
9303:   Rounding functions
9304:   ---------------------------------------------------------------- }
9305: Func RoundN(X : Float; N : Int) : Float; external 'dmath';
9306: { Rounds X to N decimal places }
9307: Func Ceil(X : Float) : Int; external 'dmath';
9308: { Ceiling Func }
9309: Func Floor(X : Float) : Int; external 'dmath';
9310: { Floor Func }
9311: { ----------------------------------------------------------------
9312:   Logarithms, exponentials and power
9313:   ---------------------------------------------------------------- }
9314: Func Expo(X : Float) : Float; external 'dmath';
9315: { Exponential }
9316: Func Exp2(X : Float) : Float; external 'dmath';
9317: { 2^X }
9318: Func Exp10(X : Float) : Float; external 'dmath';
9319: { 10^X }
9320: Func Log(X : Float) : Float; external 'dmath';
9321: { Natural log }
9322: Func Log2(X : Float) : Float; external 'dmath';
9323: { Log, base 2 }
9324: Func Log10(X : Float) : Float; external 'dmath';
9325: { Decimal log }
9326: Func LogA(X, A : Float) : Float; external 'dmath';
9327: { Log, base A }
9328: Func IntPower(X : Float; N : Int) : Float; external 'dmath';
9329: { X^N }
9330: Func Power(X, Y : Float) : Float; external 'dmath';
```

```
9331: { X^Y, X >= 0 }
9332: { ----------------------------------------------------------------
9333:   Trigonometric functions
9334:   ---------------------------------------------------------------- }
9335: Func Pythag(X, Y : Float) : Float; external 'dmath';
9336: { Sqrt(X^2 + Y^2) }
9337: Func FixAngle(Theta : Float) : Float; external 'dmath';
9338: { Set Theta in -Pi..Pi }
9339: Func Tan(X : Float) : Float; external 'dmath';
9340: { Tangent }
9341: Func ArcSin(X : Float) : Float; external 'dmath';
9342: { Arc sinus }
9343: Func ArcCos(X : Float) : Float; external 'dmath';
9344: { Arc cosinus }
9345: Func ArcTan2(Y, X : Float) : Float; external 'dmath';
9346: { Angle (Ox, OM) with M(X,Y) }
9347: { ----------------------------------------------------------------
9348:   Hyperbolic functions
9349:   ---------------------------------------------------------------- }
9350: Func Sinh(X : Float) : Float; external 'dmath';
9351: { Hyperbolic sine }
9352: Func Cosh(X : Float) : Float; external 'dmath';
9353: { Hyperbolic cosine }
9354: Func Tanh(X : Float) : Float; external 'dmath';
9355: { Hyperbolic tangent }
9356: Func ArcSinh(X : Float) : Float; external 'dmath';
9357: { Inverse hyperbolic sine }
9358: Func ArcCosh(X : Float) : Float; external 'dmath';
9359: { Inverse hyperbolic cosine }
9360: Func ArcTanh(X : Float) : Float; external 'dmath';
9361: { Inverse hyperbolic tangent }
9362: Proc SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9363: { Sinh & Cosh }
9364: { ----------------------------------------------------------------
9365:   Gamma Func and related functions
9366:   ---------------------------------------------------------------- }
9367: Func Fact(N : Int) : Float; external 'dmath';
9368: { Factorial }
9369: Func SgnGamma(X : Float) : Int; external 'dmath';
9370: { Sign of Gamma Func }
9371: Func Gamma(X : Float) : Float; external 'dmath';
9372: { Gamma Func }
9373: Func LnGamma(X : Float) : Float; external 'dmath';
9374: { Logarithm of Gamma Func }
9375: Func Stirling(X : Float) : Float; external 'dmath';
9376: { Stirling's formula for the Gamma Func }
9377: Func StirLog(X : Float) : Float; external 'dmath';
9378: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9379: Func DiGamma(X : Float ) : Float; external 'dmath';
9380: { Digamma Func }
9381: Func TriGamma(X : Float ) : Float; external 'dmath';
9382: { Trigamma Func }
9383: Func IGamma(A, X : Float) : Float; external 'dmath';
9384: { Incomplete Gamma function}
9385: Func JGamma(A, X : Float) : Float; external 'dmath';
9386: { Complement of incomplete Gamma Func }
9387: Func InvGamma(A, P : Float) : Float; external 'dmath';
9388: { Inverse of incomplete Gamma Func }
9389: Func Erf(X : Float) : Float; external 'dmath';
9390: { Error Func }
9391: Func Erfc(X : Float) : Float; external 'dmath';
9392: { Complement of error Func }
9393: { ----------------------------------------------------------------
9394:   Beta Func and related functions
9395:   ---------------------------------------------------------------- }
9396: Func Beta(X, Y : Float) : Float; external 'dmath';
9397: { Beta Func }
9398: Func IBeta(A, B, X : Float) : Float; external 'dmath';
9399: { Incomplete Beta Func }
9400: Func InvBeta(A, B, Y : Float) : Float; external 'dmath';
9401: { Inverse of incomplete Beta Func }
9402: { ----------------------------------------------------------------
9403:   Lambert's function
9404:   ---------------------------------------------------------------- }
9405: Func LambertW(X : Float; UBranch, Offset :Bool) : Float; external 'dmath';
9406:  ----------------------------------------------------------------
9407:   Binomial distribution
9408:   ---------------------------------------------------------------- }
9409: Func Binomial(N, K : Int) : Float; external 'dmath';
9410: { Binomial coefficient C(N,K) }
9411: Func PBinom(N : Int; P : Float; K : Int) : Float; external 'dmath';
9412: { Probability of binomial distribution }
9413: Func FBinom(N : Int; P : Float; K : Int) : Float; external 'dmath';
9414: { Cumulative probability for binomial distrib. }
9415: { ----------------------------------------------------------------
9416:   Poisson distribution
9417:   ---------------------------------------------------------------- }
9418: Func PPoisson(Mu : Float; K : Int) : Float; external 'dmath';
9419: { Probability of Poisson distribution }
```

```
9420: Func FPoisson(Mu : Float; K : Int) : Float; external 'dmath';
9421: { Cumulative probability for Poisson distrib. }
9422: { ----------------------------------------------------------------
9423:   Exponential distribution
9424:   ---------------------------------------------------------------- }
9425: Func DExpo(A, X : Float) : Float; external 'dmath';
9426: { Density of exponential distribution with parameter A }
9427: Func FExpo(A, X : Float) : Float; external 'dmath';
9428: { Cumulative probability Func for exponential dist. with parameter A }
9429: { ----------------------------------------------------------------
9430:   Standard normal distribution
9431:   ---------------------------------------------------------------- }
9432: Func DNorm(X : Float) : Float; external 'dmath';
9433: { Density of standard normal distribution }
9434: Func FNorm(X : Float) : Float; external 'dmath';
9435: { Cumulative probability for standard normal distrib. }
9436: Func PNorm(X : Float) : Float; external 'dmath';
9437: { Prob(|U| > X) for standard normal distrib. }
9438: Func InvNorm(P : Float) : Float; external 'dmath';
9439: { Inverse of standard normal distribution }
9440: { ----------------------------------------------------------------
9441:   Student's distribution
9442:   ---------------------------------------------------------------- }
9443: Func DStudent(Nu : Int; X : Float) : Float; external 'dmath';
9444: { Density of Student distribution with Nu d.o.f. }
9445: Func FStudent(Nu : Int; X : Float) : Float; external 'dmath';
9446: { Cumulative probability for Student distrib. with Nu d.o.f. }
9447: Func PStudent(Nu : Int; X : Float) : Float; external 'dmath';
9448: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9449: Func InvStudent(Nu : Int; P : Float) : Float; external 'dmath';
9450: { Inverse of Student's t-distribution Func }
9451: { ----------------------------------------------------------------
9452:   Khi-2 distribution
9453:   ---------------------------------------------------------------- }
9454: Func DKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9455: { Density of Khi-2 distribution with Nu d.o.f. }
9456: Func FKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9457: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9458: Func PKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9459: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9460: Func InvKhi2(Nu : Int; P : Float) : Float; external 'dmath';
9461: { Inverse of Khi-2 distribution Func }
9462: { ----------------------------------------------------------------
9463:   Fisher-Snedecor distribution
9464:   ---------------------------------------------------------------- }
9465: Func DSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9466: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9467: Func FSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9468: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9469: Func PSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9470: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9471: Func InvSnedecor(Nu1, Nu2 : Int; P : Float) : Float; external 'dmath';
9472: { Inverse of Snedecor's F-distribution Func }
9473: { ----------------------------------------------------------------
9474:   Beta distribution
9475:   ---------------------------------------------------------------- }
9476: Func DBeta(A, B, X : Float) : Float; external 'dmath';
9477: { Density of Beta distribution with parameters A and B }
9478: Func FBeta(A, B, X : Float) : Float; external 'dmath';
9479: { Cumulative probability for Beta distrib. with param. A and B }
9480: { ----------------------------------------------------------------
9481:   Gamma distribution
9482:   ---------------------------------------------------------------- }
9483: Func DGamma(A, B, X : Float) : Float; external 'dmath';
9484: { Density of Gamma distribution with parameters A and B }
9485: Func FGamma(A, B, X : Float) : Float; external 'dmath';
9486: { Cumulative probability for Gamma distrib. with param. A and B }
9487: { ----------------------------------------------------------------
9488:   Expression evaluation
9489:   ---------------------------------------------------------------- }
9490: Func InitEval : Int; external 'dmath';
9491: { Initializes built-in functions and returns their number }
9492: Func Eval(ExpressionString :Str) : Float; external 'dmath';
9493: { Evaluates an expression at run-time }
9494: Proc SetVariable(VarName : Char; Value : Float); external 'dmath';
9495: { Assigns a value to a variable }
9496: Proc SetFunction(FuncName :Str; Wrapper : TWrapper); external 'dmath';
9497: { Adds a Func to the parser }
9498: { ----------------------------------------------------------------
9499:   Matrices and linear equations
9500:   ---------------------------------------------------------------- }
9501: Proc GaussJordan(A            : TMatrix;
9502:                       Lb, Ub1, Ub2 : Int;
9503:                       var Det      : Float); external 'dmath';
9504: { Transforms a matrix according to the Gauss-Jordan method }
9505: Proc LinEq(A       : TMatrix;
9506:                B       : TVector;
9507:                Lb, Ub  : Int;
9508:                var Det : Float); external 'dmath';
```

```
9509: { Solves a linear system according to the Gauss-Jordan method }
9510: Proc Cholesky(A, L : TMatrix; Lb, Ub : Int); external 'dmath';
9511: { Cholesky factorization of a positive definite symmetric matrix }
9512: Proc LU_Decomp(A : TMatrix; Lb, Ub : Int); external 'dmath';
9513: { LU decomposition }
9514: Proc LU_Solve(A      : TMatrix;
9515:                   B      : TVector;
9516:                   Lb, Ub : Int;
9517:                   X      : TVector); external 'dmath';
9518: { Solution of linear system from LU decomposition }
9519: Proc QR_Decomp(A            : TMatrix;
9520:                    Lb, Ub1, Ub2 : Int;
9521:                    R            : TMatrix); external 'dmath';
9522: { QR decomposition }
9523: Proc QR_Solve(Q, R         : TMatrix;
9524:                   B            : TVector;
9525:                   Lb, Ub1, Ub2 : Int;
9526:                   X            : TVector); external 'dmath';
9527: { Solution of linear system from QR decomposition }
9528: Proc SV_Decomp(A            : TMatrix;
9529:                    Lb, Ub1, Ub2 : Int;
9530:                    S            : TVector;
9531:                    V            : TMatrix); external 'dmath';
9532: { Singular value decomposition }
9533: Proc SV_SetZero(S      : TVector;
9534:                    Lb, Ub : Int;
9535:                    Tol    : Float); external 'dmath';
9536: { Set lowest singular values to zero }
9537: Proc SV_Solve(U            : TMatrix;
9538:                   S            : TVector;
9539:                   V            : TMatrix;
9540:                   B            : TVector;
9541:                   Lb, Ub1, Ub2 : Int;
9542:                   X            : TVector); external 'dmath';
9543: { Solution of linear system from SVD }
9544: Proc SV_Approx(U            : TMatrix;
9545:                    S            : TVector;
9546:                    V            : TMatrix;
9547:                    Lb, Ub1, Ub2 : Int;
9548:                    A            : TMatrix); external 'dmath';
9549: { Matrix approximation from SVD }
9550: Proc EigenVals(A      : TMatrix;
9551:                    Lb, Ub : Int;
9552:                    Lambda : TCompVector); external 'dmath';
9553: { Eigenvalues of a general square matrix }
9554: Proc EigenVect(A      : TMatrix;
9555:                    Lb, Ub : Int;
9556:                    Lambda : TCompVector;
9557:                    V      : TMatrix); external 'dmath';
9558: { Eigenvalues and eigenvectors of a general square matrix }
9559: Proc EigenSym(A      : TMatrix;
9560:                   Lb, Ub : Int;
9561:                   Lambda : TVector;
9562:                   V      : TMatrix); external 'dmath';
9563: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9564: Proc Jacobi(A                : TMatrix;
9565:                 Lb, Ub, MaxIter : Int;
9566:                 Tol             : Float;
9567:                 Lambda          : TVector;
9568:                 V               : TMatrix); external 'dmath';
9569: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9570: { ----------------------------------------------------------------
9571:   Optimization
9572:   ---------------------------------------------------------------- }
9573: Proc MinBrack(Func                    : TFunc;
9574:                   var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9575: { Brackets a minimum of a Func }
9576: Proc GoldSearch(Func           : TFunc;
9577:                    A, B           : Float;
9578:                    MaxIter        : Int;
9579:                    Tol            : Float;
9580:                    var Xmin, Ymin : Float); external 'dmath';
9581: { Minimization of a Func of one variable (golden search) }
9582: Proc LinMin(Func      : TFuncNVar;
9583:                 X, DeltaX : TVector;
9584:                 Lb, Ub    : Int;
9585:                 var R     : Float;
9586:                 MaxIter   : Int;
9587:                 Tol       : Float;
9588:                 var F_min : Float); external 'dmath';
9589: { Minimization of a Func of several variables along a line }
9590: Proc Newton(Func      : TFuncNVar;
9591:                 HessGrad  : THessGrad;
9592:                 X         : TVector;
9593:                 Lb, Ub    : Int;
9594:                 MaxIter   : Int;
9595:                 Tol       : Float;
9596:                 var F_min : Float;
9597:                 G         : TVector;
```

```
9598:                 H_inv     : TMatrix;
9599:                 var Det   : Float); external 'dmath';
9600: { Minimization of a Func of several variables (Newton's method) }
9601: Proc SaveNewton(FileName :Str); external 'dmath';
9602: { Save Newton iterations in a file }
9603: Proc Marquardt(Func      : TFuncNVar;
9604:                    HessGrad  : THessGrad;
9605:                    X         : TVector;
9606:                    Lb, Ub    : Int;
9607:                    MaxIter   : Int;
9608:                    Tol       : Float;
9609:                    var F_min : Float;
9610:                    G         : TVector;
9611:                    H_inv     : TMatrix;
9612:                    var Det   : Float); external 'dmath';
9613: { Minimization of a Func of several variables (Marquardt's method) }
9614: Proc SaveMarquardt(FileName :Str); external 'dmath';
9615: { Save Marquardt iterations in a file }
9616: Proc BFGS(Func      : TFuncNVar;
9617:               Gradient  : TGradient;
9618:               X         : TVector;
9619:               Lb, Ub    : Int;
9620:               MaxIter   : Int;
9621:               Tol       : Float;
9622:               var F_min : Float;
9623:               G         : TVector;
9624:               H_inv     : TMatrix); external 'dmath';
9625: { Minimization of a Func of several variables (BFGS method) }
9626: Proc SaveBFGS(FileName :Str); external 'dmath';
9627: { Save BFGS iterations in a file }
9628: Proc Simplex(Func      : TFuncNVar;
9629:                  X         : TVector;
9630:                  Lb, Ub    : Int;
9631:                  MaxIter   : Int;
9632:                  Tol       : Float;
9633:                  var F_min : Float); external 'dmath';
9634: { Minimization of a Func of several variables (Simplex) }
9635: Proc SaveSimplex(FileName :Str); external 'dmath';
9636: { Save Simplex iterations in a file }
9637: { ----------------------------------------------------------------
9638:   Nonlinear equations
9639:   ---------------------------------------------------------------- }
9640: Proc RootBrack(Func             : TFunc;
9641:                    var X, Y, FX, FY : Float); external 'dmath';
9642: { Brackets a root of Func Func between X and Y }
9643: Proc Bisect(Func     : TFunc;
9644:                 var X, Y : Float;
9645:                 MaxIter  : Int;
9646:                 Tol      : Float;
9647:                 var F    : Float); external 'dmath';
9648: { Bisection method }
9649: Proc Secant(Func     : TFunc;
9650:                 var X, Y : Float;
9651:                 MaxIter  : Int;
9652:                 Tol      : Float;
9653:                 var F    : Float); external 'dmath';
9654: { Secant method }
9655: Proc NewtEq(Func, Deriv : TFunc;
9656:                 var X       : Float;
9657:                 MaxIter     : Int;
9658:                 Tol         : Float;
9659:                 var F       : Float); external 'dmath';
9660: { Newton-Raphson method for a single nonlinear equation }
9661: Proc NewtEqs(Equations : TEquations;
9662:                  Jacobian  : TJacobian;
9663:                  X, F      : TVector;
9664:                  Lb, Ub    : Int;
9665:                  MaxIter   : Int;
9666:                  Tol       : Float); external 'dmath';
9667: { Newton-Raphson method for a system of nonlinear equations }
9668: Proc Broyden(Equations : TEquations;
9669:                  X, F      : TVector;
9670:                  Lb, Ub    : Int;
9671:                  MaxIter   : Int;
9672:                  Tol       : Float); external 'dmath';
9673: { Broyden's method for a system of nonlinear equations }
9674: { ----------------------------------------------------------------
9675:   Polynomials and rational fractions
9676:   ---------------------------------------------------------------- }
9677: Func Poly(X    : Float;
9678:              Coef : TVector;
9679:              Deg  : Int) : Float; external 'dmath';
9680: { Evaluates a polynomial }
9681: Func RFrac(X          : Float;
9682:               Coef       : TVector;
9683:               Deg1, Deg2 : Int) : Float; external 'dmath';
9684: { Evaluates a rational fraction }
9685: Func RootPoll(A, B  : Float;
9686:                  var X : Float) : Int; external 'dmath';
```

```
9687: { Solves the linear equation A + B * X = 0 }
9688: Func RootPol2(Coef : TVector;
9689:                   Z    : TCompVector) : Int; external 'dmath';
9690: { Solves a quadratic equation }
9691: Func RootPol3(Coef : TVector;
9692:                   Z    : TCompVector) : Int; external 'dmath';
9693: { Solves a cubic equation }
9694: Func RootPol4(Coef : TVector;
9695:                  Z    : TCompVector) : Int; external 'dmath';
9696: { Solves a quartic equation }
9697: Func RootPol(Coef : TVector;
9698:                 Deg  : Int;
9699:                 Z    : TCompVector) : Int; external 'dmath';
9700: { Solves a polynomial equation }
9701: Func SetRealRoots(Deg : Int;
9702:                       Z   : TCompVector;
9703:                       Tol : Float) : Int; external 'dmath';
9704: { Set the imaginary part of a root to zero }
9705: Proc SortRoots(Deg : Int;
9706:                    Z   : TCompVector); external 'dmath';
9707: { Sorts the roots of a polynomial }
9708: { ----------------------------------------------------------------
9709:   Numerical integration and differential equations
9710:   ---------------------------------------------------------------- }
9711: Func TrapInt(X, Y : TVector; N : Int) : Float; external 'dmath';
9712: { Integration by trapezoidal rule }
9713: Func GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9714: { Integral from A to B }
9715: Func GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9716: { Integral from 0 to B }
9717: Func Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9718: { Convolution product at time T }
9719: Proc ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Int);external 'dmath';
9720: { Convolution by trapezoidal rule }
9721: Proc RKF45(F                    : TDiffEqs;
9722:                Neqn                 : Int;
9723:                Y, Yp                : TVector;
9724:                var T                : Float;
9725:                Tout, RelErr, AbsErr : Float;
9726:                var Flag             : Int); external 'dmath';
9727: { Integration of a system of differential equations }
9728: { ----------------------------------------------------------------
9729:   Fast Fourier Transform
9730:   ---------------------------------------------------------------- }
9731: Proc FFT(NumSamples        : Int;
9732:              InArray, OutArray : TCompVector); external 'dmath';
9733: { Fast Fourier Transform }
9734: Proc IFFT(NumSamples        : Int;
9735:               InArray, OutArray : TCompVector); external 'dmath';
9736: { Inverse Fast Fourier Transform }
9737: Proc FFT_Int(NumSamples     : Int;
9738:                       RealIn, ImagIn : TIntVector;
9739:                       OutArray       : TCompVector); external 'dmath';
9740: { Fast Fourier Transform for Int data }
9741: Proc FFT_Int_Cleanup; external 'dmath';
9742: { Clear memory after a call to FFT_Int }
9743: Proc CalcFrequency(NumSamples,
9744:                        FrequencyIndex : Int;
9745:                        InArray        : TCompVector;
9746:                        var FFT        : Complex); external 'dmath';
9747: { Direct computation of Fourier transform }
9748: { ----------------------------------------------------------------
9749:   Random numbers
9750:   ---------------------------------------------------------------- }
9751: Proc SetRNG(RNG : RNG_Type); external 'dmath';
9752: { Select generator }
9753: Proc InitGen(Seed : RNG_IntType); external 'dmath';
9754: { Initialize generator }
9755: Func IRanGen : RNG_IntType; external 'dmath';
9756: { 32-bit random Int in [-2^31 .. 2^31 - 1] }
9757: Func IRanGen31 : RNG_IntType; external 'dmath';
9758: { 31-bit random Int in [0 .. 2^31 - 1] }
9759: Func RanGen1 : Float; external 'dmath';
9760: { 32-bit random real in [0,1] }
9761: Func RanGen2 : Float; external 'dmath';
9762: { 32-bit random real in [0,1) }
9763: Func RanGen3 : Float; external 'dmath';
9764: { 32-bit random real in (0,1) }
9765: Func RanGen53 : Float; external 'dmath';
9766: { 53-bit random real in [0,1) }
9767: Proc InitMWC(Seed : RNG_IntType); external 'dmath';
9768: { Initializes the 'Multiply with carry' random number generator }
9769: Func IRanMWC : RNG_IntType; external 'dmath';
9770: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9771: Proc InitMT(Seed : RNG_IntType); external 'dmath';
9772: { Initializes Mersenne Twister generator with a seed }
9773: Proc InitMTbyArray(InitKey   : array of RNG_LongType;
9774:                        KeyLength : Word); external 'dmath';
9775: { 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';
{ Random number from normal distrib. with mean Mu and S. D. Sigma }
Proc RanMult(M      : TVector; L      : TMatrix;
                  Lb, Ub : Int;
                  X      : TVector); external 'dmath';
{ Random vector from multinormal distribution (correlated) }
Proc RanMultIndep(M, S   : TVector;
                       Lb, Ub : Int;
                       X      : TVector); external 'dmath';
{ Random vector from multinormal distribution (uncorrelated) }
Proc InitMHParams(NCycles, MaxSim, SavedSim : Int); external 'dmath';
{ Initializes Metropolis-Hastings parameters }
Proc GetMHParams(var NCycles, MaxSim,SavedSim:Int); external 'dmath';
{ Returns Metropolis-Hastings parameters }
Proc Hastings(Func      : TFuncNVar;
                   T         : Float;
                   X         : TVector;
                   V         : TMatrix;
                   Lb, Ub    : Int;
                   Xmat      : TMatrix;
                   X_min     : TVector;
                   var F_min : Float); external 'dmath';
{ Simulation of a probability density Func by Metropolis-Hastings }
Proc InitSAParams(NT, NS, NCycles : Int; RT : Float); external 'dmath';
{ Initializes Simulated Annealing parameters }
Proc SA_CreateLogFile(FileName :Str); external 'dmath';
{ Initializes log file }
Proc SimAnn(Func          : TFuncNVar;
                 X, Xmin, Xmax : TVector;
                 Lb, Ub        : Int;
                 var F_min     : Float); external 'dmath';
{ Minimization of a Func of several var. by simulated annealing }
Proc InitGAParams(NP, NG : Int; SR, MR, HR : Float); external 'dmath';
{ Initializes Genetic Algorithm parameters }
Proc GA_CreateLogFile(FileName :Str); external 'dmath';
{ Initializes log file }
Proc GenAlg(Func          : TFuncNVar;
                 X, Xmin, Xmax : TVector;
                 Lb, Ub        : Int;
                 var F_min     : Float); external 'dmath';
{ Minimization of a Func of several var. by genetic algorithm }
{ ----------------------------------------------------------------
  Statistics
  ---------------------------------------------------------------- }
Func Mean(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
{ Mean of sample X }
Func Min(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
{ Minimum of sample X }
Func Max(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
{ Maximum of sample X }
Func Median(X : TVector; Lb, Ub : Int; Sorted :Bool) : Float; external 'dmath';
{ Median of sample X }
Func StDev(X : TVector; Lb, Ub : Int; M : Float) : Float; external 'dmath';
{ Standard deviation estimated from sample X }
Func StDevP(X : TVector; Lb, Ub : Int; M : Float) : Float; external 'dmath';
{ Standard deviation of population }
Func Correl(X, Y : TVector; Lb, Ub : Int) : Float; external 'dmath';
{ Correlation coefficient }
Func Skewness(X : TVector; Lb, Ub : Int; M,Sigma: Float): Float; external 'dmath';
{ Skewness of sample X }
Func Kurtosis(X : TVector; Lb, Ub : Int; M,Sigma: Float): Float; external 'dmath';
{ Kurtosis of sample X }
Proc QSort(X : TVector; Lb, Ub : Int); external 'dmath';
{ Quick sort (ascending order) }
Proc DQSort(X : TVector; Lb, Ub : Int); external 'dmath';
{ Quick sort (descending order) }
Proc Interval(X1, X2            : Float;
                   MinDiv, MaxDiv     : Int;
                   var Min, Max, Step : Float); external 'dmath';
{ Determines an interval for a set of values }
Proc AutoScale(X : TVector; Lb, Ub : Int; Scale : TScale;
                    var XMin, XMax, XStep : Float); external 'dmath';
{ Finds an appropriate scale for plotting the data in X[Lb..Ub] }
Proc StudIndep(N1, N2         : Int;
                    M1, M2, S1, S2 : Float;
                    var T          : Float;
                    var DoF        : Int); external 'dmath';
{ Student t-test for independent samples }
Proc StudPaired(X, Y    : TVector;
```

```
                      Lb, Ub   : Int;
                      var T    : Float;
                      var DoF  : Int); external 'dmath';
{ Student t-test for paired samples }
Proc AnOVa1(Ns                : Int;
                  N                 : TIntVector;
                  M, S              : TVector;
                  var V_f, V_r, F   : Float;
                  var DoF_f, DoF_r  : Int); external 'dmath';
{ One-way analysis of variance }
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;
```

```
 9954:                     Lb, Ub : Int;
 9955:                     SVDTol : Float;
 9956:                     B      : TVector;
 9957:                     V      : TMatrix); external 'dmath';
 9958: { Unweighted linear regression by singular value decomposition }
 9959: Proc WSVDLinFit(X, Y, S : TVector;
 9960:                      Lb, Ub  : Int;
 9961:                      SVDTol  : Float;
 9962:                      B       : TVector;
 9963:                      V       : TMatrix); external 'dmath';
 9964: { Weighted linear regression by singular value decomposition }
 9965: Proc MulFit(X            : TMatrix;
 9966:                 Y            : TVector;
 9967:                 Lb, Ub, Nvar : Int;
 9968:                 ConsTerm     :Bool;
 9969:                 B            : TVector;
 9970:                 V            : TMatrix); external 'dmath';
 9971: { Multiple linear regression by Gauss-Jordan method }
 9972: Proc WMulFit(X            : TMatrix;
 9973:                  Y, S         : TVector;
 9974:                  Lb, Ub, Nvar : Int;
 9975:                  ConsTerm     :Bool;
 9976:                  B            : TVector;
 9977:                  V            : TMatrix); external 'dmath';
 9978: { Weighted multiple linear regression by Gauss-Jordan method }
 9979: Proc SVDFit(X            : TMatrix;
 9980:                 Y            : TVector;
 9981:                 Lb, Ub, Nvar : Int;
 9982:                 ConsTerm     :Bool;
 9983:                 SVDTol       : Float;
 9984:                 B            : TVector;
 9985:                 V            : TMatrix); external 'dmath';
 9986: { Multiple linear regression by singular value decomposition }
 9987: Proc WSVDFit(X            : TMatrix;
 9988:                  Y, S         : TVector;
 9989:                  Lb, Ub, Nvar : Int;
 9990:                  ConsTerm     :Bool;
 9991:                  SVDTol       : Float;
 9992:                  B            : TVector;
 9993:                  V            : TMatrix); external 'dmath';
 9994: { Weighted multiple linear regression by singular value decomposition }
 9995: Proc PolFit(X, Y        : TVector;
 9996:                 Lb, Ub, Deg : Int;
 9997:                 B           : TVector;
 9998:                 V           : TMatrix); external 'dmath';
 9999: { Polynomial regression by Gauss-Jordan method }
10000: Proc WPolFit(X, Y, S     : TVector;
10001:                  Lb, Ub, Deg : Int;
10002:                  B           : TVector;
10003:                  V           : TMatrix); external 'dmath';
10004: { Weighted polynomial regression by Gauss-Jordan method }
10005: Proc SVDPolFit(X, Y        : TVector;
10006:                    Lb, Ub, Deg : Int;
10007:                    SVDTol      : Float;
10008:                    B           : TVector;
10009:                    V           : TMatrix); external 'dmath';
10010: { Unweighted polynomial regression by singular value decomposition }
10011: Proc WSVDPolFit(X, Y, S     : TVector;
10012:                     Lb, Ub, Deg : Int;
10013:                     SVDTol      : Float;
10014:                     B           : TVector;
10015:                     V           : TMatrix); external 'dmath';
10016: { Weighted polynomial regression by singular value decomposition }
10017: Proc RegTest(Y, Ycalc : TVector;
10018:                  LbY, UbY : Int;
10019:                  V        : TMatrix;
10020:                  LbV, UbV : Int;
10021:                  var Test : TRegTest); external 'dmath';
10022: { Test of unweighted regression }
10023: Proc WRegTest(Y, Ycalc, S : TVector;
10024:                   LbY, UbY    : Int;
10025:                   V           : TMatrix;
10026:                   LbV, UbV    : Int;
10027:                   var Test    : TRegTest); external 'dmath';
10028: { Test of weighted regression }
10029: { ----------------------------------------------------------------
10030:   Nonlinear regression
10031:   ---------------------------------------------------------------- }
10032: Proc SetOptAlgo(Algo : TOptAlgo); external 'dmath';
10033: { Sets the optimization algorithm for nonlinear regression }
10034: Func GetOptAlgo : TOptAlgo; external 'dmath';
10035: { Returns the optimization algorithm }
10036: Proc SetMaxParam(N : Byte); external 'dmath';
10037: { Sets the maximum number of regression parameters for nonlinear regression }
10038: Func GetMaxParam : Byte; external 'dmath';
10039: { Returns the maximum number of regression parameters for nonlinear regression }
10040: Proc SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
10041: { Sets the bounds on the I-th regression parameter }
10042: Proc GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
```

```
10043: { Returns the bounds on the I-th regression parameter }
10044: Proc NLFit(RegFunc   : TRegFunc;
10045:                DerivProc : TDerivProc;
10046:                X, Y      : TVector;
10047:                Lb, Ub    : Int;
10048:                MaxIter   : Int;
10049:                Tol       : Float;
10050:                B         : TVector;
10051:                FirstPar,
10052:                LastPar   : Int;
10053:                V         : TMatrix); external 'dmath';
10054: { Unweighted nonlinear regression }
10055: Proc WNLFit(RegFunc   : TRegFunc;
10056:                 DerivProc : TDerivProc;
10057:                 X, Y, S   : TVector;
10058:                 Lb, Ub    : Int;
10059:                 MaxIter   : Int;
10060:                 Tol       : Float;
10061:                 B         : TVector;
10062:                 FirstPar, LastPar   : Int;
10063:                 V         : TMatrix); external 'dmath';
10064: { Weighted nonlinear regression }
10065: Proc SetMCFile(FileName :Str); external 'dmath';
10066: { Set file for saving MCMC simulations }
10067: Proc SimFit(RegFunc   : TRegFunc;
10068:                 X, Y      : TVector;
10069:                 Lb, Ub    : Int;
10070:                 B         : TVector;
10071:                 FirstPar, LastPar   : Int;
10072:                 V         : TMatrix); external 'dmath';
10073: { Simulation of unweighted nonlinear regression by MCMC }
10074: Proc WSimFit(RegFunc   : TRegFunc;
10075:                  X, Y, S   : TVector;
10076:                  Lb, Ub    : Int;
10077:                  B         : TVector;
10078:                  FirstPar, LastPar   : Int;
10079:                  V         : TMatrix); external 'dmath';
10080: { Simulation of weighted nonlinear regression by MCMC }
10081: { ----------------------------------------------------------------
10082:   Nonlinear regression models
10083:   ---------------------------------------------------------------- }
10084: Proc FracFit(X, Y       : TVector;
10085:                  Lb, Ub     : Int;
10086:                  Deg1, Deg2 : Int;
10087:                  ConsTerm   :Bool;
10088:                  MaxIter    : Int;
10089:                  Tol        : Float;
10090:                  B          : TVector;
10091:                  V          : TMatrix); external 'dmath';
10092: { Unweighted fit of rational fraction }
10093: Proc WFracFit(X, Y, S    : TVector;
10094:                   Lb, Ub     : Int;
10095:                   Deg1, Deg2 : Int;
10096:                   ConsTerm   :Bool;
10097:                   MaxIter    : Int;
10098:                   Tol        : Float;
10099:                   B          : TVector;
10100:                   V          : TMatrix); external 'dmath';
10101: { Weighted fit of rational fraction }
10102:
10103: Func FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10104: { Returns the value of the rational fraction at point X }
10105: Proc ExpFit(X, Y         : TVector;
10106:                 Lb, Ub, Nexp : Int;
10107:                 ConsTerm     :Bool;
10108:                 MaxIter      : Int;
10109:                 Tol          : Float;
10110:                 B            : TVector;
10111:                 V            : TMatrix); external 'dmath';
10112: { Unweighted fit of sum of exponentials }
10113: Proc WExpFit(X, Y, S      : TVector;
10114:                  Lb, Ub, Nexp : Int;
10115:                  ConsTerm     :Bool;
10116:                  MaxIter      : Int;
10117:                  Tol          : Float;
10118:                  B            : TVector;
10119:                  V            : TMatrix); external 'dmath';
10120: { Weighted fit of sum of exponentials }
10121: Func ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10122: { Returns the value of the regression Func at point X }
10123: Proc IncExpFit(X, Y     : TVector;
10124:                    Lb, Ub   : Int;
10125:                    ConsTerm :Bool;
10126:                    MaxIter  : Int;
10127:                    Tol      : Float;
10128:                    B        : TVector;
10129:                    V        : TMatrix); external 'dmath';
10130: { Unweighted fit of model of increasing exponential }
10131: Proc WIncExpFit(X, Y, S  : TVector;
```

```
10132:                      Lb, Ub   : Int;
10133:                      ConsTerm :Bool;
10134:                      MaxIter  : Int;
10135:                      Tol      : Float;
10136:                      B        : TVector;
10137:                      V        : TMatrix); external 'dmath';
10138: { Weighted fit of increasing exponential }
10139: Func IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10140: { Returns the value of the regression Func at point X }
10141: Proc ExpLinFit(X, Y    : TVector;
10142:                     Lb, Ub  : Int;
10143:                     MaxIter : Int;
10144:                     Tol     : Float;
10145:                     B       : TVector;
10146:                     V       : TMatrix); external 'dmath';
10147: { Unweighted fit of the "exponential + linear" model }
10148: Proc WExpLinFit(X, Y, S : TVector;
10149:                      Lb, Ub  : Int;
10150:                      MaxIter : Int;
10151:                      Tol     : Float;
10152:                      B       : TVector;
10153:                      V       : TMatrix); external 'dmath';
10154: { Weighted fit of the "exponential + linear" model }
10155: Func ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10156: { Returns the value of the regression Func at point X }
10157: Proc MichFit(X, Y    : TVector;
10158:                  Lb, Ub  : Int;
10159:                  MaxIter : Int;
10160:                  Tol     : Float;
10161:                  B       : TVector;
10162:                  V       : TMatrix); external 'dmath';
10163: { Unweighted fit of Michaelis equation }
10164: Proc WMichFit(X, Y, S : TVector;
10165:                   Lb, Ub  : Int;
10166:                   MaxIter : Int;
10167:                   Tol     : Float;
10168:                   B       : TVector;
10169:                   V       : TMatrix); external 'dmath';
10170: { Weighted fit of Michaelis equation }
10171: Func MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10172: { Returns the value of the Michaelis equation at point X }
10173: Proc MintFit(X, Y    : TVector;
10174:                  Lb, Ub  : Int;
10175:                  MintVar : TMintVar;
10176:                  Fit_S0  :Bool;
10177:                  MaxIter : Int;
10178:                  Tol     : Float;
10179:                  B       : TVector;
10180:                  V       : TMatrix); external 'dmath';
10181: { Unweighted fit of the integrated Michaelis equation }
10182: Proc WMintFit(X, Y, S : TVector;
10183:                   Lb, Ub  : Int;
10184:                   MintVar : TMintVar;
10185:                   Fit_S0  :Bool;
10186:                   MaxIter : Int;
10187:                   Tol     : Float;
10188:                   B       : TVector;
10189:                   V       : TMatrix); external 'dmath';
10190: { Weighted fit of the integrated Michaelis equation }
10191: Func MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10192: { Returns the value of the integrated Michaelis equation at point X }
10193: Proc HillFit(X, Y    : TVector;
10194:                  Lb, Ub  : Int;
10195:                  MaxIter : Int;
10196:                  Tol     : Float;
10197:                  B       : TVector;
10198:                  V       : TMatrix); external 'dmath';
10199: { Unweighted fit of Hill equation }
10200: Proc WHillFit(X, Y, S : TVector;
10201:                   Lb, Ub  : Int;
10202:                   MaxIter : Int;
10203:                   Tol     : Float;
10204:                   B       : TVector;
10205:                   V       : TMatrix); external 'dmath';
10206: { Weighted fit of Hill equation }
10207: Func HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10208: { Returns the value of the Hill equation at point X }
10209: Proc LogiFit(X, Y     : TVector;
10210:                  Lb, Ub   : Int;
10211:                  ConsTerm :Bool;
10212:                  General  :Bool;
10213:                  MaxIter  : Int;
10214:                  Tol      : Float;
10215:                  B        : TVector;
10216:                  V        : TMatrix); external 'dmath';
10217: { Unweighted fit of logistic Func }
10218: Proc WLogiFit(X, Y, S  : TVector;
10219:                   Lb, Ub   : Int;
10220:                   ConsTerm :Bool;
```

```
10221:                    General  :Bool;
10222:                    MaxIter  : Int;
10223:                    Tol      : Float;
10224:                    B        : TVector;
10225:                    V        : TMatrix); external 'dmath';
10226: { Weighted fit of logistic Func }
10227: Func LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10228: { Returns the value of the logistic Func at point X }
10229: Proc PKFit(X, Y    : TVector;
10230:                Lb, Ub  : Int;
10231:                MaxIter : Int;
10232:                Tol     : Float;
10233:                B       : TVector;
10234:                V       : TMatrix); external 'dmath';
10235: { Unweighted fit of the acid-base titration curve }
10236: Proc WPKFit(X, Y, S : TVector;
10237:                 Lb, Ub  : Int; MaxIter : Int;
10238:                 Tol     : Float;
10239:                 B       : TVector;
10240:                 V       : TMatrix); external 'dmath';
10241: { Weighted fit of the acid-base titration curve }
10242: Func PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10243: { Returns the value of the acid-base titration Func at point X }
10244: Proc PowFit(X, Y    : TVector;
10245:                 Lb, Ub  : Int;
10246:                 MaxIter : Int;
10247:                 Tol     : Float;
10248:                 B       : TVector;
10249:                 V       : TMatrix); external 'dmath';
10250: { Unweighted fit of power Func }
10251: Proc WPowFit(X, Y, S : TVector;
10252:                  Lb, Ub  : Int;
10253:                  MaxIter : Int;
10254:                  Tol     : Float;
10255:                  B       : TVector;
10256:                  V       : TMatrix); external 'dmath';
10257: { Weighted fit of power Func }
10258:
10259: Func PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10260: { Returns the value of the power Func at point X }
10261: Proc GammaFit(X, Y    : TVector;
10262:                   Lb, Ub  : Int;
10263:                   MaxIter : Int;
10264:                   Tol     : Float;
10265:                   B       : TVector;
10266:                   V       : TMatrix); external 'dmath';
10267: { Unweighted fit of gamma distribution Func }
10268: Proc WGammaFit(X, Y, S : TVector;
10269:                    Lb, Ub  : Int;
10270:                    MaxIter : Int;
10271:                    Tol     : Float;
10272:                    B       : TVector;
10273:                    V       : TMatrix); external 'dmath';
10274: { Weighted fit of gamma distribution Func }
10275: Func GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10276: { Returns the value of the gamma distribution Func at point X }
10277: { ---------------------------------------------------------------
10278:   Principal component analysis
10279:   ---------------------------------------------------------------- }
10280: Proc VecMean(X            : TMatrix;
10281:                  Lb, Ub, Nvar : Int;
10282:                  M            : TVector); external 'dmath';
10283: { Computes the mean vector M from matrix X }
10284: Proc VecSD(X            : TMatrix;
10285:                Lb, Ub, Nvar : Int;
10286:                M, S         : TVector); external 'dmath';
10287: { Computes the vector of standard deviations S from matrix X }
10288: Proc MatVarCov(X            : TMatrix;
10289:                    Lb, Ub, Nvar : Int;
10290:                    M            : TVector;
10291:                    V            : TMatrix); external 'dmath';
10292: { Computes the variance-covariance matrix V from matrix X }
10293: Proc MatCorrel(V    : TMatrix;
10294:                    Nvar : Int;
10295:                    R    : TMatrix); external 'dmath';
10296: { Computes the correlation matrix R from the var-cov matrix V }
10297: Proc PCA(R      : TMatrix;
10298:              Nvar   : Int;
10299:              Lambda : TVector;
10300:              C, Rc  : TMatrix); external 'dmath';
10301: { Performs a principal component analysis of the correlation matrix R }
10302: Proc ScaleVar(X            : TMatrix;
10303:                   Lb, Ub, Nvar : Int;
10304:                   M, S         : TVector;
10305:                   Z            : TMatrix); external 'dmath';
10306: { Scales a set of variables by subtracting means and dividing by SD's }
10307: Proc PrinFac(Z            : TMatrix;
10308:                  Lb, Ub, Nvar : Int;
10309:                  C, F         : TMatrix); external 'dmath';
```

```
10310: { Computes principal factors }
10311: { ----------------------------------------------------------------
10312:   Strings
10313:   ---------------------------------------------------------------- }
10314: Func LTrim(S :Str) :Str; external 'dmath';
10315: { Removes leading blanks }
10316: Func RTrim(S :Str) :Str; external 'dmath';
10317: { Removes trailing blanks }
10318: Func Trim(S :Str) :Str; external 'dmath';
10319: { Removes leading and trailing blanks }
10320: Func StrChar(N : Byte; C : Char) :Str; external 'dmath';
10321: { Returns a string made of character C repeated N times }
10322: Func RFill(S :Str; L : Byte) :Str; external 'dmath';
10323: { Completes string S with trailing blanks for a total length L }
10324: Func LFill(S :Str; L : Byte) :Str; external 'dmath';
10325: { Completes string S with leading blanks for a total length L }
10326: Func CFill(S :Str; L : Byte) :Str; external 'dmath';
10327: { Centers string S on a total length L }
10328: Func Replace(S :Str; C1, C2 : Char) :Str; external 'dmath';
10329: { Replaces in string S all the occurences of C1 by C2 }
10330: Func Extract(S:str; var Index:Byte; Delim:Char):str; external 'dmath';
10331: { Extracts a field from a string }
10332: Proc Parse(S:str;Delim:Char;Field:TStrVector;var N:Byte);external 'dmath';
10333: { Parses a string into its constitutive fields }
10334: Proc SetFormat(NumLength,MaxDec:Int;FloatPoint,NSZero:Bool); external 'dmath';
10335: { Sets the numeric format }
10336: Func FloatStr(X : Float) :Str; external 'dmath';
10337: { Converts a real to a string according to the numeric format }
10338: Func IntStr(N : LongInt) :Str; external 'dmath';
10339: { Converts an Int to a string }
10340: Func CompStr(Z : Complex) :Str; external 'dmath';
10341: { Converts a complex number to a string }
10342: {$IFDEF DELPHI}
10343: Func StrDec(S :Str) :Str; external 'dmath';
10344: { Set decimal separator to the symbol defined in SysUtils }
10345: Func IsNumeric(var S :Str; var X : Float) :Bool; external 'dmath';
10346: { Test if a string represents a number and returns it in X }
10347: Func ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10348: { Reads a floating point number from an Edit control }
10349: Proc WriteNumToFile(var F : Text; X : Float); external 'dmath';
10350: { Writes a floating point number in a text file }
10351: {$ENDIF}
10352: { ----------------------------------------------------------------
10353:   BGI / Delphi graphics
10354:   ---------------------------------------------------------------- }
10355: Func InitGraphics
10356: {$IFDEF DELPHI}
10357: (Width, Height : Int) :Bool;
10358: {$ELSE}
10359: (Pilot, Mode : Int; BGIPath :Str) :Bool; {$ENDIF} external 'dmath';
10360: { Enters graphic mode }
10361: Proc SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10362:                     X1, X2, Y1,Y2 : Int; GraphBorder:Boolean); external 'dmath';
10363: { Sets the graphic window }
10364: Proc SetOxScale(Scale                : TScale;
10365:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10366: { Sets the scale on the Ox axis }
10367: Proc SetOyScale(Scale                : TScale;
10368:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10369: { Sets the scale on the Oy axis }
10370: Proc GetOxScale(var Scale                : TScale;
10371:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10372: { Returns the scale on the Ox axis }
10373: Proc GetOyScale(var Scale                : TScale;
10374:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10375: { Returns the scale on the Oy axis }
10376: Proc SetGraphTitle(Title:str);external 'dmath';{Sets title for the graph }
10377: Proc SetOxTitle(Title String);external'dmath';{ Sets title for the Ox axis }
10378: Proc SetOyTitle(Title:str);external'dmath';{Sets the title for Oy axis }
10379: Func GetGraphTitle:str;external 'dmath';{ Returns the title for the graph }
10380: Func GetOxTitle:str; external 'dmath'; { Returns the title for the Ox axis }
10381: Func GetOyTitle:str; external 'dmath'; { Returns the title for the Oy axis }
10382: {$IFNDEF DELPHI}
10383: Proc SetTitleFont(FontIndex, Width, Height : Int); external 'dmath';
10384: { Sets the font for the main graph title }
10385: Proc SetOxFont(FontIndex, Width, Height : Int); external 'dmath';
10386: { Sets the font for the Ox axis (title and labels) }
10387: Proc SetOyFont(FontIndex, Width, Height : Int); external 'dmath';
10388: { Sets the font for the Oy axis (title and labels) }
10389: Proc SetLgdFont(FontIndex, Width, Height : Int); external 'dmath';
10390: { Sets the font for the legends }
10391: Proc SetClipping(Clip :Bool); external 'dmath';
10392: { Determines whether drawings are clipped at the current viewport
10393:   boundaries, according to the value of the Boolean parameter Clip }
10394: {$ENDIF}
10395: Proc PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10396: { Plots the horizontal axis }
10397: Proc PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10398: { Plots the vertical axis }
```

```
10399: Proc PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF}Grid:TGrid);external 'dmath';
10400: { Plots a grid on the graph }
10401: Proc WriteGraphTitle{$IFDEF DELPHI}(Canvas:TCanvas){$ENDIF}; external 'dmath';
10402: { Writes the title of the graph }
10403: Proc SetMaxCurv(NCurv : Byte); external 'dmath';
10404: { Sets the maximum number of curves and re-initializes their parameters }
10405: Proc SetPointParam
10406: {$IFDEF DELPHI}
10407: (CurvIndex, Symbol, Size : Int; Color : TColor);
10408: {$ELSE}
10409: (CurvIndex, Symbol, Size, Color : Int); {$ENDIF} external 'dmath';
10410: { Sets the point parameters for curve # CurvIndex }
10411: Proc SetLineParam
10412: {$IFDEF DELPHI}
10413: (CurvIndex : Int; Style : TPenStyle; Width : Int; Color : TColor);
10414: {$ELSE}
10415: (CurvIndex, Style, Width, Color : Int); {$ENDIF} external 'dmath';
10416: { Sets the line parameters for curve # CurvIndex }
10417: Proc SetCurvLegend(CurvIndex : Int; Legend :Str); external 'dmath';
10418: { Sets the legend for curve # CurvIndex }
10419: Proc SetCurvStep(CurvIndex, Step : Int); external 'dmath';
10420: { Sets the step for curve # CurvIndex }
10421: Func GetMaxCurv: Byte;external 'dmath';{ Returns the maximum number of curves }
10422: Proc GetPointParam
10423: {$IFDEF DELPHI}
10424: (CurvIndex : Int; var Symbol, Size : Int; var Color : TColor);
10425: {$ELSE}
10426: (CurvIndex : Int; var Symbol, Size, Color : Int); {$ENDIF} external 'dmath';
10427: { Returns the point parameters for curve # CurvIndex }
10428: Proc GetLineParam
10429: {$IFDEF DELPHI}
10430: (CurvIndex : Int; var Style : TPenStyle; var Width : Int; var Color : TColor);
10431: {$ELSE}
10432: (CurvIndex : Int; var Style, Width, Color : Int); {$ENDIF} external 'dmath';
10433: { Returns the line parameters for curve # CurvIndex }
10434: Func GetCurvLegend(CurvIndex : Int) :Str; external 'dmath';
10435: { Returns the legend for curve # CurvIndex }
10436: Func GetCurvStep(CurvIndex : Int) : Int; external 'dmath';
10437: { Returns the step for curve # CurvIndex }
10438: {$IFDEF DELPHI}
10439: Proc PlotPoint(Canvas    : TCanvas;
10440:                    X, Y      : Float; CurvIndex : Int); external 'dmath';
10441: {$ELSE}
10442: Proc PlotPoint(Xp, Yp, CurvIndex : Int); external 'dmath';
10443: {$ENDIF}
10444: { Plots a point on the screen }
10445: Proc PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10446:                    X, Y                 : TVector;
10447:                   Lb, Ub, CurvIndex     : Int); external 'dmath';
10448: { Plots a curve }
10449: Proc PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10450:                                X, Y, S                : TVector;
10451:                                Ns, Lb, Ub, CurvIndex : Int); external 'dmath';
10452: { Plots a curve with error bars }
10453: Proc PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10454:                   Func                 : TFunc;
10455:                   Xmin, Xmax           : Float;
10456:                   {$IFDEF DELPHI}Npt    : Int;{$ENDIF}
10457:                   CurvIndex            : Int); external 'dmath';
10458: { Plots a Func }
10459: Proc WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10460:                      NCurv                 : Int;
10461:                      ShowPoints, ShowLines :Bool); external 'dmath';
10462: { Writes the legends for the plotted curves }
10463: Proc ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10464:                 Nx, Ny, Nc            : Int;
10465:                 X, Y, Z               : TVector;
10466:                 F                     : TMatrix); external 'dmath';
10467: { Contour plot }
10468: Func Xpixel(X:Float):Int; external 'dmath';{Converts user abscissa X to screen coordinate }
10469: Func Ypixel(Y:Float):Int; external 'dmath';{Converts user ordinate Y to screen coordinate }
10470: Func Xuser(X:Int):Float; external 'dmath';{Converts screen coordinate X to user abscissa }
10471: Func Yuser(Y:Int):Float; external 'dmath';{Converts screen coordinate Y to user ordinate }
10472: {$IFNDEF DELPHI}
10473: Proc LeaveGraphics; external 'dmath';
10474: { Quits graphic mode }
10475: {$ENDIF}
10476: { ----------------------------------------------------------------
10477:   LaTeX graphics
10478:   ---------------------------------------------------------------- }
10479: Func TeX_InitGraphics(FileName          :Str; PgWidth, PgHeight : Int;
10480:                         Header            :Bool) :Bool; external 'dmath';
10481: { Initializes the LaTeX file }
10482: Proc TeX_SetWindow(X1,X2,Y1 Y2:Int; GraphBorder :Bool); external 'dmath';
10483: { Sets the graphic window }
10484: Proc TeX_LeaveGraphics(Footer:Boolean);external 'dmath';{ Close LaTeX file }
10485: Proc TeX_SetOxScale(Scale:TScale;OxMin,OxMax,OxStep:Float);external 'dmath';
10486: { Sets the scale on the Ox axis }
10487: Proc TeX_SetOyScale(Scale:TScale;OyMin,OyMax,OyStep:Float); external 'dmath';
```

```
10488: { Sets the scale on the Oy axis }
10489: Proc TeX_SetGraphTitle(Title:str);external 'dmath';{ Sets title for graph }
10490: Proc TeX_SetOxTitle(Title:str);external 'dmath';{ Sets title for Ox axis }
10491: Proc TeX_SetOyTitle(Title:str);external 'dmath';{ Sets title for Oy axis }
10492: Proc TeX_PlotOxAxis; external 'dmath'; { Plots  horizontal axis }
10493: Proc TeX_PlotOyAxis; external 'dmath'; { Plots  vertical axis }
10494: Proc TeX_PlotGrid(Grid:TGrid);external 'dmath'; { Plots a grid on the graph }
10495: Proc TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10496: Proc TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10497: { Sets the maximum number of curves and re-initializes their parameters }
10498: Proc TeX_SetPointParam(CurvIndex, Symbol, Size : Int); external 'dmath';
10499: { Sets the point parameters for curve # CurvIndex }
10500: Proc TeX_SetLineParam(CurvIndex, Style : Int;
10501:                Width : Float; Smooth :Bool); external 'dmath';
10502: { Sets the line parameters for curve # CurvIndex }
10503: Proc TeX_SetCurvLegend(CurvIndex : Int; Legend :Str); external 'dmath';
10504: { Sets the legend for curve # CurvIndex }
10505: Proc TeX_SetCurvStep(CurvIndex, Step : Int); external 'dmath';
10506: { Sets the step for curve # CurvIndex }
10507: Proc TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Int); external 'dmath';
10508: { Plots a curve }
10509: Proc TeX_PlotCurveWithErrorBars(X,Y,S:TVector;Ns,Lb,Ub,CurvIndex Int);external 'dmath';
10510: { Plots a curve with error bars }
10511: Proc TeX_PlotFunc(Func:TFunc;X1 X2:Float;Npt:Int;CurvIndex:Int);external 'dmath';
10512: { Plots a Func }
10513: Proc TeX_WriteLegend(NCurv:Int;ShowPoints,ShowLines:Bool); external 'dmath';
10514: { Writes the legends for the plotted curves }
10515: Proc TeX_ConRec(Nx,Ny Nc:Int; X,Y,Z :TVector; F : TMatrix); external 'dmath';
10516: { Contour plot }
10517: Func Xcm(X:Float):Float; external 'dmath'; { Converts user coordinate X to cm }
10518: Func Ycm(Y:Float):Float; external 'dmath'; { Converts user coordinate Y to cm }
10519:
10520: //*****************************************************unit uPSI_SynPdf;
10521:  Func RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10522:  Func _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10523:  Func _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10524:  Func PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10525:  Func PdfRect1( const Box : TPdfBox) : TPdfRect;
10526:  Func PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10527:  //Func _GetCharCount( Text : PAnsiChar) : Int
10528:  //Proc L2R( W : PWideChar; L : Int)
10529:  Func PdfCoord( MM : single) : Int
10530:  Func CurrentPrinterPaperSize : TPDFPaperSize
10531:  Func CurrentPrinterRes : TPoint
10532:  Proc GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10533:  Proc GDICommentOutline(MetaHandle:HDC; const aTitle : RawUTF8; aLevel : Int)
10534:  Proc GDICommentLink(MetaHandle:HDC;const aBookmarkName:RawUTF8;const aRect: TRect)
10535:  Const('Usp10','String 'usp10.dll
10536:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10537:   'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10538:  TScriptState_set', 'set of TScriptState_enum
10539: //*********************************************************************
10540:
10541: Proc SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10542: begin
10543:  Proc PMrandomize( I : word)
10544:  Func PMrandom : longint
10545:  Func Rrand : extended
10546:  Func Irand( N : word) : word
10547:  Func Brand( P : extended) :Bool
10548:  Func Nrand : extended
10549: end;
10550:
10551: Proc SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10552: begin
10553:   Func Endian( x : LongWord) : LongWord
10554:   Func Endian64( x : Int64) : Int64
10555:   Func spRol( x : LongWord; y : Byte) : LongWord
10556:   Func spRor( x : LongWord; y : Byte) : LongWord
10557:   Func Ror64( x : Int64; y : Byte) : Int64
10558: end;
10559:
10560: Proc SIRegister_MapReader(CL: TPSPascalCompiler);
10561: begin
10562:  Proc ClearModules
10563:  Proc ReadMapFile( Fname :Str)
10564:  Func AddressInfo( Address : dword) :Str
10565: end;
10566:
10567: Proc SIRegister_LibTar(CL: TPSPascalCompiler);
10568: begin
10569:  TTarPermission','(tpReadByOwner,tpWriteByOwner,tpExecuteByOwner,tpReadByGroup,tpWriteByGroup,
       tpExecuteByGroup,tpReadByOther,tpWriteByOther, tpExecuteByOther)
10570:   TTarPermissions', 'set of TTarPermission
10571:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10572:    +'Block,ftDirectory,ftFifo,ftContiguous,ftDumpDir,ftMultiVolume,ftVolumeHeader;
10573:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10574:   TTarModes', 'set of TTarMode
10575:   TTarDirRec', 'record Name :Str; Size : INT64; DateTime : TDa'
```

```
10576:    +teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10577:    +RING; UID : Int; GID : Int; UserName :Str; GroupName :Str;'
10578:    + ChecksumOK :Bool; Mode : TTarModes; Magic :Str; MajorDevNo : INTE'
10579:    +GER; MinorDevNo : Int; FilePos : INT64; end
10580:   SIRegister_TTarArchive(CL);
10581:   SIRegister_TTarWriter(CL);
10582:  Func PermissionString( Permissions : TTarPermissions) :Str
10583:  Func ConvertFilename( Filename :Str) :Str
10584:  Func FileTimeGMT( FileName :Str) : TDateTime;
10585:  Func FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10586:  Proc ClearDirRec( var DirRec : TTarDirRec)
10587: end;
10588:
10589: //*****************************************************unit uPSI_TlHelp32;
10590: Proc SIRegister_TlHelp32(CL: TPSPascalCompiler);
10591: begin
10592:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10593:  Func CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10594:  Const('TH32CS_SNAPHEAPLIST',LongWord( $00000001);
10595:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10596:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10597:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10598:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10599:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10600:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10601:   AddTypeS('THeapList32', 'tagHEAPLIST32
10602:  Const('HF32_DEFAULT','LongInt'( 1);
10603:  Const('HF32_SHARED','LongInt'( 2);
10604:  Func Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10605:  Func Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10606:  AddTypeS(tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10607:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10608:    +dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10609:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10610:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10611:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10612:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10613:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10614:  Func Heap32First(var lphe:THeapEntry32;th32ProcessID,th32HeapID:DWORD):BOOL
10615:  Func Heap32Next( var lphe : THeapEntry32) : BOOL
10616:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10617:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10618:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10619:    +'aPri : Longint; dwFlags : DWORD; end
10620:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10621:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10622:  Func Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10623:  Func Thread32Next( hSnapshot: THandle; var lpte : TThreadENtry32):BOOL end;
10624:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10625:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10626:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10627:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10628:  Const('EWX_LOGOFF','LongInt'( 0);
10629:  Const('EWX_SHUTDOWN','LongInt'( 1);
10630:  Const('EWX_REBOOT','LongInt'( 2);
10631:  Const('EWX_FORCE','LongInt'( 4);
10632:  Const('EWX_POWEROFF','LongInt'( 8);
10633:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10634:  Func GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10635:  Func GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10636:  Func GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10637:  Func GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10638:  Func GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10639:  Func GetWindowWord( hWnd : HWND; nIndex : Int) : Word
10640:  Func SetWindowWord( hWnd : HWND; nIndex : Int; wNewWord : Word) : Word
10641:  Func GetWindowLong( hWnd : HWND; nIndex : Int) : Longint
10642:  Func SetWindowLong( hWnd : HWND; nIndex : Int; dwNewLong:Longint) : Longint
10643:  Func GetClassWord( hWnd : HWND; nIndex : Int) : Word
10644:  Func SetClassWord( hWnd : HWND; nIndex : Int; wNewWord : Word) : Word
10645:  Func GetClassLong( hWnd : HWND; nIndex : Int) : DWORD
10646:  Func SetClassLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint) : DWORD
10647:  Func GetDesktopWindow : HWND
10648:  Func GetParent( hWnd : HWND) : HWND
10649:  Func SetParent( hWndChild, hWndNewParent : HWND) : HWND
10650:  Func GetTopWindow( hWnd : HWND) : HWND
10651:  Func GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10652:  Func GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10653:  //Delphi DFM
10654:  Func LoadDFMFile2Strings(const AFile:str; AStrings:TStrings; var WasText:bool):Int
10655:  Func SaveStrings2DFMFile( AStrings : TStrings; const AFile :Str) : Int
10656:  Proc GetHighlighters(AOwner:TComponent; AHighlighters:TStringList;AppendToList:Bool);
10657:  Func GetHighlightersFilter(AHighlighters: TStringList):Str;
10658:  Func GetHighlighterFromFileExt(AHighlighters: TStringList;Extension:Str):TSynCustomHighlighter;
10659:  Func ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10660:  Func OpenIcon( hWnd : HWND) : BOOL
10661:  Func CloseWindow( hWnd : HWND) : BOOL
10662:  Func MoveWindow(hWnd:HWND;X Y,nWidth, nHeight : Int; bRepaint : BOOL) : BOOL
10663:  Func SetWindowPos(hWnd:HWND;hWndInsertAfter:HWND;X,Y,cx,cy:Int;uFlags:UINT): BOOL
10664:  Func IsWindowVisible( hWnd : HWND) : BOOL
```

```
10665:  Func IsIconic( hWnd : HWND) : BOOL
10666:  Func AnyPopup : BOOL
10667:  Func BringWindowToTop( hWnd : HWND) : BOOL
10668:  Func IsZoomed( hWnd : HWND) : BOOL
10669:  Func IsWindow( hWnd : HWND) : BOOL
10670:  Func IsMenu( hMenu : HMENU) : BOOL
10671:  Func IsChild( hWndParent, hWnd : HWND) : BOOL
10672:  Func DestroyWindow( hWnd : HWND) : BOOL
10673:  Func ShowWindow( hWnd : HWND; nCmdShow : Int) : BOOL
10674:  Func AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10675:  Func ShowWindowAsync( hWnd : HWND; nCmdShow : Int) : BOOL
10676:  Func FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10677:  Func IsWindowUnicode( hWnd : HWND) : BOOL
10678:  Func EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10679:  Func IsWindowEnabled( hWnd : HWND) : BOOL
10680:
10681: Proc SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10682: begin
10683:  const('ShowSetupDialogOptLong','String '--setup
10684:  PrimaryConfPathOptLong','String '--primary-config-path=
10685:  PrimaryConfPathOptShort','String '--pcp=
10686:  SecondaryConfPathOptLong','String '--secondary-config-path=
10687:  SecondaryConfPathOptShort','String '--scp=
10688:  NoSplashScreenOptLong','String '--no-splash-screen
10689:  NoSplashScreenOptShort','String '--nsc
10690:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10691:  SkipLastProjectOpt','String '--skip-last-project
10692:  DebugLogOpt','String '--debug-log=
10693:  DebugLogOptEnable','String '--debug-enable=
10694:  LanguageOpt','String '--language=
10695:  LazarusDirOpt','String '--lazarusdir=
10696:  Proc ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:bool);
10697:  Func GetCommandLineParameters(aCmdLineParams:TStrings;isStartLazarus:Bool):Str
10698:  Func ExtractPrimaryConfigPath( aCmdLineParams : TStrings) :Str
10699:  Func IsHelpRequested :Bool
10700:  Func IsVersionRequested :Bool
10701:  Func GetLanguageSpecified :Str
10702:  Func ParamIsOption( ParamIndex : Int; const Option :Str) :Bool
10703:  Func ParamIsOptionPlusValue(ParamIndex:int;const Option:str;out AValue:str):bool;
10704:  Proc ParseNoGuiCmdLineParams
10705:  Func ExtractCmdLineFilenames : TStrings
10706: end;
10707:
10708: Proc SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10709: begin
10710:  Func CompareFilenames( const Filename1, Filename2 :Str) : Int
10711:  Func CompareFilenamesIgnoreCase( const Filename1, Filename2 :Str) : Int
10712:  Func CompareFileExt( const Filename, Ext :Str; CaseSensitive :Bool) : Int;
10713:  Func CompareFileExt1( const Filename, Ext :Str) : Int;
10714:  Func CompareFilenameStarts( const Filename1, Filename2 :Str) : Int
10715:  Func CompareFilenames(Filename1:PChar;Len1:Int; Filename2:PChar;Len2:Int):Int
10716:  Func CompareFilenamesP(Filename1,Filename2:PChar;IgnoreCase :Bool) : Int
10717:  Func DirPathExists( DirectoryName :Str) :Bool
10718:  Func DirectoryIsWritable( const DirectoryName :Str) :Bool
10719:  Func ExtractFileNameOnly( const AFilename :Str) :Str
10720:  Func FilenameIsAbsolute( const TheFilename :Str) :Bool
10721:  Func FilenameIsWinAbsolute( const TheFilename :Str) :Bool
10722:  Func FilenameIsUnixAbsolute( const TheFilename :Str) :Bool
10723:  Func ForceDirectory( DirectoryName :Str) :Bool
10724:  Proc CheckIfFileIsExecutable( const AFilename :Str)
10725:  Proc CheckIfFileIsSymlink( const AFilename :Str)
10726:  Func FileIsText( const AFilename :Str) :Bool
10727:  Func FileIsText2(const AFilename:str; out FileReadable: bool): bool
10728:  Func FilenameIsTrimmed( const TheFilename :Str) :Bool
10729:  Func FilenameIsTrimmed2( StartPos : PChar; NameLen : Int) :Bool
10730:  Func TrimFilename( const AFilename :Str) :Str
10731:  Func ResolveDots( const AFilename :Str) :Str
10732:  Proc ForcePathDelims( var FileName :Str)
10733:  Func GetForcedPathDelims( const FileName :Str) :Str
10734:  Func CleanAndExpandFilename( const Filename :Str) :Str
10735:  Func CleanAndExpandDirectory( const Filename :Str) :Str
10736:  Func TrimAndExpandFilename(const Filename:str;const BaseDir:str):str
10737:  Func TrimAndExpandDirectory(const Filename:str;const BaseDir:str):Str
10738:  Func TryCreateRelativePath(const Dest,Source:str; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder:Bool; out RelPath :Str) :Bool
10739:  Func CreateRelativePath(const Filename,
       BaseDirectory:str;UsePointDirectory:boolean;AlwaysRequireSharedBaseFolder:Bool) :Str
10740:  Func FileIsInPath( const Filename, Path :Str) :Bool
10741:  Func AppendPathDelim( const Path :Str) :Str
10742:  Func ChompPathDelim( const Path :Str) :Str
10743:  Func CreateAbsoluteSearchPath(const SearchPath,BaseDirectory:str):str
10744:  Func CreateRelativeSearchPath(const SearchPath,BaseDirectory:str):Str
10745:  Func MinimizeSearchPath( const SearchPath :Str) :Str
10746:  Func FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10747:  (*Func FileExistsUTF8( const Filename :Str) :Bool
10748:  Func FileAgeUTF8( const FileName :Str) : Longint
10749:  Func DirectoryExistsUTF8( const Directory :Str) :Bool
10750:  Func ExpandFileNameUTF8( const FileName :Str; BaseDir :Str) :Str
10751:  Func FindFirstUTF8(const Path:str;Attr:Longint;out Rslt:TSearchRec): Longint
```

```
10752:  Func FindNextUTF8( var Rslt : TSearchRec) : Longint
10753:  Proc FindCloseUTF8( var F : TSearchrec)
10754:  Func FileSetDateUTF8( const FileName :Str; Age : Longint) : Longint
10755:  Func FileGetAttrUTF8( const FileName :Str) : Longint
10756:  Func FileSetAttrUTF8( const Filename :Str; Attr : longint) : Longint
10757:  Func DeleteFileUTF8( const FileName :Str) :Bool
10758:  Func RenameFileUTF8( const OldName, NewName :Str) :Bool
10759:  Func FileSearchUTF8(const Name,DirList:str;ImplicitCurrentDir:Bool):str
10760:  Func FileIsReadOnlyUTF8( const FileName :Str) :Bool
10761:  Func GetCurrentDirUTF8 :Str
10762:  Func SetCurrentDirUTF8( const NewDir :Str) :Bool
10763:  Func CreateDirUTF8( const NewDir :Str) :Bool
10764:  Func RemoveDirUTF8( const Dir :Str) :Bool
10765:  Func ForceDirectoriesUTF8( const Dir :Str) :Bool
10766:  Func FileOpenUTF8( const FileName :Str; Mode : Int) : THandle
10767:  Func FileCreateUTF8( const FileName :Str) : THandle;
10768:  Func FileCreateUTF81( const FileName :Str; Rights :Card): THandle;
10769:  Func FileCreateUtf82(const FileName:str;ShareMode:Int;Rights:Card): THandle;
10770:  Func FileSizeUtf8( const Filename :Str) : int64
10771:  Func GetFileDescription( const AFilename :Str) :Str
10772:  Func GetAppConfigDirUTF8( Global :Bool; Create :Bool) :Str
10773:  Func GetAppConfigFileUTF8(Global:Bool;SubDir:bool;CreateDir: bool):Str
10774:  Func GetTempFileNameUTF8( const Dir, Prefix :Str) :Str*)
10775:  Func IsUNCPath( const Path :Str) :Bool
10776:  Func ExtractUNCVolume( const Path :Str) :Str
10777:  Func ExtractFileRoot( FileName :Str) :Str
10778:  Func GetDarwinSystemFilename( Filename :Str) :Str
10779:  Proc SplitCmdLineParams( const Params:str; ParamList:TStrings; ReadBackslash:bool)
10780:  Func StrToCmdLineParam( const Param :Str) :Str
10781:  Func MergeCmdLineParams( ParamList : TStrings) :Str
10782:  Proc InvalidateFileStateCache( const Filename :Str)
10783:  Func FindAllFiles(const SearchPath:str;SearchMask:Str;SearchSubDirs:Bool):TStringList);
10784:  Func FindAllDirectories(const SearchPath:str;SearchSubDirs:Bool):TStringList
10785:  Func FindAllDocs(const Root, extmask:Str): TStringlist;
10786:  Func ReadFileToString( const Filename:Str):Str
10787:  Proc Inc1(var X: longint; N: Longint);
10788:  type TCopyFileFlag = ( cffOverwriteFile,
10789:     cffCreateDestDirectory, cffPreserveTime );
10790:   TCopyFileFlags = set of TCopyFileFlag;*)
10791:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10792:   TCopyFileFlags', 'set of TCopyFileFlag
10793:   Func CopyDirTree(const SourceDir,TargetDir:str;Flags:TCopyFileFlags):Bool
10794: end;
10795:
10796: Proc SIRegister_lazMasks(CL: TPSPascalCompiler);
10797: begin
10798:   TMaskCharType','( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10799:   SIRegister_TMask(CL);
10800:   SIRegister_TParseStringList(CL);
10801:   SIRegister_TMaskList(CL);
10802:  Func MatchesMask(const FileName,Mask:str;const CaseSensitive:Bool):Bool
10803:  Func MatchesWindowsMask(const FileName,Mask:str;const CaseSensitive:Bool): Bool;
10804:  Func MatchesMaskList(const FileName,Mask:Str;Separatr:Char;const CaseSensitive:Bool):Bool;
10805:  Func MatchesWindowsMaskList(const FileName,Mask:str;Separat:Char;const CaseSensitive:Bool):Bool;
10806: end;
10807:
10808: Proc SIRegister_JvShellHook(CL: TPSPascalCompiler);
10809: begin
10810: //PShellHookInfo', '^TShellHookInfo // will not work
10811:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10812:   SHELLHOOKINFO', 'TShellHookInfo
10813:   LPSHELLHOOKINFO', 'PShellHookInfo
10814:   TJvShellHookEvent', 'Proc ( Sender : TObject; var Msg : TMessage)
10815:   SIRegister_TJvShellHook(CL);
10816:  Func InitJvShellHooks :Bool
10817:  Proc UnInitJvShellHooks
10818: end;
10819:
10820: Proc SIRegister_JvExControls(CL: TPSPascalCompiler);
10821: begin
10822:  TDlgCode','(dcWantAllKeys,dcWantArrows,dcWantChars,dcButton,dcHasSetSel,dcWantTab,dcNative)
10823:   TDlgCodes', 'set of TDlgCodedcWantMessage',' dcWantAllKeys);
10824:   SIRegister_IJvExControl(CL);
10825:   SIRegister_IJvDenySubClassing(CL);
10826:   SIRegister_TStructPtrMessage(CL);
10827:  Proc SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10828:  Proc DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl :Bool);
10829:  Proc DrawDotNetControl1(DC:HDC; R:TRect;AColor:TColor;UseFocusedColor:Bool);
10830:  Proc HandleDotNetHighlighting(Cntrl:TWinControl;const Msg:TMessage;MouseOver:Bool;Color:TColor);
10831:  Func CreateWMMessage( Msg : Int; WParam : Int; LParam : Longint) : TMessage;
10832:  Func CreateWMMessage1( Msg : Int; WParam : Int; LParam : TControl) : TMessage;
10833:  Func SmallPointToLong( const Pt : TSmallPoint) : Longint
10834:  Func ShiftStateToKeyData( Shift : TShiftState) : Longint
10835:  Func GetFocusedControl( AControl : TControl) : TWinControl
10836:  Func DlgcToDlgCodes( Value : Longint) : TDlgCodes
10837:  Func DlgCodesToDlgc( Value : TDlgCodes) : Longint
10838:  Proc GetHintColor(var HintInfo:THintInfo AControl:TControl;HintColor:TColor)
10839:  Func DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) :Bool
10840:   SIRegister_TJvExControl(CL);
```

```
10841:   SIRegister_TJvExWinControl(CL);
10842:   SIRegister_TJvExCustomControl(CL);
10843:   SIRegister_TJvExGraphicControl(CL);
10844:   SIRegister_TJvExHintWindow(CL);
10845:   SIRegister_TJvExPubGraphicControl(CL);
10846: end;
10847:
10848: (*----------------------------------------------------------------------------*)
10849: Proc SIRegister_EncdDecd(CL: TPSPascalCompiler);
10850: begin
10851:  Proc EncodeStream(Input, Output : TStream)
10852:  Proc DecodeStream(Input, Output : TStream)
10853:  Func EncodeString1(const Input :Str) :Str
10854:  Func DecodeString1(const Input :Str) :Str
10855: end;
10856:
10857: (*----------------------------------------------------------------------------*)
10858: Proc SIRegister_SockAppReg(CL: TPSPascalCompiler);
10859: begin
10860:   SIRegister_TWebAppRegInfo(CL);
10861:   SIRegister_TWebAppRegList(CL);
10862:  Proc GetRegisteredWebApps( AList : TWebAppRegList)
10863:  Proc RegisterWebApp( const AFileName, AProgID :Str)
10864:  Proc UnregisterWebApp( const AProgID :Str)
10865:  Func FindRegisteredWebApp( const AProgID :Str) :Str
10866:  Func CreateRegistry(InitializeNewFile:Bool):TCustomIniFilesUDPPort',String 'UDPPort
10867: end;
10868:
10869: Proc SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10870: begin
10871:  // TStringDynArray', 'array of string
10872:  Func GetEnvVarValue( const VarName :Str) :Str
10873:  Func SetEnvVarValue( const VarName, VarValue :Str) : Int
10874:  Func DeleteEnvVar( const VarName :Str) : Int
10875:  Func CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:str;const
       BufSize:Int):Int;
10876:  Func ExpandEnvVars( const Str :Str) :Str
10877:  Func GetAllEnvVars( const Vars : TStrings) : Int
10878:  Proc GetAllEnvVarNames( const Names : TStrings);
10879:  Func GetAllEnvVarNames1 : TStringDynArray;
10880:  Func EnvBlockSize : Int
10881:   TPJEnvVarsEnum', 'Proc ( const VarName :Str; Data : TObject)
10882:   SIRegister_TPJEnvVarsEnumerator(CL);
10883:   SIRegister_TPJEnvVars(CL);
10884:   FindClass('TOBJECT'),'EPJEnvVars
10885:   FindClass('TOBJECT'),'EPJEnvVars
10886:  //Proc Register
10887: end;
10888:
10889: (*----------------------------------------------------------------------------*)
10890: Proc SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10891: begin
10892:  'cOneSecInMS','LongInt'( 1000);
10893:  //'cDefTimeSlice','LongInt'( 50);
10894:  //'cDefMaxExecTime',' cOneMinInMS);
10895:  'cAppErrorMask','LongInt'( 1 shl 29);
10896:  Func IsApplicationError( const ErrCode : LongWord) :Bool
10897:   TPJConsoleAppPriority','( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10898:   TPJConsoleColors','record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10899:  Func MakeConsoleColors(const AForeground,ABackground:TPJConsoleColor):TPJConsoleColors;
10900:  Func MakeConsoleColors1(const AForeground,ABackground:TColor): TPJConsoleColors;
10901:  Func MakeConsoleColors2(const AForeground ABackground:TAlphaColor): TPJConsoleColors;
10902:  Func MakeSize( const ACX, ACY : LongInt) : TSize
10903:   SIRegister_TPJCustomConsoleApp(CL);
10904:   SIRegister_TPJConsoleApp(CL);
10905: end;
10906:
10907: Proc SIRegister_ip_misc(CL: TPSPascalCompiler);
10908: begin
10909:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10910:   t_encoding', '( uuencode, base64, mime )
10911:  Func internet_date( date : TDateTime) :Str
10912:  Func lookup_hostname( const hostname :Str) : longint
10913:  Func my_hostname :Str
10914:  Func my_ip_address : longint
10915:  Func ip2string( ip_address : longint) :Str
10916:  Func resolve_hostname( ip : longint) :Str
10917:  Func address_from( const s :Str; count : Int) :Str
10918:  Func encode_base64( data : TStream) : TStringList
10919:  Func decode_base64( source : TStringList) : TMemoryStream
10920:  Func posn( const s, t :Str; count : Int) : Int
10921:  Func poscn( c : char; const s :Str; n : Int) : Int
10922:  Func filename_of( const s :Str) :Str
10923:  //Func trim( const s :Str) :Str
10924:  //Proc setlength( var s :Str; l : byte)
10925:  Func TimeZoneBias : longint
10926:  Func eight2seven_quoteprint( const s :Str) :Str
10927:  Func eight2seven_german( const s :Str) :Str
10928:  Func seven2eight_quoteprint( const s :Str) :Str end;
```

```
10929:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10930:  Func socketerror : cint
10931:  Func fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10932:  Func fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint): ssize_t
10933:  Func fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint): ssize_t
10934:  //Func fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10935:  Func fplisten( s : cint; backlog : cint) : cint
10936:  //Func fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10937:  //Func fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10938:  //Func fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10939:  Func NetAddrToStr( Entry : in_addr) :Str
10940:  Func HostAddrToStr( Entry : in_addr) :Str
10941:  Func StrToHostAddr( IP :Str) : in_addr
10942:  Func StrToNetAddr( IP :Str) : in_addr
10943:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10944:   cint8', 'shortint
10945:   cuint8', 'byte
10946:   cchar', 'cint8
10947:   cschar', 'cint8
10948:   cuchar', 'cuint8
10949:   cint16', 'smallint
10950:   cuint16', 'word
10951:   cshort', 'cint16
10952:   csshort', 'cint16
10953:   cushort', 'cuint16
10954:   cint32', 'longint
10955:   cuint32', 'longword
10956:   cint', 'cint32
10957:   csint', 'cint32
10958:   cuint', 'cuint32
10959:   csigned', 'cint
10960:   cunsigned', 'cuint
10961:   cint64', 'int64
10962:   clonglong', 'cint64
10963:   cslonglong', 'cint64
10964:   cbool', 'longbool
10965:   cfloat', 'single
10966:   cdouble', 'double
10967:   clongdouble', 'extended
10968:
10969: Proc SIRegister_uLkJSON(CL: TPSPascalCompiler);
10970: begin
10971:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10972:   SIRegister_TlkJSONdotnetclass(CL);
10973:   SIRegister_TlkJSONbase(CL);
10974:   SIRegister_TlkJSONnumber(CL);
10975:   SIRegister_TlkJSONstring(CL);
10976:   SIRegister_TlkJSONboolean(CL);
10977:   SIRegister_TlkJSONnull(CL);
10978:   TlkJSONFuncEnum','Procedure(ElName:str;Elem:TlkJSONbase;data:TObject;var Continue:Boolean)
10979:   SIRegister_TlkJSONcustomlist(CL);
10980:   SIRegister_TlkJSONlist(CL);
10981:   SIRegister_TlkJSONobjectmethod(CL);
10982:   TlkHashItem', 'record hash :Card; index : Int; end
10983:   TlkHashFunction', 'Func ( const ws : WideString) :Card
10984:   SIRegister_TlkHashTable(CL);
10985:   SIRegister_TlkBalTree(CL);
10986:   SIRegister_TlkJSONobject(CL);
10987:   SIRegister_TlkJSON(CL);
10988:   SIRegister_TlkJSONstreamed(CL);
10989:  Func GenerateReadableText( vObj : TlkJSONbase; var vLevel : Int):Str
10990: end;
10991:
10992: Ex: js:= TlkJSON.create;
10993:     jsonObject:= js.ParseText(jsonstring) as TlkJSONobject;
10994:   try
10995:     writeln('get message: '+jsonObject.Field['amessage'].value);
10996:     writeln(vartostr(jsonobject.Field['book'].Child[3].field['ISBN'].value));
10997:
10998: Proc SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10999: begin
11000:   TZListSortCompare', 'Func (Item1, Item2 : TObject): Int
11001:   SIRegister_TZSortedList(CL);
11002:  Func zFirstDelimiter( const Delimiters, Str :Str) : Int
11003:  Func zLastDelimiter( const Delimiters, Str :Str) : Int
11004:  //Func MemLCompUnicode( P1, P2 : PWideChar; Len : Int) :Bool
11005:  //Func MemLCompAnsi( P1, P2 : PAnsiChar; Len : Int) :Bool
11006:  Func zStartsWith( const Str, SubStr : WideString) :Bool;
11007:  Func StartsWith1( const Str, SubStr : RawByteString) :Bool;
11008:  Func EndsWith( const Str, SubStr : WideString) :Bool;
11009:  Func EndsWith1( const Str, SubStr : RawByteString) :Bool;
11010:  Func SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
11011:  Func SQLStrToFloatDef1( Str :Str; Def : Extended) : Extended;
11012:  Func SQLStrToFloat( const Str : Ansistr) : Extended
11013:  //Func BufferToStr( Buffer : PWideChar; Length : LongInt) :Str;
11014:  //Func BufferToStr1( Buffer : PAnsiChar; Length : LongInt) :Str;
11015:  Func BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
11016:  Func StrToBoolEx( Str :Str) :Bool
11017:  Func BoolToStrEx( Bool :Bool) :Str
```

```
11018:  Func IsIpAddr( const Str :Str) :Bool  //IsIP()
11019:  Func zSplitString( const Str, Delimiters :Str) : TStrings
11020:  Proc PutSplitString( List : TStrings; const Str, Delimiters :Str)
11021:  Proc AppendSplitString( List : TStrings; const Str, Delimiters :Str)
11022:  Func ComposeString( List : TStrings; const Delimiter :Str) :Str
11023:  Func FloatToSQLStr( Value : Extended) :Str
11024:  Proc PutSplitStringEx( List : TStrings; const Str, Delimiter :Str)
11025:  Func SplitStringEx( const Str, Delimiter :Str) : TStrings
11026:  Proc AppendSplitStringEx( List : TStrings; const Str, Delimiter:Str)
11027:  Func zBytesToStr( const Value : TByteDynArray) : Ansistr
11028:  Func zStrToBytes( const Value : Ansistr) : TByteDynArray;
11029:  Func StrToBytes1( const Value : UTF8String) : TByteDynArray;
11030:  Func StrToBytes2( const Value : RawByteString) : TByteDynArray;
11031:  Func StrToBytes3( const Value : WideString) : TByteDynArray;
11032:  Func StrToBytes4( const Value : UnicodeString) : TByteDynArray;
11033:  Func BytesToVar( const Value : TByteDynArray) : Variant
11034:  Func VarToBytes( const Value : Variant) : TByteDynArray
11035:  Func AnsiSQLDateToDateTime( const Value :Str) : TDateTime
11036:  Func TimestampStrToDateTime( const Value :Str) : TDateTime
11037:  Func DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec:Bool):Str
11038:  Func EncodeCString( const Value :Str) :Str
11039:  Func DecodeCString( const Value :Str) :Str
11040:  Func zReplaceChar( const Source, Target : Char; const Str:Str):Str
11041:  Func MemPas( Buffer : PChar; Length : LongInt) :Str
11042:  Proc DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
11043:  Func EncodeSQLVersioning(const MajorVersion:Int;const MinorVersion:Int;const SubVersion:Int):Int;
11044:  Func FormatSQLVersion( const SQLVersion : Int) :Str
11045:  Func ZStrToFloat( Value : AnsiChar) : Extended;
11046:  Func ZStrToFloat1( Value : Ansistr) : Extended;
11047:  Proc ZSetString( const Src : AnsiChar; var Dest : Ansistr);
11048:  Proc ZSetString1(const Src:AnsiChar;const Len:Card;var Dest:Ansistr);
11049:  Proc ZSetString2(const Src : AnsiChar; var Dest : UTF8String);
11050:  Proc ZSetString3(const Src:AnsiChar;const Len:Card;var Dest:UTF8String);
11051:  Proc ZSetString4(const Src:AnsiChar;const Len:Card;var Dest:WideString);
11052:  Proc ZSetString5(const Src : AnsiChar; var Dest : RawByteString);
11053:  Proc ZSetString6(const Src:AnsiChar;const Len:Card;var Dest:RawByteString);
11054: end;
11055:
11056: unit uPSI_ZEncoding;
11057:  Func StringToAnsiEx( const s :Str; const FromCP, ToCP : Word) : RawByteString
11058:  Func AnsiToStringEx(const s:RawByteString;const FromCP, ToCP : Word :Str
11059:  Func ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
11060:  Func ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
11061:  Func ZConvertAnsiToRaw( const Src : Ansistr; const RawCP : Word) : RawByteString
11062:  Func ZConvertRawToAnsi(const Src:RawByteString;const RawCP:Word):Ansistr
11063:  Func ZConvertAnsiToUTF8( const Src : Ansistr) : UTF8String
11064:  Func ZConvertUTF8ToAnsi( const Src : UTF8String) : Ansistr
11065:  Func ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
11066:  Func ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
11067:  Func ZConvertRawToString(const Src:RawByteString;const RawCP,StringCP:Word):str
11068:  Func ZConvertStringToRaw( const Src :Str; const StringCP, RawCP : Word) : RawByteString
11069:  Func ZConvertStringToRawWithAutoEncode(const Src:str;const StringCP,RawCP:Word):RawByteString;
11070:  Func ZConvertUTF8ToString(const Src:UTF8String;const StringCP: Word):Str
11071:  Func ZConvertStringToUTF8(const Src:Str;const StringCP:Word):UTF8String
11072:  Func ZConvertStringToUTF8WithAutoEncode(const Src:str;const StringCP:Word): UTF8String
11073:  Func ZConvertStringToAnsi( const Src :Str; const StringCP : Word) : Ansistr
11074:  Func ZConvertStringToAnsiWithAutoEncode(const Src:str;const StringCP:Word): Ansistr
11075:  Func ZConvertAnsiToString(const Src:Ansistr;const StringCP:Word):str
11076:  Func ZConvertUnicodeToString(const Src:WideString;const StringCP:Word):str
11077:  Func ZConvertUnicodeToString_CPUTF8(const Src:WideString; const StringCP : Word):Str
11078:  Func ZConvertStringToUnicode( const Src :Str; const StringCP : Word) : WideString
11079:  Func ZConvertString_CPUTF8ToUnicode(const Src:str;const StringCP:Word): WideString
11080:  Func ZConvertStringToUnicodeWithAutoEncode(const Src:str;const StringCP:Word):WideString
11081:  Func ZMoveAnsiToRaw( const Src : Ansistr; const RawCP : Word) : RawByteString
11082:  Func ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : Ansistr
11083:  Func ZMoveAnsiToUTF8( const Src : Ansistr) : UTF8String
11084:  Func ZMoveUTF8ToAnsi( const Src : UTF8String) : Ansistr
11085:  Func ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
11086:  Func ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
11087:  Func ZMoveStringToAnsi( const Src :Str; const StringCP : Word) : Ansistr
11088:  Func ZMoveAnsiToString( const Src : Ansistr; const StringCP : Word) :Str
11089:  Func ZMoveRawToString(const Src:RawByteString;const RawCP,StringCP:Word):Str
11090:  Func ZMoveStringToRaw( const Src :Str; const StringCP, RawCP : Word) : RawByteString
11091:  Func ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) :Str
11092:  Func ZMoveStringToUTF8( const Src :Str; const StringCP : Word) : UTF8String
11093:  Func ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
11094:  Func ZUnknownRawToUnicodeWithAutoEncode(const S:RawByteString;const CP:Word): WideString
11095:  Func ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
11096:  Func ZDefaultSystemCodePage : Word
11097:  Func ZCompatibleCodePages( const CP1, CP2 : Word) :Bool
11098:  Func MPing(const AHost:Str;const ATimes:Int; out AvgMS:Double):Boolean;
11099:
11100: Proc SIRegister_BoldComUtils(CL: TPSPascalCompiler);
11101: begin
11102:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
11103:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
11104:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
11105:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
```

```
11106:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
11107:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
11108:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
11109:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
11110:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
11111:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
11112:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
11113:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
11114:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
11115:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
11116:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
11117:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
11118:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
11119:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
11120:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
11121:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
11122:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
11123:  ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
11124:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
11125:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
11126:  ('EOAC_NONE','LongWord').SetUInt( $0);
11127:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
11128:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
11129:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
11130:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
11131:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
11132:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
11133:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
11134:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
11135:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
11136:   FindClass('TOBJECT'),'EBoldCom
11137:  Func BoldVariantIsType( V : OleVariant; TypeCode : Int) :Bool
11138:  Func BoldMemoryToVariant( const Buffer, BufSize : Int) : OleVariant
11139:  Func BoldStreamToVariant( Stream : TStream) : OleVariant
11140:  Func BoldStringsToVariant( Strings : TStrings) : OleVariant
11141:  Func BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Int) : Int
11142:  Func BoldVariantToStream( V : OleVariant; Stream : TStream) : Int
11143:  Func BoldVariantArrayOfArraysOfStringToStrings(V:OleVariant;Strings: TStrings) : Int
11144:  Func BoldVariantIsNamedValues( V : OleVariant) :Bool
11145:  Func BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11146:  Func BoldGetNamedValue( Data : OleVariant; const Name :Str) : OleVariant
11147:  Proc BoldSetNamedValue( Data : OleVariant; const Name :Str; Value : OleVariant)
11148:  Func BoldCreateGUID : TGUID
11149:  Func BoldCreateComObject( const ClsId,IId:TGUID; out Obj:variant;out Res:HResult):Bool
11150:  Func BoldCreateRemoteComObject(const HostName:str;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
11151:  Proc BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11152:  Proc BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11153: end;
11154:
11155: (*----------------------------------------------------------------------------*)
11156: Proc SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11157: begin
11158:  Func ParseISODate( s :Str) : TDateTime
11159:  Func ParseISODateTime( s :Str) : TDateTime
11160:  Func ParseISOTime( str :Str) : TDateTime
11161: end;
11162:
11163: (*----------------------------------------------------------------------------*)
11164: Proc SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11165: begin
11166:  Func BoldCreateGUIDAsString( StripBrackets :Bool) :Str
11167:  Func BoldCreateGUIDWithBracketsAsString :Str
11168: end;
11169:
11170: Proc SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11171: begin
11172:   FindClass('TOBJECT'),'TBoldFileHandler
11173:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11174:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11175:   TBoldInitializeFileContents', 'Proc ( StringList : TStringList)
11176:   SIRegister_TBoldFileHandler(CL);
11177:   SIRegister_TBoldDiskFileHandler(CL);
11178:  Proc BoldCloseAllFilehandlers
11179:  Proc BoldRemoveUnchangedFilesFromEditor
11180:  Func BoldFileHandlerList : TBoldObjectArray
11181:  Func BoldFileHandlerForFile(path,
       FileName:str;ModuleType:TBoldModuleType;ShowInEditor:Bool;OnInitializeFileContents:TBoldInitializeFileContents):
       TBoldFileHandler
11182: end;
11183:
11184: Proc SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11185: begin
11186:   PCharArr', 'array of PChar
11187:  Func BoldInternetOpen(Agent:Str;AccessType:Int;Proxy:str;ProxyByPass:Str;Flags:Int):ptr);
11188:  Func BoldInternetOpenUrl(iNet:Poin;URL:str;Headers:str;Flags,Context:Card):Point
11189:  Func BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11190:  Func BoldInternetCloseHandle( HINet : Pointer) : LongBool
```

```
11191:  Func BoldHttpQueryInfo(hRequest:Pointer;InfoLevel:Card;Buffer:Pointer;BufferLength:Card;Reserved:Card) :
       LongBool
11192:  Func BoldInternetQueryDataAvailable(hFile:Pointer;var NumberOfBytesAvailable:Card; flags:Card;
       Context:Card): LongBool
11193:  Func BoldHttpOpenRequest(hConnect:Pointer;Verb,ObjectName,Version,Referrer:str;AcceptTypes:PCharArr;Flags,
       Context:Card):Pointer
11194:  Func BoldHttpSendRequest(hRequest:Ptr;Headers:str;Optional:Ptr;OptionalLength:Cardi): LongBool
11195:  Func BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11196:  Func BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11197:  Func BoldInternetConnect(hInet: HINTERNET;ServerName:str; nServerPort:INTERNET_PORT; Username:str;
       Password :Str; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11198:  Func BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11199: end;
11200:
11201: Proc SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11202: begin
11203:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11204:   SIRegister_TfrmBoldQueryUser(CL);
11205:  Func QueryUser( const Title, Query :Str) : TBoldQueryResult
11206: end;
11207:
11208: (*----------------------------------------------------------------------------*)
11209: Proc SIRegister_BoldQueue(CL: TPSPascalCompiler);
11210: begin
11211:  //('befIsInDisplayList',' BoldElementFlag0);
11212:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11213:  //('befFollowerSelected',' BoldElementFlag2);
11214:   FindClass('TOBJECT'),'TBoldQueue
11215:   FindClass('TOBJECT'),'TBoldQueueable
11216:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11217:   SIRegister_TBoldQueueable(CL);
11218:   SIRegister_TBoldQueue(CL);
11219:  Func BoldQueueFinalized :Bool
11220:  Func BoldInstalledQueue : TBoldQueue
11221: end;
11222:
11223: Proc SIRegister_Barcode(CL: TPSPascalCompiler);
11224: begin
11225:  const mmPerInch','Extended).setExtended( 25.4);
11226:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11227:    + bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11228:    +Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11229:    +bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11230:    +odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11231:   TBarLineType', '( white, black, black_half )
11232:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11233:   TShowTextPosition','(stpTopLeft,stpTopRight,stpTopCenter,stpBottomLeft,stpBottomRight, stpBottomCenter )
11234:   TCheckSumMethod', '( csmNone, csmModulo10 )
11235:   SIRegister_TAsBarcode(CL);
11236:  Func CheckSumModulo10( const data :Str) :Str
11237:  Func ConvertMmToPixelsX( const Value : Double) : Int
11238:  Func ConvertMmToPixelsY( const Value : Double) : Int
11239:  Func ConvertInchToPixelsX( const Value : Double) : Int
11240:  Func ConvertInchToPixelsY( const Value : Double) : Int
11241: end;
11242:
11243: Proc SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11244: begin
11245:   TVector3f = array[0..2] of single;
11246:   CL.AddTypeS('TAffineVector', 'TVector3f
11247:   CL.AddTypeS('TVertex', 'TAffineVector
11248:   THomogeneousByteVector', 'array[0..3] of Byte
11249:   THomogeneousWordVector', 'array[0..3] of Word
11250:   THomogeneousIntVector', 'array[0..3] of Int
11251:   THomogeneousFltVector', 'array[0..3] of single
11252:   THomogeneousDblVector', 'array[0..3] of double
11253:   THomogeneousExtVector', 'array[0..3] of extended
11254:   TAffineByteVector', 'array[0..2] of Byte
11255:   TAffineWordVector', 'array[0..2] of Word
11256:   TAffineIntVector', 'array[0..2] of Int
11257:   TAffineFltVector', 'array[0..2] of single
11258:   TAffineDblVector', 'array[0..2] of double
11259:   TAffineExtVector', 'array[0..2] of extended
11260:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11261:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11262:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11263:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11264:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11265:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11266:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11267:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11268:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11269:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11270:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11271:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11272:   TMatrix4b', 'THomogeneousByteMatrix
11273:   TMatrix4w', 'THomogeneousWordMatrix
11274:   TMatrix4i', 'THomogeneousIntMatrix
11275:   TMatrix4f', 'THomogeneousFltMatrix
```

```
11276:   TMatrix4d', 'THomogeneousDblMatrix
11277:   TMatrix4e', 'THomogeneousExtMatrix
11278:   TMatrix3b', 'TAffineByteMatrix
11279:   TMatrix3w', 'TAffineWordMatrix
11280:   TMatrix3i', 'TAffineIntMatrix
11281:   TMatrix3f', 'TAffineFltMatrix
11282:   TMatrix3d', 'TAffineDblMatrix
11283:   TMatrix3e', 'TAffineExtMatrix
11284:   //'PMatrix', '^TMatrix // will not work
11285:   TMatrixGL', 'THomogeneousFltMatrix
11286:   THomogeneousMatrix', 'THomogeneousFltMatrix
11287:   TAffineMatrix', 'TAffineFltMatrix
11288:   TQuaternion', 'record Vector : TVector4f; end
11289:   TRectangle', 'record Left : Int; Top : Int; Width : integer; Height : Int; end
11290:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11291:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11292:    +',ttTranslateZ,ttPerspectiveX,ttPerspectiveY,ttPerspectiveZ, ttPerspectiveW )
11293:  'EPSILON','Extended').setExtended( 1E-100);
11294:  'EPSILON2','Extended').setExtended( 1E-50);
11295:  Func VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11296:  Func VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11297:  Func VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11298:  Func VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11299:  Func VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11300:  Func VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11301:  Func VectorAngle( V1, V2 : TAffineVector) : Single
11302:  Func VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11303:  Func VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11304:  Func VectorDotProduct( V1, V2 : TVectorGL) : Single
11305:  Func VectorLength( V : array of Single) : Single
11306:  Func VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11307:  Proc VectorNegate( V : array of Single)
11308:  Func VectorNorm( V : array of Single) : Single
11309:  Func VectorNormalize( V : array of Single) : Single
11310:  Func VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11311:  Func VectorReflect( V, N : TAffineVector) : TAffineVector
11312:  Proc VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11313:  Proc VectorScale( V : array of Single; Factor : Single)
11314:  Func VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11315:  Func CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11316:  Func CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11317:  Func CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11318:  Func CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11319:  Func CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11320:  Proc MatrixAdjoint( var M : TMatrixGL)
11321:  Func MatrixAffineDeterminant( M : TAffineMatrix) : Single
11322:  Proc MatrixAffineTranspose( var M : TAffineMatrix)
11323:  Func MatrixDeterminant( M : TMatrixGL) : Single
11324:  Proc MatrixInvert( var M : TMatrixGL)
11325:  Func MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11326:  Proc MatrixScale( var M : TMatrixGL; Factor : Single)
11327:  Proc MatrixTranspose( var M : TMatrixGL)
11328:  Func QuaternionConjugate( Q : TQuaternion) : TQuaternion
11329:  Func QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11330:  Func QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11331:  Func QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Int; t:Single):TQuaternion
11332:  Func QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11333:  Proc QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11334:  Func ConvertRotation( Angles : TAffineVector) : TVectorGL
11335:  Func CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11336:  //Func MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) :Bool
11337:  Func VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11338:  Func VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11339:  Func VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11340:  Func MakeAffineDblVector( V : array of Double) : TAffineDblVector
11341:  Func MakeDblVector( V : array of Double) : THomogeneousDblVector
11342:  Func MakeAffineVector( V : array of Single) : TAffineVector
11343:  Func MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11344:  Func MakeVector( V : array of Single) : TVectorGL
11345:  Func PointInPolygonGL( xp, yp : array of Single; x, y : Single) :Bool
11346:  Func VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11347:  Func VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11348:  Func VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11349:  Func VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11350:  Func ArcCosGL( X : Extended) : Extended
11351:  Func ArcSinGL( X : Extended) : Extended
11352:  Func ArcTan2GL( Y, X : Extended) : Extended
11353:  Func CoTanGL( X : Extended) : Extended
11354:  Func DegToRadGL( Degrees : Extended) : Extended
11355:  Func RadToDegGL( Radians : Extended) : Extended
11356:  Proc SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11357:  Func TanGL( X : Extended) : Extended
11358:  Func Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11359:  Func Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11360:  Func Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11361:  Func Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11362:  Func Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11363:  Func Roll1(Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11364: end;
```

```
11365:
11366: Proc SIRegister_JclRegistry(CL: TPSPascalCompiler);
11367: begin
11368:  Func RegCreateKey( const RootKey : HKEY; const Key, Value :Str) : Longint
11369:  Func RegDeleteEntry( const RootKey : HKEY; const Key, Name :Str) :Bool
11370:  Func RegDeleteKeyTree( const RootKey : HKEY; const Key :Str) :Bool
11371:  Func RegReadBool( const RootKey : HKEY; const Key, Name :Str) :Bool
11372:  Func RegReadBoolDef(const RootKey:HKEY; const Key, Name:Str; Def:Bool) :Bool
11373:  Func RegReadInt( const RootKey : HKEY; const Key, Name :Str) : Int
11374:  Func RegReadIntDef( const RootKey : HKEY; const Key, Name :Str; Def : Int) : Int
11375:  Func RegReadString( const RootKey : HKEY; const Key, Name :Str) :Str
11376:  Func RegReadStringDef( const RootKey : HKEY; const Key, Name, Def :Str) :Str
11377:  Func RegReadDWORD( const RootKey : HKEY; const Key, Name :Str) : Int64
11378:  Func RegReadDWORDDef(const RootKey: HKEY; const Key, Name :Str; Def:int64) : Int64
11379:  Proc RegWriteBool( const RootKey : HKEY; const Key, Name :Str; Value :Bool)
11380:  Proc RegWriteInt( const RootKey : HKEY; const Key, Name :Str; Value : Int)
11381:  Proc RegWriteString( const RootKey : HKEY; const Key, Name, Value :Str)
11382:  Proc RegWriteDWORD( const RootKey : HKEY; const Key, Name :Str; Value : Int64)
11383:  Func RegGetValueNames(const RootKey:HKEY;const Key:Str; const List : TStrings): Bool
11384:  Func RegGetKeyNames( const RootKey: HKEY;const Key:Str; const List:TStrings):Bool
11385:  Func RegHasSubKeys( const RootKey : HKEY; const Key :Str) :Bool
11386:  Func RegKeyExists( const RootKey : HKEY; const Key :Str) :Bool
11387:   AddTypeS('TExecKind',(ekMachineRun,ekMachineRunOnce,ekUserRun,ekUserRunOnce,ekServiceRun,
      ekServiceRunOnce )
11388:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11389:  Func UnregisterAutoExec( ExecKind : TExecKind; const Name :Str) :Bool
11390:  Func RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline :Str) :Bool
11391:  Func RegSaveList(const RootKey:HKEY;const Key:str;const ListName:str;const Items:TStrings):Bool;
11392:  Func RegLoadList(const RootKey:HKEY;const Key:str;const ListName:str;const SaveTo:TStrings):Bool;
11393:  Func RegDelList( const RootKey:HKEY;const Key:str;const ListName:str): Bool
11394: end;
11395:
11396: Proc SIRegister_JclCOM(CL: TPSPascalCompiler);
11397: begin
11398:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11399:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11400:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11401:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11402:   FindClass('TOBJECT'),'EInvalidParam
11403:  Func IsDCOMInstalled :Bool
11404:  Func IsDCOMEnabled :Bool
11405:  Func GetDCOMVersion :Str
11406:  Func GetMDACVersion :Str
11407:  Func GetMDACVersion2:Str
11408:  Func MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11409:  Func MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11410:  Func MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11411:  Func MarshalInterMachineInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11412:  Func MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11413:  Func CreateComponentCategory( const CatID:TGUID; const sDescription:str): HResult
11414:  Func RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID): HResult
11415:  Func UnRegisterCLSIDInCategory(const ClassID: TGUID; const CatID : TGUID): HResult
11416:  Func ResetIStreamToStart( Stream : IStream) :Bool
11417:  Func SizeOfIStreamContents( Stream : IStream) : Largeint
11418:  Func StreamToVariantArray( Stream : TStream) : OleVariant;
11419:  Func StreamToVariantArray1( Stream : IStream) : OleVariant;
11420:  Proc VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11421:  Proc VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11422: end;
11423:
11424: Proc SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11425: begin
11426:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11427:  FahrenheitFreezingPoint','Extended).setExtended( 32.0);
11428:  KelvinFreezingPoint','Extended).setExtended( 273.15);
11429:  CelsiusAbsoluteZero','Extended).setExtended( - 273.15);
11430:  FahrenheitAbsoluteZero','Extended).setExtended( - 459.67);
11431:  KelvinAbsoluteZero','Extended).setExtended( 0.0);
11432:  DegPerCycle','Extended).setExtended( 360.0);
11433:  DegPerGrad','Extended).setExtended( 0.9);
11434:  DegPerRad','Extended).setExtended( 57.295779513082320876798154814105);
11435:  GradPerCycle','Extended).setExtended( 400.0);
11436:  GradPerDeg','Extended).setExtended( 1.1111111111111111111111111111111);
11437:  GradPerRad','Extended).setExtended( 63.661977236758134307553505349006);
11438:  RadPerCycle','Extended).setExtended( 6.2831853071795864769252867665559);
11439:  RadPerDeg','Extended).setExtended( 0.017453292519943295769236907684886);
11440:  RadPerGrad','Extended).setExtended( 0.015707963267948966192313216916398);
11441:  CyclePerDeg','Extended).setExtended( 0.0027777777777777777777777777777778);
11442:  CyclePerGrad','Extended).setExtended( 0.0025);
11443:  CyclePerRad','Extended).setExtended( 0.159154943091895335768883763372 51);
11444:  ArcMinutesPerDeg','Extended).setExtended( 60.0);
11445:  ArcSecondsPerArcMinute',Extended').setExtended( 60.0);
11446:  Func HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11447:  Func MakePercentage( const Step, Max : Longint) : Longint
11448:  Func CelsiusToKelvin( const T : double) : double
11449:  Func CelsiusToFahrenheit( const T : double) : double
11450:  Func KelvinToCelsius( const T : double) : double
11451:  Func KelvinToFahrenheit( const T : double) : double
11452:  Func FahrenheitToCelsius( const T : double) : double
```

```
11453:  Func FahrenheitToKelvin( const T : double) : double
11454:  Func CycleToDeg( const Cycles : double) : double
11455:  Func CycleToGrad( const Cycles : double) : double
11456:  Func CycleToRad( const Cycles : double) : double
11457:  Func DegToCycle( const Degrees : double) : double
11458:  Func DegToGrad( const Degrees : double) : double
11459:  Func DegToRad( const Degrees : double) : double
11460:  Func GradToCycle( const Grads : double) : double
11461:  Func GradToDeg( const Grads : double) : double
11462:  Func GradToRad( const Grads : double) : double
11463:  Func RadToCycle( const Radians : double) : double
11464:  Func RadToDeg( const Radians : double) : double
11465:  Func RadToGrad( const Radians : double) : double
11466:  Func DmsToDeg( const D, M : Int; const S : double) : double
11467:  Func DmsToRad( const D, M : Int; const S : double) : double
11468:  Proc DegToDms( const Degrees : double; out D, M : Int; out S : double)
11469:  Func DegToDmsStr( const Degrees : double; const SecondPrecision :Card):Str
11470:  Proc CartesianToPolar( const X, Y : double; out R, Phi : double)
11471:  Proc PolarToCartesian( const R, Phi : double; out X, Y : double)
11472:  Proc CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11473:  Proc CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11474:  Proc CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11475:  Proc SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11476:  Func CmToInch( const Cm : double) : double
11477:  Func InchToCm( const Inch : double) : double
11478:  Func FeetToMetre(const Feet : double) : double
11479:  Func MetreToFeet(const Metre : double) : double
11480:  Func YardToMetre(const Yard : double) : double
11481:  Func MetreToYard(const Metre : double) : double
11482:  Func NmToKm(const Nm : double) : double
11483:  Func KmToNm(const Km : double) : double
11484:  Func KmToSm(const Km : double) : double
11485:  Func SmToKm(const Sm : double) : double
11486:  Func LitreToGalUs( const Litre : double) : double
11487:  Func GalUsToLitre( const GalUs : double) : double
11488:  Func GalUsToGalCan( const GalUs : double) : double
11489:  Func GalCanToGalUs( const GalCan : double) : double
11490:  Func GalUsToGalUk( const GalUs : double) : double
11491:  Func GalUkToGalUs( const GalUk : double) : double
11492:  Func LitreToGalCan( const Litre : double) : double
11493:  Func GalCanToLitre( const GalCan : double) : double
11494:  Func LitreToGalUk( const Litre : double) : double
11495:  Func GalUkToLitre( const GalUk : double) : double
11496:  Func KgToLb( const Kg : double) : double
11497:  Func LbToKg( const Lb : double) : double
11498:  Func KgToOz( const Kg : double) : double
11499:  Func OzToKg( const Oz : double) : double
11500:  Func CwtUsToKg(const Cwt : double) : double
11501:  Func CwtUkToKg(const Cwt : double) : double
11502:  Func KaratToKg(const Karat : double) : double
11503:  Func KgToCwtUs(const Kg : double) : double
11504:  Func KgToCwtUk(const Kg : double) : double
11505:  Func KgToKarat(const Kg : double) : double
11506:  Func KgToSton(const Kg : double) : double
11507:  Func KgToLton(const Kg : double) : double
11508:  Func StonToKg(const STon : double) : double
11509:  Func LtonToKg(const Lton : double) : double
11510:  Func QrUsToKg(const Qr : double) : double
11511:  Func QrUkToKg(const Qr : double) : double
11512:  Func KgToQrUs(const Kg : double) : double
11513:  Func KgToQrUk(const Kg : double) : double
11514:  Func PascalToBar( const Pa : double) : double
11515:  Func PascalToAt( const Pa : double) : double
11516:  Func PascalToTorr( const Pa : double) : double
11517:  Func BarToPascal( const Bar : double) : double
11518:  Func AtToPascal( const At : double) : double
11519:  Func TorrToPascal( const Torr : double) : double
11520:  Func KnotToMs( const Knot : double) : double
11521:  Func HpElectricToWatt( const HpE : double) : double
11522:  Func HpMetricToWatt( const HpM : double) : double
11523:  Func MsToKnot( const ms : double) : double
11524:  Func WattToHpElectric( const W : double) : double
11525:  Func WattToHpMetric( const W : double) : double
11526:  Func getBigPI:Str; //PI of 1000 numbers
11527:
11528: Proc SIRegister_devcutils(CL: TPSPascalCompiler);
11529: begin
11530:  Func CDExecuteFile( const FileName, Params, DefaultDir :Str; ShowCmd : Int) : THandle
11531:  Proc CDCopyFile( const FileName, DestName :Str)
11532:  Proc CDMoveFile( const FileName, DestName :Str)
11533:  Func MakeCommaTextToColor( Text :Str; Index : Int; DefaultColor : TColor) : TColor
11534:  Proc CDDeleteFiles( Sender : TObject; s :Str)
11535:  Func CDGetTempDir :Str
11536:  Func CDGetFileSize( FileName :Str) : longint
11537:  Func GetFileTime( FileName :Str) : longint
11538:  Func GetShortName( FileName :Str) :Str
11539:  Func GetFullName( FileName :Str) :Str
11540:  Func WinReboot :Bool
11541:  Func WinDir :Str
```

```
11542:  Func RunFile( FileToRun :Str; Params :Str; Dir :Str; Wait :Bool) :Card
11543:  Func RunFile_( Cmd, WorkDir :Str; Wait :Bool) :Bool
11544:  Func devExecutor : TdevExecutor
11545: end;
11546:
11547: Proc SIRegister_FileAssocs(CL: TPSPascalCompiler);
11548: begin
11549: Proc CheckAssociations // AssociationsCount','LongInt'( 7);
11550: Proc Associate( Index : Int)
11551: Proc UnAssociate( Index : Int)
11552: Func IsAssociated( Index : Int) :Bool
11553: Func CheckFiletype(const extension,filetype,description, verb,serverapp:str):Bool
11554: Proc RegisterFiletype(const extension,filetype,description,verb,serverapp,IcoNum:str)
11555: Proc RegisterDDEServer( const filetype, verb, topic, servername, macro :Str)
11556: Proc RefreshIcons;
11557: Func GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11558: Func MergColor(Colors: Array of TColor): TColor;
11559: Func NewColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11560: Proc DimBitmap(ABitmap: TBitmap; Value: Int);
11561: Func GrayColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11562: Func GetInverseColor(AColor: TColor): TColor;
11563: Proc GrayBitmap(ABitmap: TBitmap; Value: Int);
11564: Proc DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: Int;ShadowColor: TColor);
11565: Proc DrawCheckMark(ACanvas: TCanvas; X, Y: Int);
11566: Proc GetSystemMenuFont(Font: TFont);
11567: end;
11568:
11569: //****************************unit uPSI_JvHLParser;*******************************
11570: Func IsStringConstant(const St:Str):Bool;
11571: Func IsIntConstant(const St:Str):Bool;
11572: Func IsRealConstant(const St:Str):Bool;
11573: Func IsIdentifier(const ID:Str):Bool;
11574: Func GetStringValue(const St:Str):Str;
11575: Proc ParseString(const S:Str; Ss: TStrings);
11576: Func IsStringConstantW(const St: WideString):Bool;
11577: Func IsIntConstantW(const St: WideString):Bool;
11578: Func IsRealConstantW(const St: WideString):Bool;
11579: Func IsIdentifierW(const ID: WideString):Bool;
11580: Func GetStringValueW(const St: WideString): WideString;
11581: Proc ParseStringW(const S: WideString; Ss: TStrings);
11582:
11583:
11584: //****************************unit uPSI_JclMapi;*******************************
11585: Func JclSimpleSendMail( const ARecipient,AName,ASubject, ABody :Str; const AAttachment : TFileName;
       ShowDialog :Bool; AParentWND : HWND) :Bool
11586: Func JclSimpleSendFax( const ARecipient, AName,ASubject,ABody:str;const AAttachment: TFileName; ShowDialog
       :Bool; AParentWND : HWND) :Bool
11587: Func JclSimpleBringUpSendMailDialog(const ASubject,ABody:str;const AAttach:TFileName;AParentWND:HWND):Bool
11588: Func MapiCheck( const Res : DWORD; IgnoreUserAbort :Bool) : DWORD
11589: Func MapiErrorMessage( const ErrorCode : DWORD) :Str
11590:
11591: Proc SIRegister_IdNTLM(CL: TPSPascalCompiler);
11592: begin
11593:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11594:  Func BuildType1Message( ADomain, AHost :Str) :Str
11595:  Func BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:str):str
11596:  Proc RegisterAuthenticationMethod(MethodName:str; AuthClass:TIdAuthenticationClass)
11597:  Func FindAuthClass( AuthName :Str) : TIdAuthenticationClass
11598: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11599: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11600: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11601: end;
11602:
11603: Proc SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11604: begin
11605: ('IpAny','LongWord').SetUInt( $00000000);
11606:  IpLoopBack','LongWord').SetUInt( $7F000001);
11607:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11608:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11609:  PortAny','LongWord( $0000);
11610:  SocketMaxConnections','LongInt'( 5);
11611:  TIpAddr', 'LongWord
11612:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4:Byte; end
11613:  Func HostToNetLong( HostLong : LongWord) : LongWord
11614:  Func HostToNetShort( HostShort : Word) : Word
11615:  Func NetToHostLong( NetLong : LongWord) : LongWord
11616:  Func NetToHostShort( NetShort : Word) : Word
11617:  Func StrToIp( Ip :Str) : TIpAddr
11618:  Func IpToStr( Ip : TIpAddr) :Str
11619: end;
11620:
11621: (*----------------------------------------------------------------------------*)
11622: Proc SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11623: begin
11624:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11625:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11626:    +'amSha1, AlsmtpClientAuthAutoSelect )
11627:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11628:   SIRegister_TAlSmtpClient(CL);
```

```
11629: end;
11630:
11631: Proc SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11632: begin
11633: TBitNo', 'IntTStByteNo', 'IntTStationNo','IntTInOutNo','IntTIo','(EE,AA,NE,NA )
11634: TBitSet', 'set of TBitNo
11635: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11636: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11637: TBitAddr', 'LongInt
11638: TByteAddrRec', 'record Kind : TAddrKind; ByteNo: Byte; end
11639: TByteAddr', 'SmallInt
11640: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11641:  Func BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11642:  Func BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11643:  Proc BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11644:  Func BitAddrToStr( Value : TBitAddr) :Str
11645:  Func StrToBitAddr( const Value :Str) : TBitAddr
11646:  Func ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11647:  Func BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11648:  Proc ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11649:  Func ByteAddrToStr( Value : TByteAddr) :Str
11650:  Func StrToByteAddr( const Value :Str) : TByteAddr
11651:  Proc IncByteAddr( var ByteAddr : TByteAddr; Increment : Int)
11652:  Proc DecByteAddr( var ByteAddr : TByteAddr; Decrement : Int)
11653:  Func InOutStateToStr( State : TInOutState) :Str
11654:  Func MasterErrorToStr( ErrorCode : TErrorCode) :Str
11655:  Func SlaveErrorToStr( ErrorCode : TErrorCode) :Str
11656: end;
11657:
11658: Proc SIRegister_WDosTimers(CL: TPSPascalCompiler);
11659: begin
11660: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11661:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4,if2 )
11662: DpmiPmVector', 'Int64
11663:  'DInterval','LongInt'( 1000);
11664:  //'DEnabled','Boolean')BoolToStr( True);
11665:  'DIntFreq','string' if64
11666:  //'DMessages','Boolean if64);
11667:  SIRegister_TwdxCustomTimer(CL);
11668:  SIRegister_TwdxTimer(CL);
11669:  SIRegister_TwdxRtcTimer(CL);
11670:  SIRegister_TCustomIntTimer(CL);
11671:  SIRegister_TIntTimer(CL);
11672:  SIRegister_TRtcIntTimer(CL);
11673:  Func RealNow : TDateTime
11674:  Func MsToDateTime( MilliSecond : LongInt) : TDateTime
11675:  Func DateTimeToMs( Time : TDateTime) : LongInt
11676: end;
11677:
11678: Proc SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11679: begin
11680: TIdSyslogPRI', 'Int
11681: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11682:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11683:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11684:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11685:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11686: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11687:   SIRegister_TIdSysLogMsgPart(CL);
11688:   SIRegister_TIdSysLogMessage(CL);
11689:  Func FacilityToString( AFac : TIdSyslogFacility) :Str
11690:  Func SeverityToString( ASec : TIdsyslogSeverity) :Str
11691:  Func NoToSeverity( ASev : Word) : TIdSyslogSeverity
11692:  Func logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11693:  Func NoToFacility( AFac : Word) : TIdSyslogFacility
11694:  Func logFacilityToNo( AFac : TIdSyslogFacility) : Word
11695: end;
11696:
11697: Proc SIRegister_TextUtils(CL: TPSPascalCompiler);
11698: begin
11699:  'UWhitespace','String '(?:\s*)
11700:  Func StripSpaces( const AText :Str) :Str
11701:  Func CharCount( const AText :Str; Ch : Char) : Int
11702:  Func BalancedText(const AText:Str; const Ch1,Ch2: Char; const Count: Int) :Str
11703:  Func BalancedTextReg( const AText:str;const Ch1, Ch2 : Char; const Count :Int):str
11704: end;
11705:
11706: Proc SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11707: begin
11708:  ExtPascalVersion','String '0.9.8
11709:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11710:    +'Opera, brKonqueror, brMobileSafari )
11711:   AddTypeS('TCSSUnit',(cssPX,cssPerc,cssEM,cssEX,cssIN,cssCM,cssMM,cssPT, cssPC, cssnone )
11712:   AddTypeS('TExtProcedure', 'Procedure
11713:  Func DetermineBrowser( const UserAgentStr :Str) : TBrowser
11714:  Func ExtExtract(const Delims:array of string;var S:str;var Matches:TStringList;Remove:bool):bool;
11715:  Func ExtExplode( Delim : char; const S :Str; Separator : char) : TStringList
11716:  Func FirstDelimiter( const Delimiters, S :Str; Offset : Int) : Int
```

```
11717:  Func RPosEx( const Substr, Str :Str; Offset : Int) : Int
11718:  Func CountStr( const Substr, Str :Str; UntilStr :Str) : Int
11719:  Func StrToJS( const S :Str; UseBR :Bool) :Str
11720:  Func CaseOf( const S :Str; const Cases : array of string) : Int
11721:  Func RCaseOf( const S :Str; const Cases : array of string) : Int
11722:  Func EnumToJSString( TypeInfo : PTypeInfo; Value : Int) :Str
11723:  Func SetPaddings(Top:Int;Right:int;Bottom:intr;Left:int;CSSUnit:TCSSUnit;Header:bool):str;
11724:  Func SetMargins(Top:Int;Right:int;Bottom:int;Left:Int;CSSUnit:TCSSUnit;Header:bool):str;
11725:  Func ExtBefore( const BeforeS, AfterS, S :Str) :Bool
11726:  Func IsUpperCase( S :Str) :Bool
11727:  Func BeautifyJS(const AScript:str;const StartingLevel:Int;SplitHTMLNewLine:bool):str;
11728:  Func BeautifyCSS( const AStyle :Str) :Str
11729:  Func LengthRegExp( Rex :Str; CountAll :Bool) : Int
11730:  Func JSDateToDateTime( JSDate :Str) : TDateTime
11731: end;
11732:
11733: Proc SIRegister_JclShell(CL: TPSPascalCompiler);
11734: begin
11735:   TSHDeleteOption','( doSilent, doAllowUndo, doFilesOnly )
11736:   TSHDeleteOptions','set of TSHDeleteOption
11737:   TSHRenameOption','( roSilent, roRenameOnCollision )
11738:   TSHRenameOptions','set of TSHRenameOption
11739:  Func SHDeleteFiles(Parent:HWND;const Files:str;Options:TSHDeleteOptions):Boolean
11740:  Func SHDeleteFolder(Parent:HWND; const Folder:str; Options: TSHDeleteOptions): Bool
11741:  Func SHRenameFile( const Src, Dest :Str; Options : TSHRenameOptions) :Bool
11742:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11743:   TEnumFolderFlags', 'set of TEnumFolderFlag
11744:   TEnumFolderRec','record DisplayName:str; Attributes:DWORD;IconLarge:HICON;IconSmall:HICON;
       Item:PItemIdList;EnumIdList:IEnumIdList;Folder:IShellFolder;end
11745:  Func SHEnumFolderFirst(const Folder:str;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11746:  Func SHEnumSpecialFolderFirst(SpecFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11747:  Proc SHEnumFolderClose( var F : TEnumFolderRec)
11748:  Func SHEnumFolderNext( var F : TEnumFolderRec) :Bool
11749:  Func GetSpecialFolderLocation( const Folder : Int) :Str
11750:  Func DisplayPropDialog( const Handle : HWND; const FileName :Str) :Bool;
11751:  Func DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) :Bool;
11752:  Func DisplayContextMenu(const Handle:HWND; const FileName:Str; Pos:TPoint):Bool
11753:  Func OpenFolder( const Path :Str; Parent : HWND) :Bool
11754:  Func OpenSpecialFolder( FolderID : Int; Parent : HWND) :Bool
11755:  Func SHReallocMem( var P : Pointer; Count : Int) :Bool
11756:  Func SHAllocMem( out P : Pointer; Count : Int) :Bool
11757:  Func SHGetMem( var P : Pointer; Count : Int) :Bool
11758:  Func SHFreeMem( var P : Pointer) :Bool
11759:  Func DriveToPidlBind( const DriveName :Str; out Folder : IShellFolder) : PItemIdList
11760:  Func PathToPidl( const Path :Str; Folder : IShellFolder) : PItemIdList
11761:  Func PathToPidlBind( const FileName :Str; out Folder : IShellFolder) : PItemIdList
11762:  Func PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11763:  Func PidlCompare( const Pidl1, Pidl2 : PItemIdList) :Bool
11764:  Func PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) :Bool
11765:  Func PidlFree( var IdList : PItemIdList) :Bool
11766:  Func PidlGetDepth( const Pidl : PItemIdList) : Int
11767:  Func PidlGetLength( const Pidl : PItemIdList) : Int
11768:  Func PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11769:  Func PidlToPath( IdList : PItemIdList) :Str
11770:  Func StrRetFreeMem( StrRet : TStrRet) :Bool
11771:  Func StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free :Bool) :Str
11772:   PShellLink', '^TShellLink // will not work
11773:   TShellLink', 'record Arguments :Str; ShowCmd : Int; Work'
11774:    +ingDirectory :Str; IdList : PItemIDList; Target :Str; Description '
11775:    +:Str; IconLocation :Str; IconIndex : Int; HotKey : Word; end
11776:  Proc ShellLinkFree( var Link : TShellLink)
11777:  Func ShellLinkResolve( const FileName :Str; var Link : TShellLink) : HRESULT
11778:  Func ShellLinkCreate( const Link : TShellLink; const FileName :Str) : HRESULT
11779:  Func ShellLinkCreateSystem(const Link:TShellLink;const Folder:Int;const FileName:str):HRESULT;
11780:  Func ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) :Bool
11781:  Func SHDllGetVersion( const FileName :Str; var Version : TDllVersionInfo) :Bool
11782:  Func GetSystemIcon( IconIndex : Int; Flags :Card) : HICON
11783:  Func OverlayIcon( var Icon : HICON; Overlay : HICON; Large :Bool) :Bool
11784:  Func OverlayIconShortCut( var Large, Small : HICON) :Bool
11785:  Func OverlayIconShared( var Large, Small : HICON) :Bool
11786:  Func SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) :Str
11787:  Func ShellExecEx(const FileName:str;const Paramtrs:str;const Verb:str;CmdShow:Int):Bool;
11788:  Func ShellExec(Wnd:Int;const Operati,FileNam,Paramtrs,Directy:str;ShowCommand:Int):Bool;
11789:  Func ShellExecAndWait(const FileName:str;const Paramets:str;const Verb:str;CmdShow:Int):Bool;
11790:  Func ShellOpenAs( const FileName :Str) :Bool
11791:  Func ShellRasDial( const EntryName :Str) :Bool
11792:  Func ShellRunControlPanel( const NameOrFileName:str; AppletNumber:Int):Boolean
11793:  Func GetFileNameIcon( const FileName :Str; Flags :Card) : HICON
11794:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11795:  Func GetFileExeType( const FileName : TFileName) : TJclFileExeType
11796:  Func ShellFindExecutable( const FileName, DefaultDir :Str) :Str
11797:  Proc keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11798:  Func OemKeyScan( wOemChar : Word) : DWORD
11799:  Proc mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11800: end;
11801:
11802: Proc SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11803: begin
11804:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
```

```
11805:  //Func xmlValidChar( const Ch : AnsiChar) :Bool;
11806:  Func xmlValidChar1( const Ch : UCS4Char) :Bool;
11807:  Func xmlValidChar2( const Ch : WideChar) :Bool;
11808:  Func xmlIsSpaceChar( const Ch : WideChar) :Bool
11809:  Func xmlIsLetter( const Ch : WideChar) :Bool
11810:  Func xmlIsDigit( const Ch : WideChar) :Bool
11811:  Func xmlIsNameStartChar( const Ch : WideChar) :Bool
11812:  Func xmlIsNameChar( const Ch : WideChar) :Bool
11813:  Func xmlIsPubidChar( const Ch : WideChar) :Bool
11814:  Func xmlValidName( const Text : UnicodeString) :Bool
11815:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11816:  //Func xmlSkipSpace( var P : PWideChar) :Bool
11817:  //Func xmlSkipEq( var P : PWideChar) :Bool
11818:  //Func xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) :Bool
11819:  //Func xmlGetEntityEncoding(const Buf:Pointer;const BufSize:Int;out HeaderSize:Int):TUnicodeCodecClass
11820:  Func xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11821:  Func xmlTag( const Tag : UnicodeString) : UnicodeString
11822:  Func xmlEndTag( const Tag : UnicodeString) : UnicodeString
11823:  Func xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11824:  Func xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11825:  Proc xmlSafeTextInPlace( var Txt : UnicodeString)
11826:  Func xmlSafeText( const Txt : UnicodeString) : UnicodeString
11827:  Func xmlSpaceIndent( const IndentLength : Int; const IndentLevel : Int):UnicodeString
11828:  Func xmlTabIndent( const IndentLevel : Int) : UnicodeString
11829:  Func xmlComment( const Comment : UnicodeString) : UnicodeString
11830:  Proc SelfTestcXMLFunctions
11831: end;
11832:
11833: (*----------------------------------------------------------------------------*)
11834: Proc SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11835: begin
11836:  Func AWaitCursor : IUnknown
11837:  Func ChangeCursor( NewCursor : TCursor) : IUnknown
11838:  Proc SuspendRedraw( AControl : TWinControl; Suspend :Bool)
11839:  Func YesNo( const ACaption, AMsg :Str) :Bool
11840:  Proc strTokenize( const S :Str; Delims : TSysCharSet; Results : TStrings)
11841:  Func GetBorlandLibPath( Version : Int; ForDelphi :Bool) :Str
11842:  Func GetExpandedLibRoot( Version : Int; ForDelphi :Bool) :Str
11843:  Proc GetPathList( Version : Int; ForDelphi :Bool; Strings : TStrings)
11844:  Proc GetSystemPaths( Strings : TStrings)
11845:  Proc MakeEditNumeric( EditHandle : Int)
11846: end;
11847:
11848: Proc SIRegister_yuvconverts(CL: TPSPascalCompiler);
11849: begin
11850:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11851:  'BI_YUY2','LongWord( $32595559);
11852:  'BI_UYVY','LongWord').SetUInt( $59565955);
11853:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11854:  'BI_YVU9','LongWord').SetUInt( $39555659);
11855:  'BI_YUV12','LongWord( $30323449);
11856:  'BI_Y8','LongWord').SetUInt( $20203859);
11857:  'BI_Y211','LongWord').SetUInt( $31313259);
11858:  Func BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11859:  Func ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Int):Bool;
11860: end;
11861:
11862: (*----------------------------------------------------------------------------*)
11863: Proc SIRegister_AviCap(CL: TPSPascalCompiler);
11864: begin
11865:  'WM_USER','LongWord').SetUInt( $0400);
11866:  'WM_CAP_START','LongWord').SetUint($0400);
11867:  'WM_CAP_END','longword').SetUint($0400+85);
11868:  //WM_CAP_START+  85
11869:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11870:  Func capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11871:  Func capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11872:  Func capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11873:  Func capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11874:  Func capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11875:  Func capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11876:  Func capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11877:  Func capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11878:  Func capGetUserData( hwnd : THandle) : LongInt
11879:  Func capDriverConnect( hwnd : THandle; I : Word) : LongInt
11880:  Func capDriverDisconnect( hwnd : THandle) : LongInt
11881:  Func capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11882:  Func capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11883:  Func capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11884:  Func capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11885:  Func capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11886:  Func capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11887:  Func capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11888:  Func capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11889:  Func capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11890:  Func capEditCopy( hwnd : THandle) : LongInt
11891:  Func capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11892:  Func capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11893:  Func capGetAudioFormatSize( hwnd : THandle) : LongInt
```

```
11894:  Func capDlgVideoFormat( hwnd : THandle) : LongInt
11895:  Func capDlgVideoSource( hwnd : THandle) : LongInt
11896:  Func capDlgVideoDisplay( hwnd : THandle) : LongInt
11897:  Func capDlgVideoCompression( hwnd : THandle) : LongInt
11898:  Func capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11899:  Func capGetVideoFormatSize( hwnd : THandle) : LongInt
11900:  Func capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11901:  Func capPreview( hwnd : THandle; f : Word) : LongInt
11902:  Func capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11903:  Func capOverlay( hwnd : THandle; f : Word) : LongInt
11904:  Func capPreviewScale( hwnd : THandle; f : Word) : LongInt
11905:  Func capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11906:  Func capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11907:  Func capGrabFrame( hwnd : THandle) : LongInt
11908:  Func capGrabFrameNoStop( hwnd : THandle) : LongInt
11909:  Func capCaptureSequence( hwnd : THandle) : LongInt
11910:  Func capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11911:  Func capCaptureStop( hwnd : THandle) : LongInt
11912:  Func capCaptureAbort( hwnd : THandle) : LongInt
11913:  Func capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11914:  Func capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11915:  Func capCaptureSingleFrame( hwnd : THandle) : LongInt
11916:  Func capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11917:  Func capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11918:  Func capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11919:  Func capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11920:  Func capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11921:  Func capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11922:  Func capPalettePaste( hwnd : THandle) : LongInt
11923:  Func capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11924:  Func capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11925:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11926:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11927:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11928:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11929:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11930:   //PCapStatus', '^TCapStatus // will not work
11931:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11932:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11933:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11934:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11935:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11936:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11937:    +' wNumAudioAllocated : WORD; end
11938:   //PCaptureParms', '^TCaptureParms // will not work
11939:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11940:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11941:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11942:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11943:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11944:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11945:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11946:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11947:    +he : BOOL; AVStreamMaster : WORD; end
11948:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11949:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11950:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11951:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11952:  Func capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Int; nWidth, nHeight : Int;
        hwndParent : THandle; nID : Int) : THandle
11953:  Func capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Int;lpszVer:PChar;cbVer:Int):Bool;
11954:  'IDS_CAP_BEGIN','LongInt'( 300);
11955:  'IDS_CAP_END','LongInt'( 301);
11956:  'IDS_CAP_INFO','LongInt'( 401);
11957:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11958:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11959:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11960:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11961:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11962:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11963:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11964:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11965:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11966:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11967:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11968:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11969:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11970:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11971:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11972:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11973:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11974:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11975:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11976:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11977:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11978:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11979:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11980:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11981:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
```

```
11982:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11983:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11984:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11985:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11986:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11987:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11988:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11989:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11990:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11991:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11992:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11993:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11994:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11995:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11996:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11997:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11998:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11999:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
12000:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
12001:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
12002:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
12003:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
12004:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
12005:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
12006:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
12007:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
12008:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
12009:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
12010:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
12011:  'AVICAP32','String 'AVICAP32.dll
12012: end;
12013:
12014: Proc SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
12015: begin
12016:  Func AlBoolToInt( Value :Bool) : Int
12017:  Func ALMediumPos( LTotal, LBorder, LObject : Int) : Int
12018:  Func AlIsValidEmail( const Value : Ansistr) :Bool
12019:  Func AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
12020:  Func ALInc( var x : Int; Count : Int) : Int
12021:  Func ALCopyStr(const aSourceString: Ansistr; aStart, aLength: Int): Ansistr
12022:  Func ALGetStringFromFile(filename:AnsiStr;const ShareMode:Word= fmShareDenyWrite):AnsiStr;
12023:  Proc ALSaveStringtoFile(Str: Ansistr; filename: Ansistr);
12024:  Func ALIsInt(const S: Ansistr):Bool;
12025:  Func ALIsDecimal(const S: Ansistr):Bool;
12026:  Func ALStringToWideString(const S: Ansistr; const aCodePage: Word): WideString;
12027:  Func AlWideStringToString(const WS: WideString; const aCodePage: Word): Ansistr;
12028:  Func ALQuotedStr(const S: Ansistr; const Quote: AnsiChar = ''''): Ansistr;
12029:  Func ALDequotedStr(const S: Ansistr; AQuote: AnsiChar): Ansistr;
12030:  Func AlUTF8removeBOM(const S: Ansistr): Ansistr;
12031:  Func ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): Ansistr;
12032:  Func ALRandomStr(const aLength: Longint): Ansistr;
12033:  Func ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char):Str;
12034:  Func ALRandomStrU(const aLength: Longint):Str;
12035: end;
12036:
12037: Proc SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
12038: begin
12039:  Proc ALJSONToTStrings(const AJsonStr:Ansistr;aLst:TALStrings; const aNullStr:AnsiStr;const
       aTrueStr:Ansistr; const aFalseStr: Ansistr)
12040: end;
12041:
12042: Proc SIRegister_ALWindows(CL: TPSPascalCompiler);
12043: begin
12044:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
12045:    +RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
12046:    +ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
12047:    +; ullAvailExtendedVirtual : Int64; end
12048:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
12049:  Func ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
12050:  Func ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
12051:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
12052:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
12053:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
12054:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
12055:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
12056: end;
12057:
12058: Proc SIRegister_IPCThrd(CL: TPSPascalCompiler);
12059: begin
12060:   SIRegister_THandledObject(CL);
12061:   SIRegister_TEvent(CL);
12062:   SIRegister_TMutex(CL);
12063:   SIRegister_TSharedMem(CL);
12064:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
12065:  'TRACE_BUFFER','String 'TRACE_BUFFER
12066:  'TRACE_MUTEX','String 'TRACE_MUTEX
12067:   //PTraceEntry', '^TTraceEntry // will not work
12068:   SIRegister_TIPCTracer(CL);
12069:  'MAX_CLIENTS','LongInt'( 6);
```

```
12070:  'IPCTIMEOUT','LongInt'( 2000);
12071:  'IPCBUFFER_NAME','String 'BUFFER_NAME
12072:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
12073:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
12074:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
12075:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
12076:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
12077:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
12078:   FindClass('TOBJECT'),'EMonitorActive
12079:   FindClass('TOBJECT'),'TIPCThread
12080:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
12081:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
12082:    +'ach, evClientSwitch, evClientSignal, evClientExit )
12083:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
12084:   TClientFlags', 'set of TClientFlag
12085:   //PEventData', '^TEventData // will not work
12086:   TEventData', 'record X:SmallInt; Y:SmallInt;Flag: TClientFlag; Flags:TClientFlags; end
12087:   TConnectEvent', 'Proc ( Sender : TIPCThread; Connecting :Bool)
12088:   TDirUpdateEvent', 'Proc ( Sender : TIPCThread)
12089:   TIPCNotifyEvent', 'Proc ( Sender : TIPCThread; Data : TEventData)
12090:   //PIPCEventInfo', '^TIPCEventInfo // will not work
12091:   TIPCEventInfo','record FID:Int;FKind:TEventKind;FData:TEventData;end
12092:   SIRegister_TIPCEvent(CL);
12093:   //PClientDirRecords', '^TClientDirRecords // will not work
12094:   SIRegister_TClientDirectory(CL);
12095:   TIPCState', '( stInActive, stDisconnected, stConnected )
12096:   SIRegister_TIPCThread(CL);
12097:   SIRegister_TIPCMonitor(CL);
12098:   SIRegister_TIPCClient(CL);
12099:  Func IsMonitorRunning( var Hndl : THandle) :Bool
12100: end;
12101:
12102: (*----------------------------------------------------------------------------*)
12103: Proc SIRegister_ALGSMComm(CL: TPSPascalCompiler);
12104: begin
12105:   SIRegister_TAlGSMComm(CL);
12106:  Func AlGSMComm_BuildPDUMessage( aSMSCenter,aSMSAddress,aMessage: Ansistr) : Ansistr
12107:  Proc AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiStr;var aSMSCenter,aSMSAddress,AMessage:AnsiStr);
12108:  Func AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : Ansistr
12109:  Func AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:Ansistr;const UseGreekAlphabet:Bool):Widestring;
12110:  Func ALMatchesMask(const Filename, Mask: Ansistr):Bool;
12111: end;
12112:
12113: Proc SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
12114: begin
12115:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Int;
12116:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
12117:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
12118:   TInternetScheme', 'Int
12119:   TALIPv6Binary', 'array[1..16] of Char;
12120:  // TALIPv6Binary = array[1..16] of ansiChar;
12121:  //   TInternetScheme = Int;
12122:   SIRegister_TALHTTPRequestHeader(CL);
12123:   SIRegister_TALHTTPCookie(CL);
12124:   SIRegister_TALHTTPCookieCollection(CL);
12125:   SIRegister_TALHTTPResponseHeader(CL);
12126:  Func ALHTTPDecode( const AStr : Ansistr) : Ansistr
12127:  Proc ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
12128: //Proc ALExtractHTTPFields(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings:TALStrings;StripQuotes:Bool;
12129: //Proc ALExtractHeaderFields(Separators,WhiteSpace,Quotes:TSysCharSet;Content:PAnsiChar;
       Strings:TALStrings;Decode:Bool;StripQuotes:Bool)
12130: //Proc ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings:TALStrings;Decode:Bool;StripQuotes:Bool)
12131:  Func AlRemoveShemeFromUrl( aUrl : Ansistr) : Ansistr
12132:  Func AlExtractShemeFromUrl( aUrl : Ansistr) : TInternetScheme
12133:  Func AlExtractHostNameFromUrl( aUrl : Ansistr) : Ansistr
12134:  Func AlExtractDomainNameFromUrl( aUrl : Ansistr) : Ansistr
12135:  Func AlExtractUrlPathFromUrl( aUrl : Ansistr) : Ansistr
12136:  Func AlInternetCrackUrl(aUrl:Ansistr;var SchemeName,HostName,UserName,Password,UrlPath,ExtraInfo:
       Ansistr; var PortNumber:Int):Bool;
12137:  Func AlInternetCrackUrl1(aUrl:Ansistr;var SchemeNam,HostName,UserName,Password,UrlPath,
       Anchor:Ansistr;Query:TALStrings;var PortNumber:Int):Bool
12138:  Func AlInternetCrackUrl2(var Url:Ansistr;var Anchor:Ansistr;Query:TALStrings):Bool;
12139:  Func AlRemoveAnchorFromUrl( aUrl : Ansistr; var aAnchor : Ansistr) : Ansistr;
12140:  Func AlRemoveAnchorFromUrl1( aUrl : Ansistr) : Ansistr;
12141:  Func AlCombineUrl( RelativeUrl, BaseUrl : Ansistr) : Ansistr;
12142:  Func AlCombineUrl1(RelativeUrl,BaseUrl,Anchor:Ansistr; Query:TALStrings): Ansistr;
12143:  Func ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : Ansistr
12144:  Func ALDateTimeToRfc822Str( const aValue : TDateTime) : Ansistr
12145:  Func ALTryRfc822StrToGMTDateTime( const S : Ansistr; out Value : TDateTime) :Bool
12146:  Func ALRfc822StrToGMTDateTime( const s : Ansistr) : TDateTime
12147:  Func ALTryIPV4StrToNumeric( aIPv4Str : Ansistr; var aIPv4Num :Card) :Bool
12148:  Func ALIPV4StrToNumeric( aIPv4 : Ansistr) :Card
12149:  Func ALNumericToIPv4Str( aIPv4 :Card) : Ansistr
12150:  Func ALZeroIpV6 : TALIPv6Binary
12151:  Func ALTryIPV6StrToBinary( aIPv6Str : Ansistr; var aIPv6Bin : TALIPv6Binary) :Bool
12152:  Func ALIPV6StrTobinary( aIPv6 : Ansistr) : TALIPv6Binary
12153:  Func ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : Ansistr
```

```
12154:  Func ALBinaryStrToIPv6Binary( aIPV6BinaryStr : Ansistr) : TALIPv6Binary
12155: end;
12156:
12157: Proc SIRegister_ALFcnHTML(CL: TPSPascalCompiler);  //JavaScript
12158: begin
12159:  Proc ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const DecodeHTMLText:Bool;
12160:  Func ALUTF8ExtractHTMLText1(HtmlContent:Ansistr;const DecodeHTMLText:Bool):Ansistr;
12161:  Func ALXMLCDataElementEncode( Src : Ansistr) : Ansistr
12162:  Func ALXMLTextElementEncode(Src:Ansistr;const useNumericReference:bool):Ansistr
12163:  Func ALUTF8XMLTextElementDecode( const Src : Ansistr) : Ansistr
12164:  Func ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const useNumRef:bool):Ansistr);
12165:  Func ALUTF8HTMLDecode( const Src : Ansistr) : Ansistr
12166:  Func ALJavascriptEncode(const Src:Ansistr;const useNumericReference:bool):Ansistr
12167:  Func ALUTF8JavascriptDecode( const Src : Ansistr) : Ansistr
12168:  Proc ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:Ansistr;const
       DeleteBodyOfUnwantedTag:Bool;const ReplaceUnwantedTagCharBy:AnsiChar)
12169:  Proc ALCompactHtmlTagParams( TagParams : TALStrings)
12170: Proc ALUTF8JavascriptDecodeV( var Str : Ansistr)
12171: //'Func ALUTF8JavascriptDecode( const Src : Ansistr) : Ansistr
12172:
12173: {This Func evaluates the Javascript code given in the
12174:  parameter "aCode" and returns result. The Func works
12175:  similar to browser's console, so you can send even the code
12176:  like this "2+2" => returns "4".}
12177:
12178: Example: writeln('runJScript: '+runJS('Math.pow(2, 64)'))
12179:
12180:  Func ALRunJavascript( const aCode : Ansistr) : Ansistr
12181:  Func RunJavascript( const aCode : Ansistr) : Ansistr
12182:  Func RunJavascript2(const aCode:Ansistr):Ansistr //no CoInitialize of COM-Server
12183:  Func RunJS( const aCode : Ansistr) : Ansistr  //no CoInitialize of COM-Server
12184:  //CL.AddDelphiFunction('Proc ALHideHtmlUnwantedTagForHTMLHandleTagfunct( var HtmlContent : Ansistr; const
       DeleteBodyOfUnwantedTag:Boolean;const ReplaceUnwantedTagCharBy:AnsiChar)
12185:  //CL.AddDelphiFunction('Proc ALCompactHtmlTagParams( TagParams : TALStrings)
12186:  Func ALJavascriptEncodeU(const Src:str; const useNumericReference:boolean) :Str
12187:  Proc ALJavascriptDecodeVU( var Str :Str)
12188:  Func ALJavascriptDecodeU( const Src :Str):Str
12189: end;
12190:
12191: Proc SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12192: begin
12193:   SIRegister_TALEMailHeader(CL);
12194:   SIRegister_TALNewsArticleHeader(CL);
12195:  Func AlParseEmailAddress(FriendlyEmail:AnsiStr;var RealName:AString;const decodeRealName:Bool):Ansistr;
12196:  Func AlExtractEmailAddress( FriendlyEmail : Ansistr) : Ansistr
12197:  Func ALMakeFriendlyEmailAddress( aRealName, aEmail : Ansistr) : Ansistr
12198:  Func ALEncodeRealName4FriendlyEmailAddress( aRealName : Ansistr) : Ansistr
12199:  Func AlGenerateInternetMessageID : Ansistr;
12200:  Func AlGenerateInternetMessageID1( ahostname : Ansistr) : Ansistr;
12201:  Func ALDecodeQuotedPrintableString( src : Ansistr) : Ansistr
12202:  Func AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:Ansistr;aDefaultCodePage:Int):Ansistr;
12203: end;
12204:
12205: (*----------------------------------------------------------------------------*)
12206: Proc SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12207: begin
12208:  Func ALHostToIP( HostName : Ansistr; var Ip : Ansistr):Boolean
12209:  Func ALIPAddrToName( IPAddr : Ansistr) : Ansistr
12210:  Func ALgetLocalIPs : TALStrings
12211:  Func ALgetLocalHostName : Ansistr
12212: end;
12213:
12214: Proc SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12215: begin
12216:  Proc AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest : TALWebRequest;ServerVariables:TALStrings);
12217:  Proc AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : Ansistr; Url : Ansistr);
12218:  Proc ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12219:  Proc AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,ScriptFileName:Ansistr;Url:AnsiStr;
12220:  Proc
       AlCGIInitServerVariablesFromWebRequest(WebRequest:TALWebRequest;ServerVariables:TALStrings;ScriptName,ScriptFileName:A
       Ansistr);
12221:  Proc AlCGIExec( InterpreterFilename: Ansistr; ServerVariables: TALStrings; RequestContentStream: Tstream;
       ResponseContentStream: Tstream; ResponseHeader: TALHTTPResponseHeader);
12222:  Proc AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:Ansistr; WebRequest :
       TALIsapiRequest; overloadedCookies:Ansistr;overloadedQueryString:Ansistr;overloadedReferer: Ansistr;'
12223:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:Ansistr;ResponseHeader:TALHTTPResponseHeader;
12224:  Proc AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,InterpreterFilename:Ansistr;WebRequest:
       TALIsapiRequest; var ResponseContentString : Ansistr; ResponseHeader : TALHTTPResponseHeader);
12225: end;
12226:
12227: Proc SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12228: begin
12229:   TStartupInfoA', 'TStartupInfo
12230:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12231:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12232:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12233:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
```

```
12234:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12235:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12236:  SE_TCB_NAME','String 'SeTcbPrivilege
12237:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12238:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12239:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12240:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12241:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12242:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12243:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12244:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12245:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12246:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12247:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12248:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12249:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12250:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12251:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12252:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12253:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12254:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12255:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12256:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12257:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12258:  Func AlGetEnvironmentString : Ansistr
12259:  Func ALWinExec32(const FileName,CurrentDir,Environment:Ansistr;InStream:Tstream;OutStream:TStream):Dword;
12260:  Func ALWinExec321(const FileName:AnsiStr;InputStream:Tstream;OutputStream:TStream):Dword;
12261:  Func ALWinExecAndWait32( FileName : Ansistr; Visibility : Int) : DWORD
12262:  Func ALWinExecAndWait32V2( FileName : Ansistr; Visibility : Int) : DWORD
12263:  Func ALNTSetPrivilege( sPrivilege : Ansistr; bEnabled :Bool) :Bool
12264: end;
12265:
12266: Proc SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12267: begin
12268:  Func AlEmptyDirectory(Directory:Ansistr;SubDirectory:Bool;IgnoreFiles:array of Ansistr; const
       RemoveEmptySubDirectory :Bool; const FileNameMask : Ansistr; const MinFileAge : TdateTime):Boolean;
12269:  Func AlEmptyDirectory1( Directory : Ansistr; SubDirectory :Bool; const RemoveEmptySubDirectory:Bool;const
       FileNameMask:ansiStr;const MinFileAge:TdateTime):Bool;
12270:  Func AlCopyDirectory(SrcDirectory,DestDirectory:Ansistr;SubDirectory:Boolean; const FileNameMask:
       Ansistr; const FailIfExists:Boolean):Bool
12271:  Func ALGetModuleName : Ansistr
12272:  Func ALGetModuleFileNameWithoutExtension : Ansistr
12273:  Func ALGetModulePath : Ansistr
12274:  Func AlGetFileSize( const AFileName : Ansistr) : int64
12275:  Func AlGetFileVersion( const AFileName : Ansistr) : Ansistr
12276:  Func ALGetFileCreationDateTime( const aFileName : Ansistr) : TDateTime
12277:  Func ALGetFileLastWriteDateTime( const aFileName : Ansistr) : TDateTime
12278:  Func ALGetFileLastAccessDateTime( const aFileName : Ansistr) : TDateTime
12279:  Proc ALSetFileCreationDateTime(const aFileName:Ansistr;const aCreationDateTime:TDateTime)
12280:  Func ALIsDirectoryEmpty( const directory : Ansistr) :Bool
12281:  Func ALFileExists( const Path : Ansistr) :Bool
12282:  Func ALDirectoryExists( const Directory : Ansistr) :Bool
12283:  Func ALCreateDir( const Dir : Ansistr) :Bool
12284:  Func ALRemoveDir( const Dir : Ansistr) :Bool
12285:  Func ALDeleteFile( const FileName : Ansistr) :Bool
12286:  Func ALRenameFile( const OldName, NewName : Ansistr) :Bool
12287: end;
12288:
12289: Proc SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12290: begin
12291:   NativeInt', 'Int NativeUInt', 'Cardinal
12292:  Func ALMimeBase64EncodeString( const S : Ansistr) : Ansistr
12293:  Func ALMimeBase64EncodeStringNoCRLF( const S : Ansistr) : Ansistr
12294:  Func ALMimeBase64DecodeString( const S : Ansistr) : Ansistr
12295:  Func ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12296:  Func ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12297:  Func ALMimeBase64DecodedSize( const InputSize : NativeInt): NativeInt
12298:  Proc ALMimeBase64Encode(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12299:  Proc ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray;InputOffset:NativeInt; const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12300:  Proc ALMimeBase64EncodeFullLines( const InputBuffer: TByteDynArray; InputOffset : NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12301:  Func ALMimeBase64Decode(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt): NativeInt;
12302:  Func ALMimeBase64DecodePartial(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
       InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt;'
12303:  + var ByteBuffer:Card;var ByteBufferSpace:Card):NativeInt;
12304:  Func ALMimeBase64DecodePartialEnd(out OutputBuffer:TByteDynArray;OutputOffset:NativeInt;const
       ByteBuffer:Card;const ByteBufferSpace:Card):NativeInt;
12305:  Proc ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12306:  Proc ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12307:  Proc ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12308:  Func ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12309:  Func ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray;var ByteBuffer:Card;var ByteBufferSpace:Card) : NativeInt;
```

```
12310:  Func ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Card;const
        ByteBufferSpace:Card):NativeInt;
12311:  Proc ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12312:  Proc ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12313:  Proc ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12314:  Proc ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12315:  Proc ALMimeBase64EncodeStreamNoCRLF(const InputStream:TStream;const OutputStream:TStream)
12316:  Proc ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12317:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12318:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'(cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12319:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12320:  Proc ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12321:  Proc ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12322:  Func ALGetDefaultFileExtFromMimeContentType( aContentType : Ansistr) : Ansistr
12323:  Func ALGetDefaultMIMEContentTypeFromExt( aExt : Ansistr) : Ansistr
12324: end;
12325:
12326: Proc SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12327: begin
12328:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12329:   FindClass('TOBJECT'),TALXMLNode
12330:   FindClass('TOBJECT'),TALXMLNodeList
12331:   FindClass('TOBJECT'),TALXMLDocument
12332:   TAlXMLParseProcessingInstructionEvent','Proc (Sender:TObject;const Target,Data:AnsiStr)
12333:   TAlXMLParseTextEvent','Proc (Sender: TObject; const str: Ansistr)
12334:   TAlXMLParseStartElementEvent','Procedure(Sender:TObject;const Name:Ansistr;const Attributes:TALStrings)
12335:   TAlXMLParseEndElementEvent','Procedure(Sender : TObject;const Name: Ansistr)
12336:   TALXmlNodeType','(ntReserved,ntElement,ntAttribute,ntText,ntCData,ntEntityRef,ntEntity,ntProcessingInstr,
        ntComment,ntDocumentntDocType,ntDocFragment,ntNotation)
12337:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12338:   TALXMLDocOptions', 'set of TALXMLDocOption
12339:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12340:   TALXMLParseOptions', 'set of TALXMLParseOption
12341:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12342:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12343:   SIRegister_EALXMLDocError(CL);
12344:   SIRegister_TALXMLNodeList(CL);
12345:   SIRegister_TALXMLNode(CL);
12346:   SIRegister_TALXmlElementNode(CL);
12347:   SIRegister_TALXmlAttributeNode(CL);
12348:   SIRegister_TALXmlTextNode(CL);
12349:   SIRegister_TALXmlDocumentNode(CL);
12350:   SIRegister_TALXmlCommentNode(CL);
12351:   SIRegister_TALXmlProcessingInstrNode(CL);
12352:   SIRegister_TALXmlCDataNode(CL);
12353:   SIRegister_TALXmlEntityRefNode(CL);
12354:   SIRegister_TALXmlEntityNode(CL);
12355:   SIRegister_TALXmlDocTypeNode(CL);
12356:   SIRegister_TALXmlDocFragmentNode(CL);
12357:   SIRegister_TALXmlNotationNode(CL);
12358:   SIRegister_TALXMLDocument(CL);
12359:  cAlXMLUTF8EncodingStr','String 'UTF-8
12360:  cAlXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12361:  CALNSDelim','String ':
12362:  CALXML','String 'xml
12363:  CALVersion','String 'version
12364:  CALEncoding','String 'encoding
12365:  CALStandalone','String 'standalone
12366:  CALDefaultNodeIndent','String '
12367:  CALXmlDocument','String 'DOCUMENT
12368:  Func ALCreateEmptyXMLDocument(const Rootname: Ansistr): TalXMLDocument
12369:  Proc ALClearXMLDocument(const rootname:Ansistr;xmldoc:TalXMLDocument;const EncodingStr:AnsiStr);
12370:  Func ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,ChildNodeValue:Ansistr;const
        Recurse:Bool):TalxmlNode
12371:  Func ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : Ansistr; const
        ChildNodeName, ChildNodeValue : Ansistr; const Recurse :Bool) : TalxmlNode
12372:  Func ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:Ansistr;const
        Recurse:Bool):TalxmlNode
12373:  Func ALFindXmlNodeByNameAndAttribute(xmlrec:TalxmlNode;const NodeName:Ansistr;const AttributeName,
        AttributeValue:Ansistr;const Recurse:Bool):TalxmlNode
12374:  Func ALExtractAttrValue(const AttrName,AttrLine:Ansistr;const Default:Ansistr): Ansistr
12375: end;
12376:
12377: Proc SIRegister_TProcess(CL: TPSPascalCompiler);
12378: begin
12379:   //with RegClassS(CL,'TComponent', 'TProcess') do
12380:   with CL.AddClassN(CL.FindClass('TComponent'),'TProcess') do begin
12381:     Constructor Create( AOwner : TComponent)
12382:     Proc Free
12383:     Proc Execute
12384:     Func Resume : Integer
12385:     Func Suspend : Integer
12386:     Func Terminate( AExitCode : Integer) :Bool
12387:     Func WaitOnExit : DWord
12388:     WindowRect', 'Trect', iptrw);
12389:     StartupInfo', 'TStartupInfo', iptr);
12390:     ProcessAttributes', 'TSecurityAttributes', iptrw);
12391:     ProcessInformation', 'TProcessInformation', iptr);
12392:     Handle', 'THandle', iptr);
```

```
12393:     ThreadHandle', 'THandle', iptr);
12394:     Input', 'TOutPutPipeStream', iptr);
12395:     OutPut', 'TInputPipeStream', iptr);
12396:     StdErr', 'TinputPipeStream', iptr);
12397:     ExitStatus', 'Integer', iptr);
12398:     Active', 'Boolean', iptrw);
12399:     ApplicationName', 'String', iptrw);
12400:     CommandLine', 'String', iptrw);
12401:     ConsoleTitle', 'String', iptrw);
12402:     CurrentDirectory', 'String', iptrw);
12403:     DeskTop', 'String', iptrw);
12404:     Environment', 'TStrings', iptrw);
12405:     FillAttribute', 'Cardinal', iptrw);
12406:     InheritHandles', 'LongBool', iptrw);
12407:     Options', 'TProcessOptions', iptrw);
12408:     Priority', 'TProcessPriority', iptrw);
12409:     StartUpOptions', 'TStartUpOptions', iptrw);
12410:     Running', 'Boolean', iptr);
12411:     ShowWindow', 'TShowWindowOptions', iptrw);
12412:     ThreadAttributes', 'TSecurityAttributes', iptrw);
12413:     WindowColumns', 'Cardinal', iptrw);
12414:     WindowHeight', 'Cardinal', iptrw);
12415:     WindowLeft', 'Cardinal', iptrw);
12416:     WindowRows', 'Cardinal', iptrw);
12417:     WindowTop', 'Cardinal', iptrw);
12418:     WindowWidth', 'Cardinal', iptrw);
12419:   end;
12420: end;
12421:
12422: (*----------------------------------------------------------------------------*)
12423: Proc SIRegister_TAlWebSpider(CL: TPSPascalCompiler);
12424: begin
12425:   //with RegClassS(CL,'TObject', 'TAlWebSpider') do
12426:   with CL.AddClassN(CL.FindClass('TObject'),'TAlWebSpider') do begin
12427:     RegisterMethod('Proc Crawl
12428:     RegisterMethod('Proc UpdateLinkToLocalPath
12429:     OnCrawlBeforeDownload', 'TAlWebSpiderCrawlBeforeDownloadEvent', iptrw);
12430:     OnCrawlAfterDownload', 'TAlWebSpiderCrawlAfterDownloadEvent', iptrw);
12431:     OnCrawlDownloadSuccess', 'TAlWebSpiderCrawlDownloadSuccessEvent', iptrw);
12432:     OnCrawlDownloadRedirect', 'TAlWebSpiderCrawlDownloadRedirectEvent', iptrw);
12433:     OnCrawlDownloadError', 'TAlWebSpiderCrawlDownloadErrorEvent', iptrw);
12434:     OnCrawlGetNextLink', 'TAlWebSpiderCrawlGetNextLinkEvent', iptrw);
12435:     OnCrawlFindLink', 'TAlWebSpiderCrawlFindLinkEvent', iptrw);
12436:     OnCrawlEnd', 'TAlWebSpiderCrawlEndEvent', iptrw);
12437:     OnUpdateLinkToLocalPathGetNextFile','TAlWebSpiderUpdateLinkToLocalPathGetNextFileEvent'rw);
12438:     OnUpdateLinkToLocalPathFindLink', 'TAlWebSpiderUpdateLinkToLocalPathFindLinkEvent', iptrw);
12439:     OnUpdateLinkToLocalPathEnd', 'TAlWebSpiderUpdateLinkToLocalPathEndEvent', iptrw);
12440:     RegisterProperty('HttpClient', 'TalHttpClient', iptrw);
12441:   end;
12442: end;
12443:
12444: https://sourceforge.net/p/alcinoe/code/HEAD/tree/source/ALWebSpider.pas#l184
12445:    {----------------------------------}
12446: TAlTrivialWebSpider = Class(Tobject)
12447: Private
12448:   FWebSpider: TalWebSpider;
12449:   fStartUrl: Ansistr;
12450:   fLstUrlCrawled: TALStrings;
12451:   fLstErrorEncountered: TALStrings;
12452:   FPageDownloadedBinTree: TAlStringKeyAVLBinaryTree;
12453:   FPageNotYetDownloadedBinTree: TAlStringKeyAVLBinaryTree;
12454:   FCurrentDeepLevel: Integer;
12455:   FCurrentLocalFileNameIndex: Integer;
12456:   fMaxDeepLevel: Integer;
12457:   fOnCrawlBeforeDownload: TAlWebSpiderCrawlBeforeDownloadEvent;
12458:   fUpdateLinkToLocalPath:Bool;
12459:   fExcludeMask: Ansistr;
12460:   fStayInStartDomain:Bool;
12461:   fSaveDirectory: Ansistr;
12462:   fSplitDirectoryAmount: integer;
12463:   FHttpClient: TalHttpClient;
12464:   fIncludeMask: Ansistr;
12465:   fOnCrawlAfterDownload: TAlWebSpiderCrawlAfterDownloadEvent;
12466:   fOnCrawlFindLink: TAlTrivialWebSpiderCrawlFindLinkEvent;
12467:   fDownloadImage:Bool;
12468:   fOnUpdateLinkToLocalPathProgress: TAlTrivialWebSpiderUpdateLinkToLocalPathProgressEvent;
12469:   fOnCrawlProgress: TAlTrivialWebSpiderCrawlProgressEvent;
12470:   Proc WebSpiderCrawlDownloadError(Sender:TObject;const URL,ErrorMessage:Ansistr; HTTPResponseHeader:
      TALHTTPResponseHeader; var StopCrawling: Bool);
12471:   Proc WebSpiderCrawlDownloadRedirect(Sender:TObject;const Url,RedirectedTo:Ansistr;
12472:                        HTTPResponseHeader: TALHTTPResponseHeader; var StopCrawling:Bool);
12473:   Proc WebSpiderCrawlDownloadSuccess(Sender: TObject; const Url: Ansistr; HTTPResponseHeader:
      TALHTTPResponseHeader;HttpResponseContentTStream;var StopCrawling:Bool);
12474:  Proc WebSpiderCrawlFindLink(Sender:TObject;const HtmlTagString:AnsiStr;HtmlTagParams:
12475:                        TALStrings; const URL: Ansistr);
12476:  Proc WebSpiderCrawlGetNextLink(Sender: TObject; var Url: Ansistr);
12477:  Proc WebSpiderUpdateLinkToLocalPathFindLink(Sender:TObject;const HtmlTagString:AnsiStr;
12478:             HtmlTagParams: TALStrings;const URL: Ansistr;var LocalPath: Ansistr);
12479:  Proc WebSpiderUpdateLinkToLocalPathGetNextFile(Sender:TObject;var FileN,BaseHref:AnsiStr);
```

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

```
12556: begin
12557: CL.AddClassN(CL.FindClass('TOBJECT'),'EALIsapiRequestContentSizeTooBig
12558: CL.AddClassN(CL.FindClass('TOBJECT'),'EALIsapiRequestConnectionDropped
12559: SIRegister_TALWebRequest(CL);
12560: SIRegister_TALISAPIRequest(CL);
12561: SIRegister_TALWebResponse(CL);
12562: SIRegister_TALISAPIResponse(CL);
12563:  Func ALIsapiHttpStatusString( StatusCode : Integer) : Ansistr
12564:  CL.AddConstantN('HSE_IO_SYNC','LongWord').SetUInt( $00000001);
12565:  'HSE_IO_ASYNC','LongWord').SetUInt( $00000002);
12566:  'HSE_IO_DISCONNECT_AFTER_SEND','LongWord').SetUInt( $00000004);
12567:  'HSE_IO_SEND_HEADERS','LongWord').SetUInt( $00000008);
12568: end;
12569:
12570: Proc SIRegister_uUsb(CL: TPSPascalCompiler);
12571: begin
12572:   //CL.AddTypeS('PDevBroadcastHdr', '^DEV_BROADCAST_HDR // will not work
12573:   CL.AddTypeS(DEV_BROADCAST_HDR','record dbch_size:DWORD;dbch_devicetype:DWORD;dbch_reserved:DWORD;end;
12574:   CL.AddTypeS('TDevBroadcastHdr', 'DEV_BROADCAST_HDR
12575:   //CL.AddTypeS('PDevBroadcastDeviceInterface','^DEV_BROADCAST_DEVICEINTERFACE//will not work
12576:   CL.AddTypeS(DEV_BROADCAST_DEVICEINTERFACE', 'record dbcc_size : DWORD; dbcc_'
12577:    +devicetype: DWORD;dbcc_reserved:DWORD; dbcc_classguid: TGUID; dbcc_name:Char; end
12578:   CL.AddTypeS('TDevBroadcastDeviceInterface', 'DEV_BROADCAST_DEVICEINTERFACE
12579:  //CL.AddConstantN('GUID_DEVINTERFACE_USB_DEVICE','TGUID').SetString( '{A5DCBF10-6530-11D2-901F-
    00C04FB951ED}
12580:  'DBT_DEVICEARRIVAL','LongWord').SetUInt( $8000);
12581:  'DBT_DEVICEREMOVECOMPLETE','LongWord').SetUInt( $8004);
12582:  'DBT_DEVTYP_DEVICEINTERFACE','LongWord').SetUInt( $00000005);
12583:   CL.AddTypeS('TUsbNotifyProc', 'Proc ( Sender : TObject; const DeviceName:Str)
12584:   SIRegister_TUsbNotifier(CL);
12585: end;
12586:
12587: Proc SIRegister_uWebcam(CL: TPSPascalCompiler);
12588: begin
12589:  CL.AddConstantN('WM_CAP_START','LongWord').SetUInt( $0400);
12590:  'WM_CAP_DRIVER_CONNECT','LongInt').SetInt( $0400 + 10);
12591:  'WM_CAP_DRIVER_DISCONNECT','LongInt').SetInt( $0400 + 11);
12592:  'WM_CAP_SAVEDIB','LongInt').SetInt( $0400 + 25);
12593:  'WM_CAP_GRAB_FRAME','LongInt').SetInt( $0400 + 60);
12594:  'WM_CAP_STOP','LongInt').SetInt( $0400 + 68);
12595:  Func Connectwebcam( WebcamID : integer) :Bool
12596:  Proc CaptureWebCam( FilePath :Str)
12597:  Proc CloseWebcam( );
12598:  Proc WebcamInit;
12599:  Func WebcamList: TStringlist;
12600: end;
12601:
12602: Proc WebcamInit;
12603: begin
12604:   LibHandle := LoadLibrary('avicap32.dll
12605:   CapGetDriverDescriptionA:= GetProcAddress(LibHandle,'capGetDriverDescriptionA
12606:   CapCreateCaptureWindowA := GetProcAddress(LibHandle,'capCreateCaptureWindowA
12607: end;
12608: Example: webcaminit;
12609:          writeln(webcamlist.text)
12610:          if Connectwebcam(0) then CaptureWebCam(Exepath+'foto4.png
12611:          CloseWebcam();
12612:
12613: Top5 Functions mX4.7.1.80
12614: Proc LoadResourceFile3HTML(aFile:str; ms:TMemoryStream);
12615: Func VarArrayToStr(const vArray: variant):Str;
12616: Func VarStrNull(const V:OleVariant):str; //avoid problems with null strings
12617: Func GetWMIObject(const objectName:Str): IDispatch; //create the Wmi instance
12618: Func GetAntiVirusProductInfo: TStringlist;
12619: Func EnDeCrypt(const Value :Str) :Str;
12620: //http://www.delphibasics.info/home/delphibasicssnippets/executingpreparedshellcodeindelphi
12621:
12622: Proc SIRegister_TeCanvas(CL: TPSPascalCompiler);
12623: //based on TEEProc, TeCanvas, TEEngine, TChart
12624: begin
12625:  'TeePiStep','Double').setExtended( Pi / 180.0);
12626:  'TeeDefaultPerspective','LongInt'( 100);
12627:  'TeeMinAngle','LongInt'( 270);
12628:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12629:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12630:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12631:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12632:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12633:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12634:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12635:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12636:  'TA_LEFT','LongInt'( 0);
12637:  'TA_RIGHT','LongInt'( 2);
12638:  'TA_CENTER','LongInt'( 6);
12639:  'TA_TOP','LongInt'( 0);
12640:  'TA_BOTTOM','LongInt'( 8);
12641:  'teePATCOPY','LongInt'( 0);
12642:  'NumCirclePoints','LongInt'( 64);
12643:  'teeDEFAULT_CHARSET','LongInt'( 1);
```

```
12644:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12645:  'TA_LEFT','LongInt'( 0);
12646:  'bs_Solid','LongInt'( 0);
12647:  'teepf24Bit','LongInt'( 0);
12648:  'teepfDevice','LongInt'( 1);
12649:  'CM_MOUSELEAVE','LongInt'( 10000);
12650:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12651:  'DC_BRUSH','LongInt'( 18);
12652:  'DC_PEN','LongInt'( 19);
12653:   teeCOLORREF', 'LongWord
12654:  TLogBrush', 'record lbStyle : Int; lbColor : TColor; lbHatch: Int; end
12655:  //TNotifyEvent', 'Proc ( Sender : TObject)
12656:  SIRegister_TFilterRegion(CL);
12657:  SIRegister_IFormCreator(CL);
12658:  SIRegister_TTeeFilter(CL);
12659:  //TFilterClass', 'class of TTeeFilter
12660:  SIRegister_TFilterItems(CL);
12661:  SIRegister_TConvolveFilter(CL);
12662:  SIRegister_TBlurFilter(CL);
12663:  SIRegister_TTeePicture(CL);
12664:  TPenEndStyle', '( esRound, esSquare, esFlat )
12665:  SIRegister_TChartPen(CL);
12666:  SIRegister_TChartHiddenPen(CL);
12667:  SIRegister_TDottedGrayPen(CL);
12668:  SIRegister_TDarkGrayPen(CL);
12669:  SIRegister_TWhitePen(CL);
12670:  SIRegister_TChartBrush(CL);
12671:  TTeeView3DScrolled', 'Proc ( IsHoriz :Bool)
12672:  TTeeView3DChangedZoom', 'Proc ( NewZoom : Int)
12673:  SIRegister_TView3DOptions(CL);
12674:  FindClass('TOBJECT'),'TTeeCanvas
12675:  TTeeTransparency', 'Int
12676:  SIRegister_TTeeBlend(CL);
12677:  FindClass('TOBJECT'),'TCanvas3D
12678:  SIRegister_TTeeShadow(CL);
12679:  teeTGradientDirection','(gdTopBottom,gdBottomTop,gdLeftRight,gdRightLeft,gdFromCenter, gdFromTopLeft,
       gdFromBottomLeft,gdRadial,gdDiagonalUp,gdDiagonalDown )
12680:  FindClass('TOBJECT'),'TSubGradient
12681:  SIRegister_TCustomTeeGradient(CL);
12682:  SIRegister_TSubGradient(CL);
12683:  SIRegister_TTeeGradient(CL);
12684:  SIRegister_TTeeFontGradient(CL);
12685:  SIRegister_TTeeFont(CL);
12686:  TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12687:  TCanvasTextAlign', 'Int
12688:   TTeeCanvasHandle', 'HDC
12689:   SIRegister_TTeeCanvas(CL);
12690:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12691:   SIRegister_TFloatXYZ(CL);
12692:   TPoint3D', 'record x : Int; y : Int; z : Int; end
12693:   TRGB', 'record blue: byte; green: byte; red: byte; end
12694:   {TRGB=packed record
12695:     Blue  : Byte;
12696:     Green : Byte;
12697:     Red   : Byte;
12698:  //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12699:
12700: TTeeCanvasCalcPoints','Function(x,z:Int;var P0,P1:TPoint3D;var Color0,Color1:TColor):Bool
12701: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12702: TCanvas3DPlane', '( cpX, cpY, cpZ )
12703:   //IInterface', 'IUnknown
12704:   SIRegister_TCanvas3D(CL);
12705:   SIRegister_TTeeCanvas3D(CL);
12706:   TTrianglePoints', 'Array[0..2] of TPoint;
12707:   TFourPoints', 'Array[0..3] of TPoint;
12708:  Func ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12709:  Func ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12710:  Func Point3D( const x, y, z : Int) : TPoint3D
12711:  Proc SwapDouble( var a, b : Double)
12712:  Proc SwapInt( var a, b : Int)
12713:  Proc RectSize( const R : TRect; var RectWidth, RectHeight : Int)
12714:  Proc teeRectCenter( const R : TRect; var X, Y : Int)
12715:  Func RectFromPolygon( const Points : array of TPoint; NumPoints : Int): TRect
12716:  Func RectFromTriangle( const Points : TTrianglePoints) : TRect
12717:  Func RectangleInRectangle( const Small, Big : TRect) :Bool
12718:  Proc ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12719:  Proc UnClipCanvas( ACanvas : TCanvas)
12720:  Proc ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12721:  Proc ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Int)
12722:  Proc ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Int)
12723:  'TeeCharForHeight','String 'W
12724:  'DarkerColorQuantity','Byte').SetUInt( 128);
12725:  'DarkColorQuantity','Byte').SetUInt( 64);
12726:   TButtonGetColorProc', 'Func  : TColor
12727:   SIRegister_TTeeButton(CL);
12728:   SIRegister_TButtonColor(CL);
12729:   SIRegister_TComboFlat(CL);
12730:  Proc TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12731:  Func TeePoint( const aX, aY : Int) : TPoint
```

```
12732:  Func TEEPointInRect( const Rect : TRect; const P : TPoint) :Bool;
12733:  Func PointInRect1( const Rect : TRect; x, y : Int) :Bool;
12734:  Func TeeRect( Left, Top, Right, Bottom : Int) : TRect
12735:  Func OrientRectangle( const R : TRect) : TRect
12736:  Proc TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Int)
12737:  Func PolygonBounds( const P : array of TPoint) : TRect
12738:  Func PolygonInPolygon( const A, B : array of TPoint) :Bool
12739:  Func RGBValue( const Color : TColor) : TRGB
12740:  Func EditColor( AOwner : TComponent; AColor : TColor) : TColor
12741:  Func EditColorDialog( AOwner : TComponent; var AColor : TColor) :Bool
12742:  Func PointAtDistance( AFrom, ATo : TPoint; ADist : Int) : TPoint
12743:  Func TeeCull( const P : TFourPoints) :Bool;
12744:  Func TeeCull1( const P0, P1, P2 : TPoint) :Bool;
12745:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12746:  Proc SmoothStretch( Src, Dst : TBitmap);
12747:  Proc SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12748:  Func TeeDistance( const x, y : Double) : Double
12749:  Func TeeLoadLibrary( const FileName :Str) : HInst
12750:  Proc TeeFreeLibrary( hLibModule : HMODULE)
12751:  Proc TeeBlendBitmaps(const Percent:Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12752:  //Proc TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12753:  Proc TeeShadowSmooth(Bitmap,Back:TBitmap;Left,Top,Width,Height,horz,
       vert:Int;Smoothness:Double;FullDraw:Bool;ACanvas:TCanvas3D;Clip:Bool)
12754:   SIRegister_ICanvasHyperlinks(CL);
12755:   SIRegister_ICanvasToolTips(CL);
12756:  Func Supports( const Instance : IInterface; const IID : TGUID) :Bool
12757: end;
12758:
12759: Proc SIRegister_ovcmisc(CL: TPSPascalCompiler);
12760: begin
12761:   TOvcHdc', 'Int
12762:   TOvcHWND', 'Cardinal
12763:   TOvcHdc', 'HDC
12764:   TOvcHWND', 'HWND
12765:  Func LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12766:  Func LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12767:  Func ovCompStruct( const S1, S2, Size :Card) : Int
12768:  Func DefaultEpoch : Int
12769:  Func DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12770:  Proc FixRealPrim( P : PChar; DC : Char)
12771:  Func GetDisplayString(Canvas: TCanvas; const S:Str; MinChars, MaxWidth : Int) :Str
12772:  Func GetLeftButton : Byte
12773:  Func GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12774:  Proc GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12775:  Func GetShiftFlags : Byte
12776:  Func ovCreateRotatedFont( F : TFont; Angle : Int) : hFont
12777:  Func GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Int;Ctl3D:Boolean): Int
12778:  Func ovExtractWord( N : Int; const S :Str; WordDelims : TCharSet) :Str
12779:  Func ovIsForegroundTask :Bool
12780:  Func ovTrimLeft( const S :Str) :Str
12781:  Func ovTrimRight( const S :Str) :Str
12782:  Func ovQuotedStr( const S :Str) :Str
12783:  Func ovWordCount( const S :Str; const WordDelims : TCharSet) : Int
12784:  Func ovWordPosition(const N:Int;const S:Str;const WordDelims : TCharSet) : Int
12785:  Func PtrDiff( const P1, P2 : PChar) : Word
12786:  Proc PtrInc( var P, Delta : Word)
12787:  Proc PtrDec( var P, Delta : Word)
12788:  Proc FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Int)
12789:  Proc TransStretchBlt(DstDC:TOvcHdc;DstX,DstY,DstW,DstH:Int;SrcDC:TOvcHdc;SrcX,SrcY,SrcW,
       SrcH:Int;MaskDC:TOvcHdc;MaskX,MaskY:Int)
12790:  Func ovMinI( X, Y : Int) : Int
12791:  Func ovMaxI( X, Y : Int) : Int
12792:  Func ovMinL( X, Y : LongInt) : LongInt
12793:  Func ovMaxL( X, Y : LongInt) : LongInt
12794:  Func GenerateComponentName( PF : TWinControl; const Root :Str) :Str
12795:  Func PartialCompare( const S1, S2 :Str) :Bool
12796:  Func PathEllipsis( const S :Str; MaxWidth : Int) :Str
12797:  Func ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12798:  Proc ovCopyParentImage( Control : TControl; Dest : TCanvas)
12799:  Proc ovDrawTransparentBitmap(Dest:TCanvas;X,Y,W,H:Int;Rect:TRect;Bitmap:TBitmap; TransparentColor:TColor)
12800:  Proc DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,
       Height:Int;Rect:TRect;TransparentColor : TColorRef)
12801:  Func ovWidthOf( const R : TRect) : Int
12802:  Func ovHeightOf( const R : TRect) : Int
12803:  Proc ovDebugOutput( const S :Str)
12804:  Func GetArrowWidth( Width, Height : Int) : Int
12805:  Proc StripCharSeq( CharSeq :Str; var Str :Str)
12806:  Proc StripCharFromEnd( aChr : Char; var Str :Str)
12807:  Proc StripCharFromFront( aChr : Char; var Str :Str)
12808:  Func SystemParametersInfo(uiAction,uiParam: UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12809:  Func SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12810:  Func SystemParametersInfoA(uiAction,uiParam: UINT; pvParam: UINT; fWinIni : UINT) : BOOL
12811:  Func CreateEllipticRgn( p1, p2, p3, p4 : Int) : HRGN
12812:  Func CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12813:  Func CreateFontIndirect( const p1 : TLogFont) : HFONT
12814:  Func CreateMetaFile( p1 : PChar) : HDC
12815:  Func DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12816:  Func DrawText(hDC:HDC;lpString:PChar;nCount:Int;var lpRect: TRect;uFormat:UINT):Int
12817:  Func DrawTextS(hDC:HDC;lpString:str;nCount:Int;var lpRect:TRect;uFormat:UINT):Int
```

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

```
12904:  '_EL','').SetString( _CR);
12905:  '_EOF','Char').SetString( #26);
12906:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12907:  'minErrDist','LongInt'( 2);
12908:  Func ovPadL(S:Str; ch : char; L : Int) :Str
12909: end;
12910:
12911:   TFormatSettings = record
12912:     CurrencyFormat: Byte;
12913:     NegCurrFormat: Byte;
12914:     ThousandSeparator: Char;
12915:     DecimalSeparator: Char;
12916:     CurrencyDecimals: Byte;
12917:     DateSeparator: Char;
12918:     TimeSeparator: Char;
12919:     ListSeparator: Char;
12920:     CurrencyString:Str;
12921:     ShortDateFormat:Str;
12922:     LongDateFormat:Str;
12923:     TimeAMString:Str;
12924:     TimePMString:Str;
12925:     ShortTimeFormat:Str;
12926:     LongTimeFormat:Str;
12927:     ShortMonthNames: array[1..12] of string;
12928:     LongMonthNames: array[1..12] of string;
12929:     ShortDayNames: array[1..7] of string;
12930:     LongDayNames: array[1..7] of string;
12931:     TwoDigitYearCenturyWindow: Word;
12932:   end;
12933:
12934: Proc SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12935: begin
12936:  Func ovFormatSettings : TFormatSettings
12937: end;
12938:
12939: Proc SIRegister_ovcstr(CL: TPSPascalCompiler);
12940: begin
12941:   TOvcCharSet', 'set of Char
12942:   ovBTable', 'array[0..255] of Byte
12943:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12944:  Func BinaryBPChar( Dest : PChar; B : Byte) : PChar
12945:  Func BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12946:  Func BinaryWPChar( Dest : PChar; W : Word) : PChar
12947:  Proc BMMakeTable( MatchString : PChar; var BT : ovBTable)
12948:  Func BMSearch(var Buffer,BufLength:Card;var BT:ovBTable;MatchString:PChar;var Pos:Card):Bool;
12949:  Func BMSearchUC(var Buffer,BufLength:Card;var BT:ovBTable;MatchString:PChar;var Pos:Card):Bool
12950:  Func CharStrPChar( Dest : PChar; C : Char; Len :Card) : PChar
12951:  Func DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12952:  Func HexBPChar( Dest : PChar; B : Byte) : PChar
12953:  Func HexLPChar( Dest : PChar; L : LongInt) : PChar
12954:  Func HexPtrPChar( Dest : PChar; P : TObject) : PChar
12955:  Func HexWPChar( Dest : PChar; W : Word) : PChar
12956:  Func LoCaseChar( C : Char) : Char
12957:  Func OctalLPChar( Dest : PChar; L : LongInt) : PChar
12958:  Func StrChDeletePrim( P : PChar; Pos :Card) : PChar
12959:  Func StrChInsertPrim( Dest : PChar; C : Char; Pos:Card) : PChar
12960:  Func StrChPos( P : PChar; C : Char; var Pos :Card) :Bool
12961:  Proc StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12962:  Func StrStCopy( Dest, S : PChar; Pos, Count :Card) : PChar
12963:  Func StrStDeletePrim( P : PChar; Pos, Count :Card) : PChar
12964:  Func StrStInsert( Dest, S1, S2 : PChar; Pos :Card) : PChar
12965:  Func StrStInsertPrim( Dest, S : PChar; Pos :Card) : PChar
12966:  Func StrStPos( P, S : PChar; var Pos :Card) :Bool
12967:  Func StrToLongPChar( S : PChar; var I : LongInt) :Bool
12968:  Proc TrimAllSpacesPChar( P : PChar)
12969:  Func TrimEmbeddedZeros( const S :Str) :Str
12970:  Proc TrimEmbeddedZerosPChar( P : PChar)
12971:  Func TrimTrailPrimPChar( S : PChar) : PChar
12972:  Func TrimTrailPChar( Dest, S : PChar) : PChar
12973:  Func TrimTrailingZeros( const S :Str) :Str
12974:  Proc TrimTrailingZerosPChar( P : PChar)
12975:  Func UpCaseChar( C : Char) : Char
12976:  Func ovcCharInSet( C : Char; const CharSet : TOvcCharSet) :Bool
12977:  Func ovc32StringIsCurrentCodePage( const S : WideString) :Bool;
12978:  //Func ovc32StringIsCurrentCodePage1( const S:PWideChar; CP :Card) :Bool;
12979: end;
12980:
12981: Proc SIRegister_AfUtils(CL: TPSPascalCompiler);
12982: begin
12983:   //PRaiseFrame', '^TRaiseFrame // will not work
12984:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12985:    +ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12986:  Proc SafeCloseHandle( var Handle : THandle)
12987:  Proc ExchangeInt( X1, X2 : Int)
12988:  Proc FillInt( const Buffer, Size, Value : Int)
12989:  Func LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12990:  Func afCompareMem( P1, P2 : TObject; Length : Int) :Bool
12991:
```

```
12992: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12993:     Func AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12994: Func AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12995:     Func AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12996:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12997:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12998:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12999:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
13000:     Func AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
13001:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
13002:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
13003:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
13004:       ObjectTypeListLength:DWORD;const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
13005:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
13006:     Func AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
13007:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
13008:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
13009:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
13010:       ObjectTypeListLength: DWORD;const GenericMapping:TGenericMapping;  ObjectCreation: BOOL;
13011: var GrantedAccess:DWORD;var AccessStatusList:DWORD;var pfGenerateOnClose:BOOL):BOOL; stdcall;
13012:     Func BackupEventLog(hEventLog:THandle; lpBackupFileName: PKOLChar):BOOL; stdcall;
13013:     Func ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
13014:     Func CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
13015:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
13016:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
13017:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
13018: const lpStartupInfo:TStartupInfo;var lpProcessInformation: TProcessInformation):BOOL; stdcall;
13019:     Func GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
13020:     Func GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
13021: pSecurityDescriptor:PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded:DWORD):BOOL; stdcall;
13022:     Func GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
13023:     Func InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
13024:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
13025:     Func LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
13026:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
13027:     Func LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
13028:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
13029:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
13030:     Func LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
13031:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
13032:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
13033:     Func LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
13034:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
13035:     Func LookupPrivilegeName(lpSystemName: PKOLChar;
13036:       var lpLuid: TLargeInt; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
13037:     Func LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
13038:       var lpLuid: TLargeInt): BOOL; stdcall;
13039:     Func ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
13040:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
13041:     Func ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
13042:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
13043:     Func ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
13044:       ObjectTypeName:PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
13045:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
13046:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
13047:       var GenerateOnClose: BOOL): BOOL; stdcall;
13048:     Func ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
13049:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
13050:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
13051:     Func OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
13052:     Func OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
13053:     Func PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
13054:       ClientToken: THandle;var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
13055:     Func ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
13056:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
13057:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
13058:     Func RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
13059:       var phkResult: HKEY): Longint; stdcall;
13060:     Func RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
13061:       var phkResult: HKEY): Longint; stdcall;
13062:     Func RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
13063:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
13064:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
13065:       lpdwDisposition: PDWORD): Longint; stdcall;
13066:     Func RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
13067:     Func RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
13068:     Func RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
13069:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
13070:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
13071:     Func RegEnumKey(hKey:HKEY;dwIndex:DWORD;lpName:PKOLChar;cbName:DWORD):Longint;stdcall;
13072:     Func RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
13073:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
13074:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
13075:     Func RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
13076:     Func RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
13077:     Func RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
13078:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
13079:     Func RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
13080:       lpcbClass: PDWORD; lpReserved: Pointer;
```

```
13081:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
13082:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
13083:       lpftLastWriteTime: PFileTime): Longint; stdcall;
13084:     Func RegQueryMultipleValues(hKey: HKEY; var ValList;
13085:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
13086:     Func RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
13087:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
13088:     Func RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
13089:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
13090:     Func RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
13091:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
13092:     Func RegRestoreKey(hKey: HKEY; lpFile:PKOLChar; dwFlags: DWORD):Longint; stdcall;
13093:     Func RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
13094:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
13095:     Func RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
13096:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
13097:     Func RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
13098:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
13099:     Func RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
13100:     Func RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
13101:     Func ReportEvent(hEventLog: THandle; wType, wCategory: Word;
13102:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
13103:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
13104:     Func SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
13105:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
13106:  Func wAddAtom( lpString : PKOLChar) : ATOM
13107:  Func wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL): THandle
13108:  //Func wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
        lpOutBuffer:Pointer;nOutBufferSize:DWORD;var lpBytesRead:DWORD;nTimeOut: DWORD) : BOOL
13109:  //Func wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig): BOOL
13110:  Func
        wCompareString(Locale:LCID;dwCmpFlags:DWORD;lpString1:PKOLChar;cchCount1:Int;lpString2:PKOLChar;cchCount2:Int):Int
13111:  Func wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL):BOOL
13112:  //Func wCopyFileEx(lpExistingFileName,
        lpNewFileName:PKOLChar;lpProgressRoutine:TFNProgressRoutine;lpData:Pointer;pbCancel:PBool;
        dwCopyFlags:DWORD):BOOL
13113:  Func wCreateDirectory(lpPathName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
13114:  Func wCreateDirectoryEx(lpTemplateDirectory,lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
13115:  Func wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
        bInitialState:BOOL;lpName:PKOLChar):THandle;
13116:  Func wCreateFile(lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
        PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
13117:  Func wCreateFileMapping(hFile:THandle;lpFileMappingAttributes:PSecurityAttributes; flProtect,
        dwMaximumSizeHigh,dwMaximumSizeLow:DWORD;lpName:PKOLChar):THandle
13118:  Func wCreateHardLink(lpFileName,lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
13119:  Func
        CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
13120:  Func wCreateNamedPipe(lpName:PKOLChar;dwOpenMode,dwPipeMode,nMaxInstances,nOutBufferSize,nInBufferSize,
        nDefaultTimeOut:DWORD;lpSecurityAttributes:PSecurityAttributes): THandle
13121:  //Func CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
        lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
        Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
        lpProcessInfo:TProcessInformation):BOOL
13122:  Func wCreateSemaphore(lpSemaphoreAttributes:PSecurityAttributes;lInitialCount,
        lMaximumCount:Longint;lpName:PKOLChar):THandle
13123:  Func
        wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
13124:  Func wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
13125:  Func wDeleteFile( lpFileName : PKOLChar) : BOOL
13126:  Func wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
13127:  //Func
        wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
13128:  //Func wEnumDateFormats(lpDateFmtEnumProc:TFNDateFmtEnumProc;Locale:LCID;dwFlags:DWORD): BOOL
13129:  //Func wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
13130:  //Func wEnumResourceTypes(hModule:HMODULE;lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
13131:  //Func wEnumSystemCodePages(lpCodePageEnumProc:TFNCodepageEnumProc;dwFlags:DWORD): BOOL
13132:  //Func wEnumSystemLocales(lpLocaleEnumProc:TFNLocaleEnumProc;dwFlags:DWORD): BOOL
13133:  //Func wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
13134:  Func wExpandEnvironmentStrings(lpSrc:PKOLChar;lpDst:PKOLChar;nSize:DWORD): DWORD
13135:  Proc wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
13136:  //Func wFillConsoleOutputCharacter(hConsoleOutput:THandle;cCharacter:KOLChar;nLength:DWORD; dwWriteCoord:
        TCoord;var lpNumberOfCharsWritten:DWORD):BOOL
13137:  Func wFindAtom( lpString : PKOLChar) : ATOM
13138:  Func wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
13139:  Func wFindFirstFile(lpFileName:PKOLChar;var lpFindFileData:TWIN32FindData): THandle
13140:  //Func wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
13141:  Func wFindNextFile(hFindFile:THandle;var lpFindFileData:TWIN32FindData):BOOL
13142:  Func wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
13143:  Func wFindResourceEx(hModule:HMODULE;lpType,lpName:PKOLChar;wLanguage:Word): HRSRC
13144:  Func wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
13145:  //Func wFormatMessage(dwFlags:DWORD;lpSource:Pointer;dwMessageId:DWORD;dwLanguageId:DWORD;
        lpBuffer:PKOLChar;nSize:DWORD;Arguments:Pointer):DWORD
13146:  Func wFreeEnvironmentStrings( EnvBlock : PKOLChar): BOOL
13147:  Func wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Int) : UINT
13148:  Func wGetBinaryType(lpApplicationName:PKOLChar;var lpBinaryType:DWORD):BOOL
13149:  Func wGetCommandLine: PKOLChar
13150:  //Func wGetCompressedFileSize(lpFileName:PKOLChar;lpFileSizeHigh:PDWORD): DWORD
13151:  Func wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
```

```
13152:  Func wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
13153:  //Func wGetCurrencyFormat(Locale:LCID;dwFlags:DWORD;lpValue:PKOLChar;lpFormat :
        PCurrencyFmt;lpCurrencyStr:PKOLChar;cchCurrency:Int):Int
13154:  Func wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
13155:  //Func wGetDateFormat(Locale:LCID; dwFlags:DWORD;lpDate:PSystemTime;lpFormat: PKOLChar;
        lpDateStr:PKOLChar; cchDate: Int):Int
13156:  //Func wGetDefaultCommConfig(lpszName:PKOLChar;var lpCC:TCommConfig;var lpdwSize:DWORD):BOOL
13157:  Func wGetDiskFreeSpace(lpRootPathName:PKOLChar;var lpSectorsPerCluster,lpBytesPerSector,
        lpNumberOfFreeClusters,lpTotalNumberOfClusters:DWORD):BOOL
13158:  //Func wGetDiskFreeSpaceEx(lpDirectoryName:PKOLChar;var lpFreeBytesAvailableToCaller,lpTotalNumberOfBytes,
        lpTotalNumberOfFreeBytes:PLargeInt):BOOL
13159:  Func wGetDriveType( lpRootPathName : PKOLChar) : UINT
13160:  Func wGetEnvironmentStrings : PKOLChar
13161:  Func wGetEnvironmentVariable(lpName:PKOLChar;lpBuffer:PKOLChar;nSize:DWORD): DWORD;
13162:  Func wGetFileAttributes( lpFileName : PKOLChar) : DWORD
13163:  //Func
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
13164:  Func wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
13165:  //Func wGetLocaleInfo(Locale:LCID;LCType:LCTYPE;lpLCData:PKOLChar;cchData:Int):Int
13166:  Func wGetLogicalDriveStrings(nBufferLength:DWORD;lpBuffer:PKOLChar):DWORD
13167:  Func wGetModuleFileName(hModule:HINST;lpFilename:PKOLChar;nSize:DWORD):DWORD
13168:  Func wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
13169:  //Func wGetNamedPipeHandleState(hNamedPipe: THandle;lpState,lpCurInstances,lpMaxCollectionCount,
        lpCollectDataTimeout:PDWORD;lpUserName:PKOLChar;nMaxUserNameSize:DWORD): BOOL
13170: //Func
        wGetNumberFormat(Locale:CID;dwFlags:DWORD;lpValue:PKOLChar;lpFormat:PNumberFmt;lpNumberStr:PKOLChar;cchNumb
13171:  Func wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Int;lpFileName:PKOLChar):UINT;
13172:  Func
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
13173:  Func wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
13174:  Func wGetPrivateProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD
13175:  Func wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Int) : UINT
13176:  Func wGetProfileSection(lpAppName:PKOLChar;lpReturnedString:PKOLChar;nSize:DWORD):DWORD
13177:  Func wGetProfileString(lpAppName,lpKeyName,lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
13178:  Func wGetShortPathName(lpszLongPath:PKOLChar;lpszShortPath:PKOLChar;cchBuffer:DWORD):DWORD
13179:  //Proc wGetStartupInfo( var lpStartupInfo : TStartupInfo)
13180: //Func wGetStringTypeEx(Locale:LCID;dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;var lpCharType):BOOL
13181:  Func wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
13182:  Func wGetTempFileName(lpPathName,lpPrefixString:PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
13183:  Func wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
13184: //Func
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
13185: //Func wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13186: //Func GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize :
        DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength,lpFileSystemFlags:DWORD;
        lpFileSystemNameBuffer:PKOLChar;nFileSystemNameSize: DWORD) : BOOL
13187:  Func wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
13188:  Func wGlobalAddAtom( lpString : PKOLChar) : ATOM
13189:  Func wGlobalFindAtom( lpString : PKOLChar) : ATOM
13190:  Func wGlobalGetAtomName(nAtom:ATOM;lpBuffer:PKOLChar;nSize:Int):UINT
13191:  Func wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
13192:  Func
        wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
13193:  Func wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
13194:  Func wLoadLibraryEx(lpLibFileName:PKOLChar;hFile:THandle;dwFlags:DWORD): HMODULE
13195:  Func wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
13196:  Func wMoveFileEx(lpExistingFileName,lpNewFileName:PKOLChar;dwFlags:DWORD): BOOL
13197:  //Func wMoveFileWithProgress(lpExistingFileName,lpNewFileName:PKOLChar;lpProgressRoutine:
        TFNProgressRoutine;lpData:Pointer;dwFlags:DWORD): BOOL
13198:  Func wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
13199:  Func wOpenFileMapping(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpName: PKOLChar):THandle
13200:  Func wOpenMutex(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpName:PKOLChar): THandle
13201:  Func wOpenSemaphore( dwDesiredAccess:DWORD; bInheritHandle:BOOL;lpName:PKOLChar):THandle
13202:  Func wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
13203:  Proc wOutputDebugString( lpOutputString : PKOLChar)
13204: //Func wPeekConsoleInput(hConsoleInput:THandle;varlpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsRead:DWORD):BOOL;
13205:  Func wQueryDosDevice(lpDeviceName:PKOLChar;lpTargetPath:PKOLChar;ucchMax:DWORD):DWORD
13206:  //Func wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
13207:  //Func wReadConsole(hConsoleInput:THandle;lpBuffer:Pointer;nNumberOfCharsToRead:DWORD; var
        lpNumberOfCharsRead:DWORD;lpReserved Pointer):BOOL
13208:  //Func wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
        lpNumbOfEventsRead:DWORD):BOOL;
13209:  //Func wReadConsoleOutput(hConsoleOutput:THandle;lpBuffer:Pointer;dwBufferSize,dwBufferCoord:TCoord;var
        lpReadRegion:TSmallRect) : BOOL
13210:  //Func wReadConsoleOutputCharacter(hConsoleOutput:THandle;lpCharacter:PKOLChar;nLength :
        DWORD;dwReadCoord:TCoord;var lpNumberOfCharsRead:DWORD): BOOL
13211:  Func wRemoveDirectory( lpPathName : PKOLChar) : BOOL
13212:  //Func wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
        lpClipRectangle:PSmallRect;dwDestinationOrigin:TCoord;var lpFill:TCharInfo):BOOL
13213:  Func wSearchPath(lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
13214:  Func wSetComputerName( lpComputerName: PKOLChar): BOOL
13215:  Func wSetConsoleTitle( lpConsoleTitle: PKOLChar): BOOL
13216:  Func wSetCurrentDirectory( lpPathName: PKOLChar): BOOL
13217:  //Func wSetDefaultCommConfig(lpszName: PKOLChar; lpCC:PCommConfig; dwSize: DWORD) : BOOL
13218:  Func wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
```

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

```
13289: // Func wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
13290:  Func wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
13291:  Func wGetEnhMetaFileDescription(p1 HENHMETAFILE;p2:UINT;p3:PKOLChar):UINT
13292: // Func wGetGlyphIndices(DC:HDC;p2:PKOLChar;p3:Int;p4:PWORD;p5:DWORD): DWORD
13293: //Func wGetGlyphOutline(DC:HDC;uChar,uFormat:UINT;const
       lpgm:TGlyphMetrics;cbBuffer:DWORD;lpvBuffer:Pointer;const lpmat2:TMat2): DWORD
13294:  Func wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
13295: // Func wGetLogColorSpace(p1:HCOLORSPACE;var ColorSpace:TLogColorSpace;Size:DWORD):BOOL
13296:  Func wGetMetaFile( p1 : PKOLChar) : HMETAFILE
13297: // Func wGetObject( p1 : HGDIOBJ; p2 : Int; p3 : Pointer) : Int
13298:  //Func wGetOutlineTextMetrics(DC:HDC;p2:UINT;OTMetricStructs:Pointer): UINT
13299:  //Func wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Int;p5,p6:PInt;var p7:TSize):BOOL
13300:  Func wGetTextExtentPoint32(DC:HDC;Str:PKOLChar;Count:Int;var Size:TSize):BOOL
13301:  Func wGetTextExtentPoint(DC:HDC;Str:PKOLChar;Count:Int;var Size:TSize):BOOL
13302:  Func wGetTextFace( DC : HDC; Count : Int; Buffer : PKOLChar) : Int
13303:  //Func wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
13304:  Func wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Int) : BOOL
13305:  Func wRemoveFontResource( FileName : PKOLChar) : BOOL
13306:  //Func wRemoveFontResourceEx(p1:PKOLChar;p2 DWORD;p3: PDesignVector): BOOL
13307:  //Func wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
13308:  Func wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
13309:  //Func wStartDoc( DC : HDC; const p2 : TDocInfo) : Int
13310:  Func wTextOut( DC : HDC; X, Y : Int; Str : PKOLChar; Count : Int) : BOOL
13311:  Func wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
13312:  Func wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
13313:  //Func wwglUseFontOutlines(p1:HDC;p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
13314:  Func wAppendMenu(hMenu:HMENU;uFlags,uIDNewItem:UINT;lpNewItem:PKOLChar):BOOL
13315:  Func wCallMsgFilter( var lpMsg : TMsg; nCode : Int) : BOOL
13316:  //Func wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
13317:  //Func wChangeDisplaySettings(var lpDevMode:TDeviceMode; dwFlags : DWORD) : Longint
13318: // Func wChangeDisplaySettingsEx(lpszDeviceName:PKOLChar;var lpDevMode:
       TDeviceMode;wnd:HWND;dwFlags:DWORD;lParam:Pointer):Longint
13319:  Func wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
13320:  Func wCharLower( lpsz : PKOLChar) : PKOLChar
13321:  Func wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
13322:  Func wCharNext( lpsz : PKOLChar) : PKOLChar
13323:  //Func wCharNextEx(CodePage:Word;lpCurrentChar:LPCSTR;dwFlags:DWORD): LPSTR
13324:  Func wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
13325: // Func wCharPrevEx(CodePage:Word;lpStart,lpCurrentChar:LPCSTR; dwFlags:DWORD) : LPSTR
13326:  Func wCharToOem(lpszSrc:PKOLChar; lpszDst : PKOLChar) : BOOL
13327:  Func wCharToOemBuff(lpszSrc:PKOLChar;lpszDst:PKOLChar;cchDstLength:DWORD): BOOL
13328:  Func wCharUpper( lpsz : PKOLChar) : PKOLChar
13329:  Func wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
13330:  Func wCopyAcceleratorTable(hAccelSrc:HACCEL;var lpAccelDst,cAccelEntries:Int) :Int
13331:  Func wCreateAcceleratorTable( var Accel, Count : Int) : HACCEL
13332:  //Func wCreateDesktop(lpszDesktop,
       lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
13333:  //Func wCreateDialogIndirectParam(hInstance:HINST;const lpTemplate:TDlgTemplate;hWndParent
       :HWND;lpDialogFunc:TFNDlgProc;dwInitParam:LPARAM):HWND
13334:  //Func
       wCreateDialogParam(hInstance:HINST;lpTemplateName:PKOLChar;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitPa
13335:  Func wCreateMDIWindow(lpClassName,lpWindowName:PKOLChar;dwStyle:DWORD;X,Y,nWidth,
       nHeight:Int;hWndParent:HWND;hInstance:HINST;lParam:LPARAM):HWND
13336: //Func wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,nWidth,
        nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
13337:  //Func wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
       dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
13338:  Func wDefDlgProc(hDlg:HWND;Msg:UINT;wParam WPARAM;lParam:LPARAM):LRESULT
13339:  Func wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
13340:  Func wDefMDIChildProc(hWnd:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
13341:  Func wDefWindowProc(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM): RESULT
13342:  //Func wDialogBoxIndirectParam(hInstance:HINST;const
       lpDialogTemplate:TDlgTemplate;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitParam:LPARAM):Int
13343:  //Func
       wDialogBoxParam(hInstance:HINST;lpTemplateName:PKOLChar;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitParam
       LPARAM): Int
13344:  Func wDispatchMessage(const lpMsg : TMsg) : Longint
13345:  Func wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,nIDStaticPath:Int;uFileType:UINT):Int;
13346:  Func wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,nIDStaticPath:Int;uFiletype:UINT):Int;
13347:  Func wDlgDirSelectComboBoxEx(hDlg:HWND;lpString:PKOLChar;nCount,nIDComboBox:Int): BOOL
13348:  Func wDlgDirSelectEx(hDlg:HWND;lpString:PKOLChar;nCount,nIDListBox:Int):BOOL
13349: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
       cy:Int;Flags:UINT):BOOL;
13350:  Func wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Int;var lpRect:TRect;uFormat:UINT):Int;
13351:  Func wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
13352:  Func wFindWindowEx(Parent,Child:HWND;ClassName,WindowName:PKOLChar): HWND
13353:  //Func wGetAltTabInfo(hwnd:HWND;iItem:Int;var
       pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
13354: // Func wGetClassInfo(hInstance:HINST;lpClassName:PKOLChar;var lpWndClass:TWndClass):BOOL
13355:  //Func wGetClassInfoEx(Instance:HINST;Classname:PKOLChar;var WndClass:TWndClassEx):BOOL
13356:  Func wGetClassLong( hWnd : HWND; nIndex : Int) : DWORD
13357:  Func wGetClassName(hWnd:HWND; lpClassName:PKOLChar;nMaxCount:Int) : Int
13358:  Func wGetClipboardFormatName(format:UINT;lpszFormatName:PKOLChar;cchMaxCount:Int):Int;
13359:  Func wGetDlgItemText(hDlg:HWND;nIDDlgItem:Int;lpString:PKOLChar;nMaxCount:Int):UINT
13360:  Func wGetKeyNameText(lParam:Longint;lpString:PKOLChar;nSize:Int):Int
13361:  Func wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
13362:  //Func wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
13363:  Func wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Int;uFlag:UINT):Int;
13364:  Func wGetMessage(var lpMsg:TMsg;hWnd:HWND;wMsgFilterMin wMsgFilterMax:UINT):BOOL
```

```
13365:  Func wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
13366: //Func wGetTabbedTextExtent(hDC:HDC;lpString:PKOLChar;nCount,nTabPositions:Int;var
       lpnTabStopPositions):DWORD
13367: //Func wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var lpnLengthNeed:DWORD)BOOL;
13368:  Func wGetWindowLong( hWnd : HWND; nIndex : Int) : Longint
13369:  Func wGetWindowModuleFileName(hwnd:HWND;pszFileName:PKOLChar;cchFileNameMax:UINT): UINT
13370:  Func wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Int) : Int
13371:  Func wGetWindowTextLength( hWnd : HWND) : Int
13372: //Func wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
       nHeigt:Int):BOOL;
13373:  Func wInsertMenu(hMenu:HMENU;uPosition,uFlags,uIDNewItem:UINT; lpNewItem: PKOLChar): BOOL
13374:  //Func wInsertMenuItem(p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13375:  Func wIsCharAlpha( ch : KOLChar) : BOOL
13376:  Func wIsCharAlphaNumeric( ch : KOLChar) : BOOL
13377:  Func wIsCharLower( ch : KOLChar) : BOOL
13378:  Func wIsCharUpper( ch : KOLChar) : BOOL
13379:  Func wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
13380:  Func wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
13381:  Func wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
13382:  Func wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
13383:  Func wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
13384:  Func wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
13385:  Func wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Int;Flags:UINT): THandle
13386:  Func wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
13387:  Func wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
13388:  //Func wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
13389:  Func wLoadString(hInstance:HINST;uID:UINT;lpBuffer:PKOLChar;nBufferMax:Int):Int
13390:  Func wMapVirtualKey( uCode, uMapType : UINT) : UINT
13391:  Func wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
13392:  Func wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Int
13393:  Func wMessageBoxEx(hWnd:HWND;lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word)Int
13394:  //Func wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
13395:  Func wModifyMenu(hMnu: HMENU; uPosition,uFlags,uIDNewItem:UINT; lpNewItem:PKOLChar): BOOL
13396:  //Func wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
13397:  //7Func wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
13398:  //Func wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
13399:  Func wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
13400:  Func wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
13401:  Func wOpenWindowStation(lpszWinSta:PKOLChar;fInherit:BOOL;dwDesiredAccess:DWORD):HWINSTA
13402:  Func wPeekMessage(var lpMsg:TMsg;hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
13403:  Func wPostMessage(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):BOOL
13404:  Func wPostThreadMessage(idThread:DWORD;Msg:UINT;wParam:WPARAM;lParam:LPARAM): BOOL
13405:  Func wRealGetWindowClass(hwnd:HWND;pszType PKOLChar;cchType:UINT):UINT
13406: // Func wRegisterClass( const lpWndClass : TWndClass) : ATOM
13407: // Func wRegisterClassEx( const WndClass : TWndClassEx): ATOM
13408:  Func wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
13409: //Func wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13410:  Func wRegisterWindowMessage( lpString : PKOLChar) : UINT
13411:  Func wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
13412:  Func wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13413:  Func wSendMessage(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM): LRESULT
13414:  //Func wSendMessageCallback(hWnd: HWND; Msg:UINT;wParam:
       WPARAM;lParam:LPARAM;lpResultCallBack:TFNSendAsyncProc;dwData DWORD):BOOL
13415:  Func wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM;fuFlags,uTimeout:UINT;var
       lpdwResult:DWORD):LRESULT
13416:  Func wSendNotifyMessage(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):BOOL
13417:  Func wSetClassLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint) : DWORD
13418:  Func wSetDlgItemText(hDlg: HWND;nIDDlgItem:Int;lpString : PKOLChar) : BOOL
13419:  //Func wSetMenuItemInfo(p1:HMENU;p2:UINT;p3:BOOL;const p4:TMenuItemInfo):BOOL
13420:  Func wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13421: // Func wSetUserObjectInformation(hObj:THandle;nIndex:Int;pvInfo:Pointer;nLength:DWORD):BOOL
13422:  Func wSetWindowLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint) : Longint
13423:  Func wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13424:  //Func wSetWindowsHook(nFilterType:Int; pfnFilterProc:TFNHookProc): HHOOK
13425:  //Func wSetWindowsHookEx(idHook:Int;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13426: // Func wSystemParametersInfo(uiAction,uiParam:UINT; pvParam:Pointer; fWinIni: UINT):BOOL
13427:  Func wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var lpnTabStopPositions,
       nTabOrigin:Int):Longint;
13428:  Func wTranslateAccelerator(hWnd:HWND;hAccTable:HACCEL;var lpMsg:TMsg): Int
13429:  Func wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13430:  Func wVkKeyScan( ch : KOLChar) : SHORT
13431:  Func wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13432:  Func wWinHelp(hWndMain:HWND;lpszHelp:PKOLChar;uCommand:UINT;dwData:DWORD):BOOL
13433:  Func wwsprintf( Output : PKOLChar; Format : PKOLChar) : Int
13434:  Func wwvsprintf( Output : PKOLChar; Format: PKOLChar; arglist:va_list): Int
13435:
13436: //TestDrive!
13437: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13438:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13439:  Func GetDomainUserSidS(const domainName:Str;const userName:Str;var foundDomain:Str):str;
13440:  Func GetLocalUserSidStr( const UserName :Str) :Str
13441:  Func getPid4user(const domain:str;const user:str;var pid:dword):boolean
13442:  Func Impersonate2User( const domain :Str; const user :Str):Bool
13443:  Func GetProcessUserBypid(pid:DWORD;var UserName,Domain:Ansistr):Bool
13444:  Func KillProcessbyname( const exename :Str; var found : Int) : Int
13445:  Func getWinProcessList : TStringList
13446:  Func WaitTilClose(hWnd: Int): Int;
13447:  Func DoUserMsgs:Bool;
13448:  Func MsgFunc(hWnd,Msg,wParam,lParam:Int):Int; stdcall;
```

```
13449:  Proc ShowMsg(hParent:Int; const Mess, Title:Str);  //modal but NOT blockable
13450:  Proc DeleteMsgForm(Handle: Int);
13451:  Proc DisableForms;
13452:  Func FoundTopLevel(hWnd, LParam: Int): BOOL; StdCall;
13453: end;
13454:
13455: Proc SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13456: begin
13457:  'AfMaxSyncSlots','LongInt'( 64);
13458:  'AfSynchronizeTimeout','LongInt'( 2000);
13459:   TAfSyncSlotID', 'DWORD
13460:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13461:   TAfSafeSyncEvent', 'Proc ( ID : TAfSyncSlotID)
13462:   TAfSafeDirectSyncEvent', 'Procedure
13463:  Func AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13464:  Func AfReleaseSyncSlot( const ID : TAfSyncSlotID) :Bool
13465:  Func AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable :Bool) :Bool
13466:  Func AfValidateSyncSlot( const ID : TAfSyncSlotID) :Bool
13467:  Func AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) :Bool
13468:  Func AfDirectSyncEvent(Event:TAfSafeDirectSyncEvent;Timeout:DWORD):Boolean
13469:  Func AfIsSyncMethod :Bool
13470:  Func AfSyncWnd : HWnd
13471:  Func AfSyncStatistics : TAfSyncStatistics
13472:  Proc AfClearSyncStatistics
13473: end;
13474:
13475: Proc SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13476: begin
13477:  'fBinary','LongWord')( $00000001);
13478:  'fParity','LongWord'( $00000002);
13479:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13480:  'fOutxDsrFlow','LongWord')( $00000008);
13481:  'fDtrControl','LongWord')( $00000030);
13482:  'fDtrControlDisable','LongWord')( $00000000);
13483:  'fDtrControlEnable','LongWord')( $00000010);
13484:  'fDtrControlHandshake','LongWord')( $00000020);
13485:  'fDsrSensitivity','LongWord')( $00000040);
13486:  'fTXContinueOnXoff','LongWord')( $00000080);
13487:  'fOutX','LongWord')( $00000100);
13488:  'fInX','LongWord')( $00000200);
13489:  'fErrorChar','LongWord')( $00000400);
13490:  'fNull','LongWord')( $00000800);
13491:  'fRtsControl','LongWord')( $00003000);
13492:  'fRtsControlDisable','LongWord')( $00000000);
13493:  'fRtsControlEnable','LongWord')( $00001000);
13494:  'fRtsControlHandshake','LongWord')( $00002000);
13495:  'fRtsControlToggle','LongWord')( $00003000);
13496:  'fAbortOnError','LongWord')( $00004000);
13497:  'fDummy2','LongWord')( $FFFF8000);
13498:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13499:   FindClass('TOBJECT'),'EAfComPortCoreError
13500:   FindClass('TOBJECT'),'TAfComPortCore
13501:   TAfComPortCoreEvent', 'Proc ( Sender : TAfComPortCore; Even'
13502:    +'tKind : TAfCoreEvent; Data : DWORD)
13503:   SIRegister_TAfComPortCoreThread(CL);
13504:   SIRegister_TAfComPortEventThread(CL);
13505:   SIRegister_TAfComPortWriteThread(CL);
13506:   SIRegister_TAfComPortCore(CL);
13507:  Func FormatDeviceName( PortNumber : Int) :Str
13508: end;
13509:
13510: Proc SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13511: begin
13512:   TAFIOFileStreamEvent', 'Func ( const fileName :Str; mode: Word) : TStream
13513:   TAFIOFileStreamExistsEvent', 'Func ( const fileName :Str) :Bool
13514:   SIRegister_TApplicationFileIO(CL);
13515:   TDataFileCapability', '( dfcRead, dfcWrite )
13516:   TDataFileCapabilities', 'set of TDataFileCapability
13517:   SIRegister_TDataFile(CL);
13518:   //TDataFileClass', 'class of TDataFile
13519:  Func ApplicationFileIODefined :Bool
13520:  Func CreateFileStream(const fileName:str;mode:WordfmShareDenyNone):TStream
13521:  Func FileStreamExists(const fileName:Str):Bool
13522:  //Proc Register
13523: end;
13524:
13525: Proc SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13526: begin
13527:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13528:    +', uftCstring, uftSmallint, uftInt, uftQuad, uftFloat, uftDoublePrecisi'
13529:    +'on,uftTimestamp,uftBlob,uftBlobId,uftDate,uftTime,uftInt64,uftArray,uftNull)
13530:   TALFBXScale', 'Int
13531:   FindClass('TOBJECT'),'EALFBXConvertError
13532:   SIRegister_EALFBXError(CL);
13533:   SIRegister_EALFBXException(CL);
13534:   FindClass('TOBJECT'),'EALFBXGFixError
13535:   FindClass('TOBJECT'),'EALFBXDSQLError
13536:   FindClass('TOBJECT'),'EALFBXDynError
13537:   FindClass('TOBJECT'),'EALFBXGBakError
```

```
13538:   FindClass('TOBJECT'),'EALFBXGSecError
13539:   FindClass('TOBJECT'),'EALFBXLicenseError
13540:   FindClass('TOBJECT'),'EALFBXGStatError
13541:   //EALFBXExceptionClass', 'class of EALFBXError
13542:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13543:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13544:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13545:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13546:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13547:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13548:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13549:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13550:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13551:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13552:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13553:    +'Commit,tpRecVersion,tpNoRecVersion,tpRestartRequests,tpNoAutoUndo, tpLockTimeout)
13554:   TALFBXTransParams', 'set of TALFBXTransParam
13555:  Func ALFBXStrToCharacterSet( const CharacterSet : Ansistr) : TALFBXCharacterSet
13556:  Func ALFBXCreateDBParams( Params : Ansistr; Delimiter : Char) : Ansistr
13557:  Func ALFBXCreateBlobParams( Params : Ansistr; Delimiter : Char) : Ansistr
13558:  'cALFBXMaxParamLength','LongInt'( 125);
13559:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13560:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13561:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13562:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13563:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13564:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13565:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13566:   SIRegister_TALFBXSQLDA(CL);
13567:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13568:   SIRegister_TALFBXPoolStream(CL);
13569:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13570:   TALFBXBlobData', 'record Size : Int; Buffer :Str; end
13571:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13572:   //TALFBXArrayDesc', 'TISCArrayDesc
13573:   //TALFBXBlobDesc', 'TISCBlobDesc
13574:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13575:   //TALFBXArrayInfo', 'record index : Int; size:Int;info: TALFBXArrayDesc; end
13576:   SIRegister_TALFBXSQLResult(CL);
13577:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13578:   SIRegister_TALFBXSQLParams(CL);
13579:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13580:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13581:    +'atementType : TALFBXStatementType; end
13582:   FindClass('TOBJECT'),'TALFBXLibrary
13583:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13584:   TALFBXOnConnectionLost', 'Proc ( Lib : TALFBXLibrary)
13585:   //TALFBXOnGetDBExceptionClass', 'Proc ( Number : Int; out'
13586:    //+' Excep : EALFBXExceptionClass)
13587:   SIRegister_TALFBXLibrary(CL);
13588:  'cAlFBXDateOffset','LongInt'( 15018);
13589:  'cALFBXTimeCoeff','LongInt'( 864000000);
13590:  //Proc ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13591:  //Proc ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13592:  //Func ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13593:  Proc ALFBXDecodeSQLDate( v : Int; out Year : SmallInt; out Month, Day : Word)
13594:  Proc ALFBXDecodeSQLTime(v:Card;out Hour,Minute,Second:Word;out Fractions: LongWord)
13595:  //Proc ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13596:  //Proc ALFBXEncodeTimeStamp1( const Date : Int; v : PISCTimeStamp);
13597:  //Proc ALFBXEncodeTimeStamp2( const Time :Card; v : PISCTimeStamp);
13598:  Func ALFBXEncodeSQLDate(Year : Int; Month, Day : Int) : Int
13599:  Func ALFBXEncodeSQLTime(Hour,Minute,Second:Word;var Fractions:LongWord):Card
13600:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13601:   TALFBXDPBInfo', 'record Name : Ansistr; ParamType : TALFBXParamType; end
13602:  Func ALFBXSQLQuote( const name : Ansistr) : Ansistr
13603:  Func ALFBXSQLUnQuote( const name : Ansistr) : Ansistr
13604: end;
13605:
13606: Proc SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13607: begin
13608:   TALFBXClientSQLParam', 'record Value : Ansistr; IsNull :Bool; end
13609:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13610:   TALFBXClientSelectDataSQL', 'record SQL : Ansistr; Params : T'
13611:    +ALFBXClientSQLParams; RowTag : Ansistr; ViewTag : Ansistr; Skip : in'
13612:    +teger; First : Int; CacheThreshold : Int; end
13613:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13614:   TALFBXClientUpdateDataSQL','record SQL : Ansistr; Params: TALFBXClientSQLParams; end
13615:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13616:   TALFBXClientMonitoringIOStats','record
       page_reads:int64;page_writes:int64;page_fetches:int64;page_marks:int64;end
13617:   SIRegister_TALFBXClient(CL);
13618:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13619:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13620:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13621:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13622:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13623:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13624:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13625:   SIRegister_TALFBXConnectionPoolClient(CL);
```

```
13626:   SIRegister_TALFBXEventThread(CL);
13627:   Func AlMySqlClientSlashedStr( const Str : Ansistr) : Ansistr
13628: end;
13629:
13630: Proc SIRegister_ovcBidi(CL: TPSPascalCompiler);
13631: begin
13632: _OSVERSIONINFOA = record
13633:   dwOSVersionInfoSize: DWORD;
13634:   dwMajorVersion: DWORD;
13635:   dwMinorVersion: DWORD;
13636:   dwBuildNumber: DWORD;
13637:   dwPlatformId: DWORD;
13638:   szCSDVersion: array[0..127] of AnsiChar;{ Maintenance Ansistr for PSS usage}
13639: end;
13640:  TOSVersionInfoA', '_OSVERSIONINFOA
13641:   TOSVersionInfo', 'TOSVersionInfoA
13642:  'WS_EX_RIGHT','LongWord')( $00001000);
13643:  'WS_EX_LEFT','LongWord')( $00000000);
13644:  'WS_EX_RTLREADING','LongWord')( $00002000);
13645:  'WS_EX_LTRREADING','LongWord')( $00000000);
13646:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13647:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13648:  Func SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13649:  'LAYOUT_RTL','LongWord')( $00000001);
13650:  'LAYOUT_BTT','LongWord')( $00000002);
13651:  'LAYOUT_VBH','LongWord')( $00000004);
13652:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13653:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13654:  Func SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13655:  Func GetLayout( dc : hdc) : DWORD
13656:  Func IsBidi :Bool
13657:  Func GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13658:  Func GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13659:  Func SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13660:  Func GetPriorityClass( hProcess : THandle) : DWORD
13661:  Func OpenClipboard( hWndNewOwner : HWND) : BOOL
13662:  Func CloseClipboard : BOOL
13663:  Func GetClipboardSequenceNumber : DWORD
13664:  Func GetClipboardOwner : HWND
13665:  Func SetClipboardViewer( hWndNewViewer : HWND) : HWND
13666:  Func GetClipboardViewer : HWND
13667:  Func ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13668:  Func SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13669:  Func GetClipboardData( uFormat : UINT) : THandle
13670:  Func RegisterClipboardFormat( lpszFormat : PChar) : UINT
13671:  Func CountClipboardFormats : Int
13672:  Func EnumClipboardFormats( format : UINT) : UINT
13673:  Func GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Int):Int
13674:  Func EmptyClipboard : BOOL
13675:  Func IsClipboardFormatAvailable( format : UINT) : BOOL
13676:  Func GetPriorityClipboardFormat(var paFormatPriorityList,cFormats:Int):Int
13677:  Func GetOpenClipboardWindow : HWND
13678:  Func EndDialog( hDlg : HWND; nResult : Int) : BOOL
13679:  Func GetDlgItem( hDlg : HWND; nIDDlgItem : Int) : HWND
13680:  Func SetDlgItemInt(hDlg:HWND;nIDDlgItem:Int;uValue:UINT;bSigned:BOOL): BOOL
13681:  Func GetDlgItemInt(hDlg:HWND;nIDDlgItem:Int;var lpTranslated:BOOL;bSigned: BOOL):UINT
13682:  Func SetDlgItemText( hDlg : HWND; nIDDlgItem : Int; lpString : PChar) : BOOL
13683:  Func CheckDlgButton( hDlg : HWND; nIDButton : Int; uCheck : UINT) : BOOL
13684:  Func CheckRadioButton(hDlg:HWND;nIDFirstButton,nIDLastButton,nIDCheckButton:Int):BOOL
13685:  Func IsDlgButtonChecked( hDlg : HWND; nIDButton : Int) : UINT
13686:  Func SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13687: end;
13688:
13689: Proc SIRegister_DXPUtils(CL: TPSPascalCompiler);
13690: begin
13691:  Func glExecuteAndWait(cmdLine:Str;visibility:Word;timeout:Card;killAppOnTimeOut:Bool):Int;
13692:  Func GetTemporaryFilesPath :Str
13693:  Func GetTemporaryFileName :Str
13694:  Func FindFileInPaths( const fileName, paths :Str) :Str
13695:  Func PathsToString( const paths : TStrings) :Str
13696:  Proc StringToPaths( const pathsString :Str; paths : TStrings)
13697:  //Func MacroExpandPath( const aPath :Str) :Str
13698: end;
13699:
13700: Proc SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13701: begin
13702:   SIRegister_TALMultiPartBaseContent(CL);
13703:   SIRegister_TALMultiPartBaseContents(CL);
13704:   SIRegister_TAlMultiPartBaseStream(CL);
13705:   SIRegister_TALMultipartBaseEncoder(CL);
13706:   SIRegister_TALMultipartBaseDecoder(CL);
13707:  Func ALMultipartExtractBoundaryFromContentType(aContentType:Ansistr): Ansistr
13708:  Func ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:Ansistr;aName:Ansistr):Ansistr;
13709:  Func ALMultipartSetSubValueInHeaderLine(aHeaderLine:Ansistr;aName,AValue:Ansistr):Ansistr;
13710: end;
13711:
13712: Proc SIRegister_SmallUtils(CL: TPSPascalCompiler);
13713: begin
13714:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
```

```
13715:   TWinVerRec','record WinPlatform : Int; WinMajorVersion : In'
13716:    +teger; WinMinorVersion :Int; WinBuildNumber : Int; WinCSDVersion:Str; end
13717:  Func aAllocPadedMem( Size :Card) : TObject
13718:  Proc aFreePadedMem( var P : TObject);
13719:  Proc aFreePadedMem1( var P : PChar);
13720:  Func aCheckPadedMem( P : Pointer) : Byte
13721:  Func aGetPadMemSize( P : Pointer) :Card
13722:  Func aAllocMem( Size :Card) : Pointer
13723:  Func aStrLen( const Str : PChar) :Card
13724:  Func aStrLCopy( Dest : PChar; const Source : PChar; MaxLen :Card) : PChar
13725:  Func aStrECopy( Dest : PChar; const Source : PChar) : PChar
13726:  Func aStrCopy( Dest : PChar; const Source : PChar) : PChar
13727:  Func aStrEnd( const Str : PChar) : PChar
13728:  Func aStrScan( const Str : PChar; aChr : Char) : PChar
13729:  Func aStrMove( Dest : PChar; const Source : PChar; Count :Card) : PChar
13730:  Func aPCharLength( const Str : PChar) :Card
13731:  Func aPCharUpper( Str : PChar) : PChar
13732:  Func aPCharLower( Str : PChar) : PChar
13733:  Func aStrCat( Dest : PChar; const Source : PChar) : PChar
13734:  Func aLastDelimiter( const Delimiters, S :Str) : Int
13735:  Func aCopyTail( const S :Str; Len : Int) :Str
13736:  Func aInt2Thos( I : Int64) :Str
13737:  Func aUpperCase( const S :Str) :Str
13738:  Func aLowerCase( const S :Str) :Str
13739:  Func aCompareText( const S1, S2 :Str) : Int
13740:  Func aSameText( const S1, S2 :Str) :Bool
13741:  Func aInt2Str( Value : Int64) :Str
13742:  Func aStr2Int( const Value :Str) : Int64
13743:  Func aStr2IntDef( const S :Str; Default : Int64) : Int64
13744:  Func aGetFileExt( const FileName :Str) :Str
13745:  Func aGetFilePath( const FileName :Str) :Str
13746:  Func aGetFileName( const FileName :Str) :Str
13747:  Func aChangeExt( const FileName, Extension :Str) :Str
13748:  Func aAdjustLineBreaks( const S :Str) :Str
13749:  Func aGetWindowStr( WinHandle : HWND) :Str
13750:  Func aDiskSpace( Drive :Str) : TdriveSize
13751:  Func aFileExists( FileName :Str) :Bool
13752:  Func aFileSize( FileName :Str) : Int64
13753:  Func aDirectoryExists( const Name :Str) :Bool
13754:  Func aSysErrorMessage( ErrorCode : Int) :Str
13755:  Func aShortPathName( const LongName :Str) :Str
13756:  Func aGetWindowVer : TWinVerRec
13757:  Proc InitDriveSpacePtr;
13758: end;
13759:
13760: Proc SIRegister_MakeApp(CL: TPSPascalCompiler);
13761: begin
13762:  aZero','LongInt'( 0);
13763:  'makeappDEF','LongInt'( - 1);
13764:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13765:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13766:  'CS_KEYCVTWINDOW','LongInt'( 4);
13767:  'CS_DBLCLKS','LongInt'( 8);
13768:  'CS_OWNDC','LongWord')( $20);
13769:  'CS_CLASSDC','LongWord')( $40);
13770:  'CS_PARENTDC','LongWord')( $80);
13771:  'CS_NOKEYCVT','LongWord')( $100);
13772:  'CS_NOCLOSE','LongWord')( $200);
13773:  'CS_SAVEBITS','LongWord')( $800);
13774:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13775:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13776:  'CS_GLOBALCLASS','LongWord')( $4000);
13777:  'CS_IME','LongWord')( $10000);
13778:  'CS_DROPSHADOW','LongWord')( $20000);
13779:   //PPanelFunc', '^TPanelFunc // will not work
13780:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13781:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13782:   TFontLooks', 'set of TFontLook
13783:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Int):Int)
13784:  Func SetWinClass(const ClassName:str; pMessFunc: Tmessagefunc; wcStyle : Int): Word
13785:  Func SetWinClassO( const ClassNam:str; pMessFunc: TObject; wcStyle : Int): Word
13786:  Func SetWinClass(const ClassName:str; pMessFunc : TObject; wcStyle : Int) : Word
13787:  Func MakeForm(Left,Top,Width,Height:Int;const Caption:str;WinStyle:Int):Int
13788:  Proc RunMsgLoop( Show :Bool)
13789:  Func MakeFont(Height,Width:Int;const FontName:str; Look:TFontLooks;Roman:Boolean):Int
13790:  Func MakeButton(Left,Top,Width,Height:Int;pCaption:PChar;hParent,ID_Number:Cardi;hFont:Int):Int;
13791:  Func MakeListBox(Left,Top,Width,Height,Parent:Int;const ListItems:str;WinStyle:Int):Int
13792:  Func MakeComboBox(Left,Top,Width,Height,Parent:Int;const ListItems:str;WinStyle:Int):Int
13793:  Func MakePanel(Left,Top,Width,Height,hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13794:  Func MakeSubMenu(const ItemList :Str; ID1, ID2 :Card; hMenu : Int) : Int
13795:  Func id4menu( a, b : Byte; c : Byte; d : Byte) :Card
13796:  Proc DoInitMakeApp   //set first to init formclasscontrol!
13797: end;
13798:
13799: Proc SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13800: begin
13801:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13802:    +KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13803:   TScreenSaverOptions', 'set of TScreenSaverOption
```

```
13804:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13805:    TScreenSaverPreviewEvent', 'Proc ( Sender : TObject; previewHwnd: HWND)
13806:    SIRegister_TScreenSaver(CL);
13807:   //Proc Register
13808:   Proc SetScreenSaverPassword
13809: end;
13810:
13811: Proc SIRegister_XCollection(CL: TPSPascalCompiler);
13812: begin
13813:    FindClass('TOBJECT'),'TXCollection
13814:    SIRegister_EFilerException(CL);
13815:    SIRegister_TXCollectionItem(CL);
13816:    //TXCollectionItemClass', 'class of TXCollectionItem
13817:    SIRegister_TXCollection(CL);
13818:   Proc RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13819:   Proc DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13820:   Proc RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13821:   Proc UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13822:   Func FindXCollectionItemClass( const className :Str) : TXCollectionItemClass
13823:   Func GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13824: end;
13825:
13826: Proc SIRegister_XOpenGL(CL: TPSPascalCompiler);
13827: begin
13828:    TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13829:   Proc xglMapTexCoordToNull
13830:   Proc xglMapTexCoordToMain
13831:   Proc xglMapTexCoordToSecond
13832:   Proc xglMapTexCoordToDual
13833:   Proc xglMapTexCoordToArbitrary( const units : array of Cardinal);
13834:   Proc xglMapTexCoordToArbitrary1( const bitWiseUnits :Card);
13835:   Proc xglMapTexCoordToArbitraryAdd( const bitWiseUnits :Card)
13836:   Proc xglBeginUpdate
13837:   Proc xglEndUpdate
13838:   Proc xglPushState
13839:   Proc xglPopState
13840:   Proc xglForbidSecondTextureUnit
13841:   Proc xglAllowSecondTextureUnit
13842:   Func xglGetBitWiseMapping :Card
13843: end;
13844:
13845: Proc SIRegister_VectorLists(CL: TPSPascalCompiler);
13846: begin
13847:    TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13848:    TBaseListOptions', 'set of TBaseListOption
13849:    SIRegister_TBaseList(CL);
13850:    SIRegister_TBaseVectorList(CL);
13851:    SIRegister_TAffineVectorList(CL);
13852:    SIRegister_TVectorList(CL);
13853:    SIRegister_TTexPointList(CL);
13854:    SIRegister_TXIntList(CL);
13855:    //PSingleArrayList', '^TSingleArrayList // will not work
13856:    SIRegister_TSingleList(CL);
13857:    SIRegister_TByteList(CL);
13858:    SIRegister_TQuaternionList(CL);
13859:   Proc QuickSortLists(startIndex,endIndex:Int;refList:TSingleList; objList : TList);
13860:   Proc QuickSortLists1(startIndex,endIndex:Int;refList: TSingleList; objList: TBaseList);
13861:   Proc FastQuickSortLists(startIndex,endIndex:Int;refList:TSingleList;objList:TPersistentObjectList);
13862: end;
13863:
13864: ********** File C:\maXbox\maxbox3\maxbox3\maXbox3\source\REST\uPSI_UIntList.pas
13865:            File C:\maXbox\maxbox3\maxbox3\maXbox3\source\JvJCLUtils.pas
13866: { compile-time registration functions }
13867: 24: Proc SIRegister_TIntList(CL: TPSPascalCompiler);
13868: Proc SIRegister_UIntList(CL: TPSPascalCompiler);
13869: { run-time registration functions }
13870: 28: Proc RIRegister_TIntList(CL: TPSRuntimeClassImporter);
13871: Proc RIRegister_UIntList(CL: TPSRuntimeClassImporter);
13872: (*----------------------------------------------------------------------------*)
13873: 49: Proc SIRegister_TIntList(CL: TPSPascalCompiler);
13874: 51:    //with RegClassS(CL,'TPersistent', 'TIntList') do
13875:    with ClassN(CL.FindClass('TPersistent'),'TIntList') do begin
13876:    with ClassN(CL.FindClass('TList'),'TIntList') do begin
13877:     //with RegClassS(CL,'TPersistent', 'TIntList') do
13878:
13879:   TIntList = class(TPersistent)
13880:    private
13881:      FUpDateCount: Int;
13882:      FList: PIntItemList;
13883:      FCount: Int;
13884:      FCapacity: Int;
13885:      FSorted:Bool;
13886:      FDuplicates: TDuplicates;
13887:      FOnChange: TNotifyEvent;
13888:      FOnChanging: TNotifyEvent;
13889:      Proc ExchangeItems(Index1, Index2: Int);
13890:      Proc Grow;
13891:      Proc QuickSort(L, R: Int; SCompare: TIntListSortCompare);
```

```
13892:     Proc InsertItem(Index: Int; const S: int64);
13893:     Proc SetSorted(Value:Bool);
13894:   protected
13895:     Proc Error(const Msg:Str; Data: Int);
13896:     Proc Changed; virtual;
13897:     Proc Changing; virtual;
13898:     Func Get(Index: Int): int64;
13899:     Func GetCapacity: Int;
13900:     //Func GetCount: Int;
13901:     Func GetObject(Index: Int): TObject;
13902:     Proc Put(Index: Int; const S: int64);
13903:     Proc PutObject(Index: Int; AObject: TObject);
13904:     Proc SetCapacity(NewCapacity: Int);
13905:     Proc SetUpdateState(Updating:Bool);
13906:   public
13907:     Func GetCount: Int;
13908:     destructor Destroy; override;
13909:     Func Add(const S: int64): Int;
13910:     Func AddObject(const S: int64; AObject: TObject): Int; virtual;
13911:     Proc Clear;
13912:     Proc Delete(Index: Int);
13913:     Proc Exchange(Index1, Index2: Int);
13914:     Func Find(const S: int64; var Index: Int):Bool; virtual;
13915:     Func IndexOf(const S: int: Int;
13916:     Proc Insert(Index: Int; const S: int64);
13917:     Proc Sort; virtual;
13918:     Proc CustomSort(Compare: TIntListSortCompare); virtual;
13919:     Proc LoadFromFile(const FileName:Str); virtual;
13920:     Proc LoadFromStream(Stream:TStream); virtual;
13921:     Proc SaveToFile(const FileName:Str); virtual;
13922:     Proc SaveToStream(Stream: TStream);
13923:     property Duplicates: TDuplicates read FDuplicates write FDuplicates;
13924:     property Sorted:Bool read FSorted write SetSorted;
13925:     property OnChange: TNotifyEvent read FOnChange write FOnChange;
13926:     property OnChanging: TNotifyEvent read FOnChanging write FOnChanging;
13927:     property Ints [Index: Int]: int64 read Get write Put; default;
13928:     property Count: Int read GetCount;
13929:     property Objects[Index: Int]: TObject read GetObject write PutObject;
13930:   end;
13931:
13932: procedure SIRegister_TInteger(CL: TPSPascalCompiler);  //biginteger   bigint
13933: begin
13934:   //with RegClassS(CL,'TObject', 'TInteger') do
13935:   with CL.AddClassN(CL.FindClass('TObject'),'TInteger') do begin
13936:     Constructor Create1( const initialValue : int64)
13937:     Constructor Create
13938:     Procedure Free
13939:     RegisterProperty('Digits', 'TDigits', iptr);
13940:     Procedure Assign( const I2 : TInteger);');
13941:     Procedure Assign1( const I2 : int64);
13942:     Procedure Assign2( const I2 : string);
13943:     Procedure AbsoluteValue');
13944:     Procedure Add( const I2 : TInteger);
13945:     Procedure Add1( const I2 : int64);
13946:     Procedure AssignZero');
13947:     Procedure AssignOne');
13948:     Procedure Subtract( const I2 : TInteger);
13949:     Procedure Subtract1( const I2 : int64);
13950:     Procedure Mult( const I2 : TInteger);
13951:     Procedure Mult1( const I2 : int64);
13952:     Procedure FastMult( const I2 : TInteger);
13953:     Procedure Divide( const I2 : TInteger);
13954:     Procedure Divide1( const I2 : int64);
13955:     Procedure Modulo( const I2 : TInteger);
13956:     Procedure Modulo1( const N : int64);
13957:     Procedure ModPow( const I2, m : TInteger);
13958:     Procedure InvMod( I2 : TInteger);
13959:     Procedure DivideRem( const I2 : TInteger; var remain : TInteger);
13960:     Procedure DivideRemTrunc( const I2 : TInteger; var remain : TInteger);
13961:     Procedure DivideRemFloor( const I2 : TInteger; var remain : TInteger);
13962:     Procedure DivideRemEuclidean( const I2 : TInteger; var remain : TInteger);
13963:     Function Compare( I2 : TInteger) : integer;
13964:     Function Compare1( I2 : int64) : integer;
13965:     Procedure Factorial');
13966:     Function ConvertToDecimalString( commas : boolean) : string');
13967:     Function ConvertToInt64( var N : int64) : boolean');
13968:     Function DigitCount : ');
13969:     Procedure SetSign( s : integintegerer);
13970:     Function GetSign : integer');
13971:     Function IsOdd : boolean');
13972:     Function IsPositive : boolean');
13973:     Function IsNegative : boolean');
13974:     Function IsProbablyPrime : boolean');
13975:     Function IsZero : boolean');
13976:     Procedure ChangeSign');
13977:     Procedure Pow( const exponent : int64);
13978:     Procedure Sqroot');
13979:     Procedure Square');
13980:     Procedure FastSquare');
```

```
13981:     Procedure Gcd( const I2 : TInteger);
13982:     Procedure Gcd1( const I2 : int64);
13983:     Procedure NRoot( const Root : int64);
13984:     Function GetBase : integer');
13985:     Function BitCount : integer');
13986:     Function ConvertToHexString : String');
13987:     Function AssignRandomPrime( BitLength:integer; seed:String; mustMatchBitLength:boolean):boolean');
13988:     Function AssignHex( HexStr : String) : boolean');
13989:     Procedure RandomOfSize( size : integer);
13990:     Procedure Random( maxint : TInteger);
13991:     Procedure Getnextprime');
13992:   end;
13993: end;
13994:
13995: procedure SIRegister_TBigFloat(CL: TPSPascalCompiler); //bigdecimal bigdec
13996: begin
13997:   //with RegClassS(CL,'TObject', 'TBigFloat') do
13998:   with CL.AddClassN(CL.FindClass('TObject'),'TBigFloat') do begin
13999:     decpart', 'TFloatInt', iptrw);
14000:     sigdigits', 'word', iptrw);
14001:     exponent', 'integer', iptrw);
14002:     Constructor Create;
14003:     Constructor Create1( const MaxSig : TMaxSig);
14004:     Procedure Assign( A : TBigFloat);
14005:     Procedure Assign3( A : TBigFloat; SigDig : word);
14006:     Procedure Assign4( A : TInteger);
14007:     Procedure Assign5( N : int64);
14008:     Procedure Assign6( N : int64; SigDig : integer);
14009:     Procedure Assign7( d : extended);
14010:     Procedure Assign8( S : string);
14011:     Procedure Assign9( S : string; SigDig : word);
14012:     Procedure AssignZero');
14013:     Procedure AssignOne');
14014:     Procedure Add( B : TBigFloat);
14015:     Procedure Add1( B : int64);
14016:     Procedure AbsAdd( B : TBigFloat);
14017:     Procedure Subtract( B : TBigFloat);
14018:     Procedure Subtract1( B : int64);
14019:     Procedure Mult( B : TBigFloat);
14020:     Procedure Mult1( B : TBigfloat; const MaxSig : TMaxSig);
14021:     Procedure Mult2( B : TInteger);
14022:     Procedure Mult3( B : int64);
14023:     Procedure MultRaw( B : TBigFloat);
14024:     Procedure Reciprocal( const MaxSig : TMaxSig);
14025:     Procedure Divide( B : TBigFloat; const MaxSig : TMaxSig);
14026:     Procedure Divide1( B : TInteger; const MaxSig : TMaxSig);
14027:     Procedure Divide2( B : int64; const MaxSig : TMaxSig);
14028:     Procedure Square( const MaxSig : TMaxSig);
14029:     Function Compare( B : TBigFloat) : integer');
14030:     Function IsZero : boolean');
14031:     Procedure MaxBigFloat( B : TBigFloat);
14032:     Procedure MinBigFloat( B : TBigFloat);
14033:     Procedure Sqrt;
14034:     Procedure Sqrt1( const MaxSig : TMaxSig);
14035:     Procedure NRoot( N : integer; const MaxSig : TMaxSig);
14036:     Procedure IntPower( intpower : integer; const MaxSig : TMaxSig);
14037:     Procedure Power( power : TBigfloat; const MaxSig : TMaxSig);
14038:     Procedure Log( const MaxSig : TMaxSig);
14039:     Procedure Log10( const MaxSig : TMaxSig);
14040:     Procedure Exp( const MaxSig : TMaxSig);
14041:     Procedure PiConst( const MaxSig : TMaxSig);
14042:     Procedure Log2Const( const MaxSig : TMaxSig);
14043:     Procedure RoundToPrec( const MaxSig : TMaxSig);
14044:     Procedure RoundToPrec1;
14045:     Procedure Trunc( const x : integer);
14046:     Procedure Floor( const x : integer);
14047:     Procedure Ceiling( const x : integer);
14048:     Procedure Round( const x : integer);
14049:     Procedure AbsoluteValue');
14050:     Procedure Negate');
14051:     Procedure SetSigDigits( const newsigdigits : integer);
14052:     Function ConvertToString( const View : TView) : string');
14053:     Function ToString( const View : TView) : string');
14054:     Function ConvertToExtended( var num : extended) : boolean');
14055:     Function ConvertToInt64( var N : int64) : boolean');
14056:     Function IntPart : int64');
14057:   end;
14058: end;
14059:
14060: Proc SIRegister_MeshUtils(CL: TPSPascalCompiler);
14061: begin
14062:  Proc ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
14063:  Proc ConvertStripToList1( const strip : TIntList; list : TIntList);
14064:  Proc ConvertStripToList2(const strip:TAffineVectorList;const indices:TIntList;list:TAffineVectorList);
14065:  Proc ConvertIndexedListToList(const data:TAffineVectlist;const indices:TIntList;list:TAffineVectorList);
14066:  Func BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList;const
       normals:TAffineVectorList;const texCoords:TAffineVectorList):TIntList
14067:  Proc RemapReferences( reference : TAffineVectorList; const indices : TIntList);
14068:  Proc RemapReferences1( reference : TIntList; const indices : TIntList);
```

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

```
14155:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
14156:   TJclBorRADToolPath', 'string
14157:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
14158:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
14159:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
14160:  BorRADToolRepositoryPagesSection','String 'Repository Pages
14161:  BorRADToolRepositoryDialogsPage','String 'Dialogs
14162:  BorRADToolRepositoryFormsPage','String 'Forms
14163:  BorRADToolRepositoryProjectsPage','String 'Projects
14164:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
14165:  BorRADToolRepositoryObjectType','String 'Type
14166:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
14167:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
14168:  BorRADToolRepositoryObjectName','String 'Name
14169:  BorRADToolRepositoryObjectPage','String 'Page
14170:  BorRADToolRepositoryObjectIcon','String 'Icon
14171:  BorRADToolRepositoryObjectDescr','String 'Description
14172:  BorRADToolRepositoryObjectAuthor','String 'Author
14173:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
14174:  BorRADToolRepositoryObjectDesigner','String 'Designer
14175:  BorRADToolRepositoryDesignerDfm','String 'dfm
14176:  BorRADToolRepositoryDesignerXfm','String 'xfm
14177:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
14178:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
14179:  SourceExtensionDelphiPackage','String '.dpk
14180:  SourceExtensionBCBPackage','String '.bpk
14181:  SourceExtensionDelphiProject','String '.dpr
14182:  SourceExtensionBCBProject','String '.bpr
14183:  SourceExtensionBDSProject','String '.bdsproj
14184:  SourceExtensionDProject','String '.dproj
14185:  BinaryExtensionPackage','String '.bpl
14186:  BinaryExtensionLibrary','String '.dll
14187:  BinaryExtensionExecutable','String '.exe
14188:  CompilerExtensionDCP','String '.dcp
14189:  CompilerExtensionBPI','String '.bpi
14190:  CompilerExtensionLIB','String '.lib
14191:  CompilerExtensionTDS','String '.tds
14192:  CompilerExtensionMAP','String '.map
14193:  CompilerExtensionDRC','String '.drc
14194:  CompilerExtensionDEF','String '.def
14195:  SourceExtensionCPP','String '.cpp
14196:  SourceExtensionH','String '.h
14197:  SourceExtensionPAS','String '.pas
14198:  SourceExtensionDFM','String '.dfm
14199:  SourceExtensionXFM','String '.xfm
14200:  SourceDescriptionPAS','String 'Pascal source file
14201:  SourceDescriptionCPP','String 'C++ source file
14202:  DesignerVCL','String 'VCL
14203:  DesignerCLX','String 'CLX
14204:  ProjectTypePackage','String 'package
14205:  ProjectTypeLibrary','String 'library
14206:  ProjectTypeProgram','String 'program
14207:  Personality32Bit','String '32 bit
14208:  Personality64Bit','String '64 bit
14209:  PersonalityDelphi','String 'Delphi
14210:  PersonalityDelphiDotNet','String 'Delphi.net
14211:  PersonalityBCB','String 'C++Builder
14212:  PersonalityCSB','String 'C#Builder
14213:  PersonalityVB','String 'Visual Basic
14214:  PersonalityDesign','String 'Design
14215:  PersonalityUnknown','String 'Unknown personality
14216:  PersonalityBDS','String 'Borland Developer Studio
14217:  DOFDirectoriesSection','String 'Directories
14218:  DOFUnitOutputDirKey','String 'UnitOutputDir
14219:  DOFSearchPathName','String 'SearchPath
14220:  DOFConditionals','String 'Conditionals
14221:  DOFLinkerSection','String 'Linker
14222:  DOFPackagesKey','String 'Packages
14223:  DOFCompilerSection','String 'Compiler
14224:  DOFPackageNoLinkKey','String 'PackageNoLink
14225:  DOFAdditionalSection','String 'Additional
14226:  DOFOptionsKey','String 'Options
14227:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
14228:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
14229:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
14230:   TJclBorPersonalities', 'set of TJclBorPersonality
14231:   TJclBorDesigner', '( bdVCL, bdCLX )
14232:   TJclBorDesigners', 'set of TJCl BorDesigner
14233:   TJclBorPlatform', '( bp32bit, bp64bit )
14234:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
14235:   SIRegister_TJclBorRADToolInstallationObject(CL);
14236:   SIRegister_TJclBorlandOpenHelp(CL);
14237:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
14238:   TJclHelp2Objects', 'set of TJclHelp2Object
14239:   SIRegister_TJclHelp2Manager(CL);
14240:   SIRegister_TJclBorRADToolIdeTool(CL);
14241:   SIRegister_TJclBorRADToolIdePackages(CL);
14242:   SIRegister_IJclCommandLineTool(CL);
14243:   FindClass('TOBJECT'),'EJclCommandLineToolError
```

```
14244:   SIRegister_TJclCommandLineTool(CL);
14245:   SIRegister_TJclBorlandCommandLineTool(CL);
14246:   SIRegister_TJclBCC32(CL);
14247:   SIRegister_TJclDCC32(CL);
14248:   TJclDCC', 'TJclDCC32
14249:   SIRegister_TJclBpr2Mak(CL);
14250:   SIRegister_TJclBorlandMake(CL);
14251:   SIRegister_TJclBorRADToolPalette(CL);
14252:   SIRegister_TJclBorRADToolRepository(CL);
14253:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
14254:   TCommandLineTools', 'set of TCommandLineTool
14255:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
14256:   SIRegister_TJclBorRADToolInstallation(CL);
14257:   SIRegister_TJclBCBInstallation(CL);
14258:   SIRegister_TJclDelphiInstallation(CL);
14259:   SIRegister_TJclDCCIL(CL);
14260:   SIRegister_TJclBDSInstallation(CL);
14261:   TTraverseMethod', 'Func ( Installation: TJclBorRADToolInstallation) :Bool
14262:   SIRegister_TJclBorRADToolInstallations(CL);
14263:  Func BPLFileName( const BPLPath, PackageFileName :Str) :Str
14264:  Func BinaryFileName( const OutputPath, ProjectFileName :Str) :Str
14265:  Func IsDelphiPackage( const FileName :Str) :Bool
14266:  Func IsDelphiProject( const FileName :Str) :Bool
14267:  Func IsBCBPackage( const FileName :Str) :Bool
14268:  Func IsBCBProject( const FileName :Str) :Bool
14269:  Proc GetDPRFileInfo(const DPRFileName:str;out BinaryExtensio:str;const LibSuffx:PString);
14270:  Proc GetBPRFileInfo(const BPRFileName:str;out BinaryFileName:str;const Descript:PString);
14271:  Proc GetDPKFileInfo(const DPKFileName:str;out RunOnly:Bool;const LibSuffix:PString;const Descript:PString;
14272:  Proc GetBPKFileInfo(const BPKFileName:str;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
14273:  Func SamePath(const Path1, Path2:Str):Bool;
14274: end;
14275:
14276: Proc SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
14277: begin
14278:  'ERROR_NO_MORE_FILES','LongInt'( 18);
14279:  //Func stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Int
14280:  //Func fstat64( FileDes: Int;var StatBuffer : TStatBuf64) : Int
14281:  //Func lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Int
14282:  'LPathSeparator','String '/
14283:  'LDirDelimiter','String '/
14284:  'LDirSeparator','String ':
14285:  'JXPathDevicePrefix','String '\\.\
14286:  'JXPathSeparator','String '\
14287:  'JXDirDelimiter','String '\
14288:  'JXDirSeparator','String ';
14289:  'JXPathUncPrefix','String '\\
14290:  'faNormalFile','LongWord')( $00000080);
14291:  //'faUnixSpecific',' faSymLink);
14292:   JXTCompactPath', '( cpCenter, cpEnd )
14293:      _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
14294:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
14295:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
14296:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
14297:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
14298:  Func jxPathAddSeparator( const Path :Str) :Str
14299:  Func jxPathAddExtension( const Path, Extension :Str) :Str
14300:  Func jxPathAppend( const Path, Append :Str) :Str
14301:  Func jxPathBuildRoot( const Drive : Byte) :Str
14302:  Func jxPathCanonicalize( const Path :Str) :Str
14303:  Func jxPathCommonPrefix( const Path1, Path2 :Str) : Int
14304:  Func jxPathCompactPath(const DC:HDC;const Path:str;const Width:Int;CmpFmt:TCompactPath):str
14305:  Proc jxPathExtractElements(const Source:str;var Drive,Path,FileName,Ext:str)
14306:  Func jxPathExtractFileDirFixed( const S :Str) :Str
14307:  Func jxPathExtractFileNameNoExt( const Path :Str) :Str
14308:  Func jxPathExtractPathDepth( const Path :Str; Depth : Int) :Str
14309:  Func jxPathGetDepth( const Path :Str) : Int
14310:  Func jxPathGetLongName( const Path :Str) :Str
14311:  Func jxPathGetShortName( const Path :Str) :Str
14312:  Func jxPathGetLongName( const Path :Str) :Str
14313:  Func jxPathGetShortName( const Path :Str) :Str
14314:  Func jxPathGetRelativePath( Origin, Destination :Str) :Str
14315:  Func jxPathGetTempPath :Str
14316:  Func jxPathIsAbsolute( const Path :Str) :Bool
14317:  Func jxPathIsChild( const Path, Base :Str) :Bool
14318:  Func jxPathIsDiskDevice( const Path :Str) :Bool
14319:  Func jxPathIsUNC( const Path :Str) :Bool
14320:  Func jxPathRemoveSeparator( const Path :Str) :Str
14321:  Func jxPathRemoveExtension( const Path :Str) :Str
14322:  Func jxPathGetPhysicalPath( const LocalizedPath :Str) :Str
14323:  Func jxPathGetLocalizedPath( const PhysicalPath :Str) :Str
14324:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
14325:   JxTFileListOptions', 'set of TFileListOption
14326:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
14327:   TFileHandler', 'Proc ( const FileName :Str)
14328:   TFileHandlerEx', 'Proc ( const Directory :Str; const FileInfo : TSearchRec)
14329:  Func jxBuildFileList( const Path :Str; const Attr : Int; const List : TStrings) :Bool
14330:  Func jxAdvBuildFileList( const Path :Str; const Attr : Int; const Files : TStrings; const
       AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:str;const
       FileMatchFunc:TFileMatchFunc):Bool;
```

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

```
14418:   JxTFileFlags', 'set of TFileFlag
14419:   FindClass('TOBJECT'),'EJclFileVersionInfoError
14420:   SIRegister_TJclFileVersionInfo(CL);
14421:  Func jxOSIdentToString( const OSIdent : DWORD) :Str
14422:  Func jxOSFileTypeToString(const OSFileType: DWORD; const OSFileSubType : DWORD) :Str
14423:  Func jxVersionResourceAvailable( const FileName :Str) :Bool
14424:   TFileVersionFormat', '( vfMajorMinor, vfFull )
14425:  Func jxFormatVersionString( const HiV, LoV : Word) :Str;
14426:  Func jxFormatVersionString1( const Major, Minor, Build, Revision : Word) :Str;
14427:  //Func FormatVersionString2(const FixedInfo:TVSFixedFileInfo;VersionFrmat:TFileVersionFormat):str;
14428:  //Proc VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
14429:  //Proc VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
14430:  //Func VersionFixedFileInfo(const FileName:str;var FixedInfo:TVSFixedFileInfo):Boolean
14431:  Func jxVersionFixedFileInfoString(const FileName:str;VersionFormat:TFileVersionFormat;const
       NotAvailableText :str):str
14432:   SIRegister_TJclTempFileStream(CL);
14433:   FindClass('TOBJECT'),'TJclCustomFileMapping
14434:   SIRegister_TJclFileMappingView(CL);
14435:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
14436:   SIRegister_TJclCustomFileMapping(CL);
14437:   SIRegister_TJclFileMapping(CL);
14438:   SIRegister_TJclSwapFileMapping(CL);
14439:   SIRegister_TJclFileMappingStream(CL);
14440:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
14441:   //PPCharArray', '^TPCharArray // will not work
14442:   SIRegister_TJclMappedTextReader(CL);
14443:   SIRegister_TJclFileMaskComparator(CL);
14444:   FindClass('TOBJECT'),'EJclPathError
14445:   FindClass('TOBJECT'),'EJclFileUtilsError
14446:   FindClass('TOBJECT'),'EJclTempFileStreamError
14447:   FindClass('TOBJECT'),'EJclTempFileStreamError
14448:   FindClass('TOBJECT'),'EJclFileMappingError
14449:   FindClass('TOBJECT'),'EJclFileMappingViewError
14450:  Func jxPathGetLongName2( const Path :Str) :Str
14451:  Func jxWin32DeleteFile( const FileName :Str; MoveToRecycleBin :Bool) :Bool
14452:  Func jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName :Str) :Bool
14453:  Func jxWin32BackupFile( const FileName :Str; Move :Bool) :Bool
14454:  Func jxWin32RestoreFile( const FileName :Str) :Bool
14455:  Func jxSamePath( const Path1, Path2 :Str) :Bool
14456:  Proc jxPathListAddItems( var List :Str; const Items :Str)
14457:  Proc jxPathListIncludeItems( var List :Str; const Items :Str)
14458:  Proc jxPathListDelItems( var List :Str; const Items :Str)
14459:  Proc jxPathListDelItem( var List :Str; const Index : Int)
14460:  Func jxPathListItemCount( const List :Str) : Int
14461:  Func jxPathListGetItem( const List :Str; const Index : Int) :Str
14462:  Proc jxPathListSetItem( var List :Str; const Index : Int; const Value :Str)
14463:  Func jxPathListItemIndex( const List, Item :Str) : Int
14464:  Func jxParamName(Idx:Int;const Separator:str;const AllowedPrefixChars:str;TrimName:Bool):str;
14465:  Func jxParamValue(Index:Int; const Separator :Str; TrimValue :Bool) :Str;
14466:  Func jxParamValue1(const SearchName:str; const Separator :Str; CaseSensitive :Bool;
14467:                                  const AllowedPrefixCharacters :Str; TrimValue :Bool) :Str;
14468:  Func jxParamPos(const SearchName:Str;const Separator:str;CaseSensitive:Boolean;const
       AllowedPrefixCharacters string): Int
14469: end;
14470:
14471: Proc SIRegister_FileUtil(CL: TPSPascalCompiler);
14472: begin
14473:  'UTF8FileHeader','String #$ef#$bb#$bf);
14474:  Func lCompareFilenames( const Filename1, Filename2 :Str) : Int
14475:  Func lCompareFilenamesIgnoreCase( const Filename1, Filename2 :Str) : Int
14476:  Func lCompareFilenames(const Filename1,Filename2 :Str; ResolveLinks :Bool): Int
14477:  Func lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLiks:bool):int;
14478:  Func lFilenameIsAbsolute( const TheFilename :Str) :Bool
14479:  Func lFilenameIsWinAbsolute( const TheFilename :Str) :Bool
14480:  Func lFilenameIsUnixAbsolute( const TheFilename :Str) :Bool
14481:  Proc lCheckIfFileIsExecutable( const AFilename :Str)
14482:  Proc lCheckIfFileIsSymlink( const AFilename :Str)
14483:  Func lFileIsReadable( const AFilename :Str) :Bool
14484:  Func lFileIsWritable( const AFilename :Str) :Bool
14485:  Func lFileIsText( const AFilename :Str) :Bool
14486:  Func lFileIsText( const AFilename :Str; out FileReadable :Bool) :Bool
14487:  Func lFileIsExecutable( const AFilename :Str) :Bool
14488:  Func lFileIsSymlink( const AFilename :Str) :Bool
14489:  Func lFileIsHardLink( const AFilename :Str) :Bool
14490:  Func lFileSize( const Filename :Str) : int64;
14491:  Func lGetFileDescription( const AFilename :Str) :Str
14492:  Func lReadAllLinks( const Filename :Str; ExceptionOnError :Bool) :Str
14493:  Func lTryReadAllLinks( const Filename :Str) :Str
14494:  Func lDirPathExists( const FileName :Str) :Bool
14495:  Func lForceDirectory( DirectoryName :Str) :Bool
14496:  Func lDeleteDirectory(const DirectoryName:Str; OnlyChildren:Bool) :Bool
14497:  Func lProgramDirectory :Str
14498:  Func lDirectoryIsWritable( const DirectoryName :Str) :Bool
14499:  Func lExtractFileNameOnly( const AFilename :Str) :Str
14500:  Func lExtractFileNameWithoutExt( const AFilename :Str) :Str
14501:  Func lCompareFileExt( const Filename, Ext :Str; CaseSensitive :Bool) : Int;
14502:  Func lCompareFileExt( const Filename, Ext :Str) : Int;
14503:  Func lFilenameIsPascalUnit( const Filename :Str) :Bool
14504:  Func lAppendPathDelim( const Path :Str) :Str
```

```
14505:  Func lChompPathDelim( const Path :Str) :Str
14506:  Func lTrimFilename( const AFilename :Str) :Str
14507:  Func lCleanAndExpandFilename( const Filename :Str) :Str
14508:  Func lCleanAndExpandDirectory( const Filename :Str) :Str
14509:  Func lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory :Str) :Str
14510:  Func lCreateRelativePath(const Filename,BaseDirectory
        string;UsePointDirectory:boolean;AlwaysRequireSharedBaseFolder:Bool):str
14511:  Func lCreateAbsolutePath( const Filename, BaseDirectory :Str) :Str
14512:  Func lFileIsInPath( const Filename, Path :Str) :Bool
14513:  Func lFileIsInDirectory( const Filename, Directory :Str) :Bool
14514:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
14515:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag AllDirectoryEntriesMask','String '*
14516:  Func lGetAllFilesMask :Str
14517:  Func lGetExeExt :Str
14518:  Func lSearchFileInPath(const Filename,BasePath,SearchPath,Delimiter:Str;Flags:TSearchFileInPathFlags):str
14519:  Func lSearchAllFilesInPath(const Filename,BasePath,SearchPath,Delimiter:str;Flags:
        TSearchFileInPathFlags):TStrings
14520:  Func lFindDiskFilename( const Filename :Str) :Str
14521:  Func lFindDiskFileCaseInsensitive( const Filename :Str) :Str
14522:  Func lFindDefaultExecutablePath( const Executable :Str; const BaseDir:Str):str
14523:  Func lGetDarwinSystemFilename( Filename :Str) :Str
14524:   SIRegister_TFileIterator(CL);
14525:   TFileFoundEvent', 'Proc ( FileIterator : TFileIterator)
14526:   TDirectoryFoundEvent', 'Proc ( FileIterator : TFileIterator)
14527:   TDirectoryEnterEvent', 'Proc ( FileIterator : TFileIterator)
14528:   SIRegister_TFileSearcher(CL);
14529:  Func lFindAllFiles(const SearchPath:str;SearchMsk:str;SearchSubDirs:Bool):TStringList
14530:  Func lFindAllDirectories(const SearchPath:Str; SearchSubDirs:Bool): TStringList
14531:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
14532:  // TCopyFileFlags', 'set of TCopyFileFlag
14533:  Func lCopyFile(const SrcFilename,DestFilename:Str; Flags : TCopyFileFlags):Bool
14534:  Func lCopyFile(const SrcFilename,DestFilename:Str; PreserveTime :Bool):Bool
14535:  Func lCopyDirTree( const SourceDir, TargetDir :Str; Flags : TCopyFileFlags):Bool
14536:  Func lReadFileToString( const Filename :Str) :Str
14537:  Func lGetTempFilename( const Directory, Prefix :Str) :Str
14538:  {Func NeedRTLAnsi :Bool
14539:  Proc SetNeedRTLAnsi( NewValue :Bool)
14540:  Func UTF8ToSys( const s :Str) :Str
14541:  Func SysToUTF8( const s :Str) :Str
14542:  Func ConsoleToUTF8( const s :Str) :Str
14543:  Func UTF8ToConsole( const s :Str) :Str}
14544:  Func FileExistsUTF8( const Filename :Str) :Bool
14545:  Func FileAgeUTF8( const FileName :Str) : Longint
14546:  Func DirectoryExistsUTF8( const Directory :Str) :Bool
14547:  Func ExpandFileNameUTF8( const FileName :Str) :Str
14548:  Func ExpandUNCFileNameUTF8( const FileName :Str) :Str
14549:  Func ExtractShortPathNameUTF8( const FileName :Str) :Str
14550:  Func FindFirstUTF8(const Path:Str; Attr : Longint; out Rslt : TSearchRec) : Longint
14551:  Func FindNextUTF8( var Rslt : TSearchRec) : Longint
14552:  Proc FindCloseUTF8( var F : TSearchrec)
14553:  Func FileSetDateUTF8( const FileName :Str; Age : Longint) : Longint
14554:  Func FileGetAttrUTF8( const FileName :Str) : Longint
14555:  Func FileSetAttrUTF8( const Filename :Str; Attr : longint) : Longint
14556:  Func DeleteFileUTF8( const FileName :Str) :Bool
14557:  Func RenameFileUTF8( const OldName, NewName :Str) :Bool
14558:  Func FileSearchUTF8(const Name, DirList :Str; ImplicitCurrentDir :Bool) :Str
14559:  Func FileIsReadOnlyUTF8( const FileName :Str) :Bool
14560:  Func GetCurrentDirUTF8 :Str
14561:  Func SetCurrentDirUTF8( const NewDir :Str) :Bool
14562:  Func CreateDirUTF8( const NewDir :Str) :Bool
14563:  Func RemoveDirUTF8( const Dir :Str) :Bool
14564:  Func ForceDirectoriesUTF8( const Dir :Str) :Bool
14565:  Func FileOpenUTF8( const FileName :Str; Mode : Int) : THandle
14566:  Func FileCreateUTF8( const FileName :Str) : THandle;
14567:  Func FileCreateUTF81( const FileName :Str; Rights :Card) : THandle;
14568:  Func ParamStrUTF8( Param : Int) :Str
14569:  Func GetEnvironmentStringUTF8( Index : Int) :Str
14570:  Func GetEnvironmentVariableUTF8( const EnvVar :Str) :Str
14571:  Func GetAppConfigDirUTF8( Global :Bool; Create :Bool) :Str
14572:  Func GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir :Bool) :Str
14573:  Func SysErrorMessageUTF8( ErrorCode : Int) :Str
14574: end;
14575:
14576: Proc SIRegister_Keyboard(CL: TPSPascalCompiler);
14577: begin
14578:   //VK_F23 = 134;
14579:   //{$EXTERNALSYM VK_F24}
14580:   //VK_F24 = 135;
14581:  TVirtualKeyCode', 'Int
14582:  'VK_MOUSEWHEELUP','Int'(134);
14583:  'VK_MOUSEWHEELDOWN','Int'(135);
14584:  Func glIsKeyDown( c : Char) :Bool;
14585:  Func glIsKeyDown1( vk : TVirtualKeyCode) :Bool;
14586:  Func glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
14587:  Func glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) :Str
14588:  Func glKeyNameToVirtualKeyCode( const keyName :Str) : TVirtualKeyCode
14589:  Func glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
14590:  Proc glKeyboardNotifyWheelMoved( wheelDelta : Int)
14591: end;
```

```
14592:
14593: Proc SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14594: begin
14595:   TGLPoint', 'TPoint
14596:   //PGLPoint', '^TGLPoint // will not work
14597:   TGLRect', 'TRect
14598:   //PGLRect', '^TGLRect // will not work
14599:   TDelphiColor', 'TColor
14600:   TGLPicture', 'TPicture
14601:   TGLGraphic', 'TGraphic
14602:   TGLBitmap', 'TBitmap
14603:   //TGraphicClass', 'class of TGraphic
14604:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14605:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14606:   TGLMouseEvent', 'Proc ( Sender : TObject; Button : TGLMouse'
14607:    +'Button; Shift : TShiftState; X, Y : Int)
14608:   TGLMouseMoveEvent', 'TMouseMoveEvent
14609:   TGLKeyEvent', 'TKeyEvent
14610:   TGLKeyPressEvent', 'TKeyPressEvent
14611:   EGLOSError', 'EWin32Error
14612:   EGLOSError', 'EWin32Error
14613:   EGLOSError', 'EOSError
14614:  'glsAllFilter','string'All // sAllFilter
14615:  Func GLPoint( const x, y : Int) : TGLPoint
14616:  Func GLRGB( const r, g, b : Byte) : TColor
14617:  Func GLColorToRGB( color : TColor) : TColor
14618:  Func GLGetRValue( rgb : DWORD) : Byte
14619:  Func GLGetGValue( rgb : DWORD) : Byte
14620:  Func GLGetBValue( rgb : DWORD) : Byte
14621:  Proc GLInitWinColors
14622:  Func GLRect( const aLeft, aTop, aRight, aBottom : Int) : TGLRect
14623:  Proc GLInflateGLRect( var aRect : TGLRect; dx, dy : Int)
14624:  Proc GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14625:  Proc GLInformationDlg( const msg :Str)
14626:  Func GLQuestionDlg( const msg :Str) :Bool
14627:  Func GLInputDlg( const aCaption, aPrompt, aDefault :Str) :Str
14628:  Func GLSavePictureDialog( var aFileName :Str; const aTitle :Str) :Bool
14629:  Func GLOpenPictureDialog( var aFileName :Str; const aTitle :Str) :Bool
14630:  Func GLApplicationTerminated :Bool
14631:  Proc GLRaiseLastOSError
14632:  Proc GLFreeAndNil( var anObject: TObject)
14633:  Func GLGetDeviceLogicalPixelsX( device :Card) : Int
14634:  Func GLGetCurrentColorDepth : Int
14635:  Func GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Int
14636:  Func GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Int) : Pointer
14637:  Proc GLSleep( length :Card)
14638:  Proc GLQueryPerformanceCounter( var val : Int64)
14639:  Func GLQueryPerformanceFrequency( var val : Int64) :Bool
14640:  Func GLStartPrecisionTimer : Int64
14641:  Func GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14642:  Func GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14643:  Func GLRDTSC : Int64
14644:  Proc GLLoadBitmapFromInstance( ABitmap : TBitmap; AName :Str)
14645:  Func GLOKMessageBox( const Text, Caption :Str) : Int
14646:  Proc GLShowHTMLUrl( Url :Str)
14647:  Proc GLShowCursor( AShow :Bool)
14648:  Proc GLSetCursorPos( AScreenX, AScreenY : Int)
14649:  Proc GLGetCursorPos( var point : TGLPoint)
14650:  Func GLGetScreenWidth : Int
14651:  Func GLGetScreenHeight : Int
14652:  Func GLGetTickCount : int64
14653:  Func RemoveSpaces(const str :Str) :Str;
14654:   TNormalMapSpace','( nmsObject, nmsTangent )
14655:  Proc CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14656:  Proc SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14657:  Proc CalcTangentSpaceLightVectors(Light:TAffineVector;Vertices,Normals,Tangents,
       BiNormals:TAffineVectorList;Colors:TVectorList)
14658:  Func CreateObjectSpaceNormalMap(Width,Height:Int;HiNormals,HiTexCoords:TAffineVectorList):TGLBitmap
14659:  Func CreateTangentSpaceNormalMap(Width, Height:Int;HiNormals, HiTexCoords,LoNormals,LoTexCoords,Tangents,
14660:         BiNormals : TAffineVectorList) : TGLBitmap
14661: end;
14662:
14663: Proc SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14664: begin
14665:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14666:  // PGLStarRecord', '^TGLStarRecord // will not work
14667:  Func StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14668:  Func StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14669:  Func StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14670: end;
14671:
14672: Proc SIRegister_GeometryBB(CL: TPSPascalCompiler);
14673: begin
14674:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14675:   //PAABB', '^TAABB // will not work
14676:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14677:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14678:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14679:  Func AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
```

```
14680:  Proc AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14681:  Proc SetBB( var c : THmgBoundingBox; const v : TVector)
14682:  Proc SetAABB( var bb : TAABB; const v : TVector)
14683:  Proc BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14684:  Proc AABBTransform( var bb : TAABB; const m : TMatrix)
14685:  Proc AABBScale( var bb : TAABB; const v : TAffineVector)
14686:  Func BBMinX( const c : THmgBoundingBox) : Single
14687:  Func BBMaxX( const c : THmgBoundingBox) : Single
14688:  Func BBMinY( const c : THmgBoundingBox) : Single
14689:  Func BBMaxY( const c : THmgBoundingBox) : Single
14690:  Func BBMinZ( const c : THmgBoundingBox) : Single
14691:  Func BBMaxZ( const c : THmgBoundingBox) : Single
14692:  Proc AABBInclude( var bb : TAABB; const p : TAffineVector)
14693:  Proc AABBFromSweep(var SweepAABB: TAABB; const Start,Dest:TVector; const Radius:Single)
14694:  Func AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14695:  Func BBToAABB( const aBB : THmgBoundingBox) : TAABB
14696:  Func AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14697:  Func AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14698:  Proc OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14699:  Proc OffsetAABB1( var aabb : TAABB; const delta : TVector);
14700:  Func IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) :Bool;
14701:  Func IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) :Bool
14702:  Func IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) :Bool
14703:  Func IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) :Bool
14704:  Func AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) :Bool
14705:  Func PointInAABB( const p : TAffineVector; const aabb : TAABB) :Bool;
14706:  Func PointInAABB1( const p : TVector; const aabb : TAABB) :Bool;
14707:  Func PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) :Bool
14708:  Func TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) :Bool
14709:  Proc ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14710:  Proc AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14711:  Proc BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14712:  Func BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14713:  Func BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14714:  Func AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14715:  Func BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14716:  Func BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14717:  Func AABBContainsBSphere(const mainAABB:TAABB;const testBSphere:TBSphere): TSpaceContains
14718:  Func PlaneContainsBSphere(const Location,Normal:TAffineVector;const testBSphere:TBSphere):TSpaceContains
14719:  Func FrustumContainsBSphere(const Frustum:TFrustum;const testBSphere:TBSphere): TSpaceContains
14720:  Func FrustumContainsAABB( const Frustum:TFrustum;const testAABB:TAABB): TSpaceContains
14721:  Func ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14722:  Func BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) :Bool
14723:  Proc IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14724:  Func AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
      viewportSizeY:Int):TClipRect
14725: end;
14726:
14727: Proc SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14728: begin
14729:  Proc Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14730:  Proc Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14731:  Proc Cylindrical_Cartesian2(const r,theta,z1: single; var x,y,z  single; var ierr : Int);
14732:  Proc Cylindrical_Cartesian3(const r,theta,z1: double; var x,y,z: double; var ierr : Int);
14733:  Proc Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14734:  Proc Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14735:  Proc Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14736:  Proc Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14737:  Proc Spherical_Cartesian2(const r,theta, phi: single;var x,y,z:single;varierr:Int);
14738:  Proc Spherical_Cartesian3(const r,theta, phi:double;var x,y,z:double;var ierr:Int);
14739:  Proc Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14740:  Proc Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14741:  Proc Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14742:  Proc ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14743:  Proc ProlateSpheroidal_Cartesian1(const xi,eta,phi,a: double;var x,y, z : double);
14744:  Proc ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:singl;var x,y,z:single;var ierr:Int);
14745:  Proc ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:doubl;var x,y,z:double;var ierr:Int);
14746:  Proc OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14747:  Proc OblateSpheroidal_Cartesian1(const xi, eta, phi, a : double; var x, y, z : double);
14748:  Proc OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr:Int);
14749:  Proc OblateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr:Int);
14750:  Proc BipolarCylindrical_Cartesian(const u, v, z1, a: single; var x, y, z : single);
14751:  Proc BipolarCylindrical_Cartesian1(const u, v, z1, a: double; var x, y, z: double);
14752:  Proc BipolarCylindrical_Cartesian2(const u,v,z1,a:single;var x,y,z:single;var ierr:Int);
14753:  Proc BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double;var ierr:Int);
14754: end;
14755:
14756: Proc SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14757: begin
14758:  'EPSILON','Single').setExtended( 1e-40);
14759:  'EPSILON2','Single').setExtended( 1e-30);  }
14760:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14761:    +irection : TVector; viewPortRadius:Single;farClippingDistance:Single;frustum:TFrustum;end
14762:  THmgPlane', 'TVector
14763:   TDoubleHmgPlane', 'THomogeneousDblVector
14764:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14765:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14766:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14767:   TSingleArray', 'array of Single
```

```
14768:   TTransformations','array [0..15] of Single)
14769:   TPackedRotationMatrix','array [0..2] of Smallint)
14770:   TVertex', 'TAffineVector
14771:   //TVectorGL', 'THomogeneousFltVector
14772:   //TMatrixGL', 'THomogeneousFltMatrix
14773:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14774:  Func glTexPointMake( const s, t : Single) : TTexPoint
14775:  Func glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14776:  Func glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14777:  Proc glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14778:  Proc glSetVector( var v : TAffineVector; const x, y, z : Single);
14779:  Proc glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14780:  Proc glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14781:  Proc glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14782:  Proc glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14783:  Func glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14784:  Func glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14785:  Func glPointMake( const x, y, z : Single) : TVectorGL;
14786:  Func glPointMake1( const v : TAffineVector) : TVectorGL;
14787:  Func glPointMake2( const v : TVectorGL) : TVectorGL;
14788:  Proc glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14789:  Proc glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14790:  Proc glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14791:  Proc glMakePoint( var v : TVectorGL; const x, y, z : Single);
14792:  Proc glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14793:  Proc glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14794:  Proc glMakeVector( var v : TAffineVector; const x, y, z : Single);
14795:  Proc glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14796:  Proc glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14797:  Proc glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14798:  Proc glRstVector( var v : TAffineVector);
14799:  Proc glRstVector1( var v : TVectorGL);
14800:  Func glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14801:  Proc glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14802:  //Proc VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14803:  Func glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14804:  Proc glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14805:  Func glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14806:  Func glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14807:  Proc glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14808:  Proc glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14809:  Proc glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14810:  Proc glAddVector10( var v : TAffineVector; const f : Single);
14811:  Proc glAddVector11( var v : TVectorGL; const f : Single);
14812:  //Proc TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Int;dest:PTexPointArray);
14813:  //Proc TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Int;const scale:
      TTexPoint; dest : PTexPointArray);
14814:  //Proc VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Int;dest:
      PAffineVectorArray);
14815:  Func glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14816:  Proc glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14817:  Proc glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14818:  Proc glVectorSubtract3(const v1: TVectorGL; v2: TAffineVector; var result : TVectorGL);
14819:  Func glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14820:  Proc glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14821:  Proc glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14822:  Func glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14823:  Func glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14824:  Proc glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14825:  Proc glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14826:  Proc glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14827:  //Proc CombineVector1(var vr: TAffineVector; const v : TAffineVector; pf : PFloat);
14828:  Func glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14829:  Func glVectorCombine2(const V1,V2: TAffineVector; const F1, F2 : Single) : TAffineVector;
14830:  Func glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14831:  Proc glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14832:  Proc glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14833:  Proc glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14834:  Func glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14835:  Func glVectorCombine8(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single): TVectorGL;
14836:  Proc glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14837:  Proc glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14838:  Proc glVectorCombine11(const V1,V2 : TVectorGL; const F2 : Single; var vr: TVectorGL);
14839:  Func glVectorCombine3(const V1,V2,V3: TVectorGL; const F1, F2, F3 : Single): TVectorGL;
14840:  Proc glVectorCombine31(const V1, V2, V3:TVectorGL; const F1,F2, F3:Single; var vr : TVectorGL);
14841:  Func glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14842:  Func glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14843:  Func glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14844:  Func glPointProject( const p, origin, direction : TAffineVector) : Single;
14845:  Func glPointProject1( const p, origin, direction : TVectorGL) : Single;
14846:  Func glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14847:  Func glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14848:  Proc glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14849:  Proc glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14850:  Proc glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14851:  Proc glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14852:  Func glLerp( const start, stop, t : Single) : Single
14853:  Func glAngleLerp( start, stop, t : Single) : Single
14854:  Func glDistanceBetweenAngles( angle1, angle2 : Single) : Single
```

```
14855:  Func glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14856:  Func glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14857:  Proc glVectorLerp1(const v1,v2: TAffineVector; t : Single; var vr : TAffineVector);
14858:  Func glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14859:  Proc glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14860:  Func glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14861:  Func glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14862: //Proc VectorArrayLerp(const src1,src2:PVectorArray; t:Single; n:Int; dest:PVectorArray);
14863: //Proc VectorArrayLerp1(const src1,src2:PAffineVectorArray; t:Single;n:Int;dest: PAffineVectorArray);
14864:  Func glVectorLength( const x, y : Single) : Single;
14865:  Func glVectorLength1( const x, y, z : Single) : Single;
14866:  Func glVectorLength2( const v : TAffineVector) : Single;
14867:  Func glVectorLength3( const v : TVectorGL) : Single;
14868:  Func glVectorLength4( const v : array of Single) : Single;
14869:  Func glVectorNorm( const x, y : Single) : Single;
14870:  Func glVectorNorm1( const v : TAffineVector) : Single;
14871:  Func glVectorNorm2( const v : TVectorGL) : Single;
14872:  Func glVectorNorm3( var V : array of Single) : Single;
14873:  Proc glNormalizeVector( var v : TAffineVector);
14874:  Proc glNormalizeVector1( var v : TVectorGL);
14875:  Func glVectorNormalize( const v : TAffineVector) : TAffineVector;
14876:  Func glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14877: // Proc NormalizeVectorArray( list : PAffineVectorArray; n : Int);
14878:  Func glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14879:  Func glVectorNegate( const v : TAffineVector) : TAffineVector;
14880:  Func glVectorNegate1( const v : TVectorGL) : TVectorGL;
14881:  Proc glNegateVector( var V : TAffineVector);
14882:  Proc glNegateVector2( var V : TVectorGL);
14883:  Proc glNegateVector3( var V : array of Single);
14884:  Proc glScaleVector( var v : TAffineVector; factor : Single);
14885:  Proc glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14886:  Proc glScaleVector2( var v : TVectorGL; factor : Single);
14887:  Proc glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14888:  Func glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14889:  Proc glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14890:  Func glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14891:  Proc glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14892:  Proc glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14893:  Proc glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14894:  Func glVectorEquals( const V1, V2 : TVectorGL) :Bool;
14895:  Func glVectorEquals1( const V1, V2 : TAffineVector) :Bool;
14896:  Func glAffineVectorEquals( const V1, V2 : TVectorGL) :Bool;
14897:  Func glVectorIsNull( const v : TVectorGL) :Bool;
14898:  Func glVectorIsNull1( const v : TAffineVector) :Bool;
14899:  Func glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14900:  Func glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14901:  Func glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14902:  Func glVectorDistance( const v1, v2 : TAffineVector) : Single;
14903:  Func glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14904:  Func glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14905:  Func glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14906:  Func glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14907:  Func glVectorReflect( const V, N : TAffineVector) : TAffineVector
14908:  Proc glRotateVector(var vector:TVectorGL;const axis: TAffineVector; angle : Single);
14909:  Proc glRotateVector1( var vector:TVectorGL;const axis : TVectorGL; angle : Single);
14910:  Proc glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14911:  Func glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14912:  Func glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14913:  Proc glVectorRotateAroundY1(const v:TAffineVector;alpha:Single; var vr : TAffineVector);
14914:  Func glVectorRotateAroundZ(const v : TAffineVector; alpha : Single) : TAffineVector;
14915:  Proc glAbsVector( var v : TVectorGL);
14916:  Proc glAbsVector1( var v : TAffineVector);
14917:  Func glVectorAbs( const v : TVectorGL) : TVectorGL;
14918:  Func glVectorAbs1( const v : TAffineVector) : TAffineVector;
14919:  Proc glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14920:  Proc glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14921:  Proc glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14922:  Proc glSetMatrixRow( var dest : TMatrixGL; rowNb : Int; const aRow : TVectorGL);
14923:  Func glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14924:  Func glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14925:  Func glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14926:  Func glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14927:  Func glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14928:  Func glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14929:  Func glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14930:  Func glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14931:  Func glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14932:  Func glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14933:  Func glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14934:  Func glCreateRotationMatrix(const anAxis:TAffineVector; angle: Single): TMatrixGL;
14935:  Func glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14936:  Func glCreateAffineRotationMatrix(const anAxis:TAffineVector;angle:Single):TAffineMatrix
14937:  Func glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14938:  Func glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14939:  Proc glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14940:  Func glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14941:  Func glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14942:  Func glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14943:  Func glVectorTransform3(const V : TAffineVector; const M : TAffineMatrix):TAffineVector;
```

```
14944:  Func glMatrixDeterminant( const M : TAffineMatrix) : Single;
14945:  Func glMatrixDeterminant1( const M : TMatrixGL) : Single;
14946:  Proc glAdjointMatrix( var M : TMatrixGL);
14947:  Proc glAdjointMatrix1( var M : TAffineMatrix);
14948:  Proc glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14949:  Proc glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14950:  Proc glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14951:  Proc glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14952:  Proc glNormalizeMatrix( var M : TMatrixGL)
14953:  Proc glTransposeMatrix( var M : TAffineMatrix);
14954:  Proc glTransposeMatrix1( var M : TMatrixGL);
14955:  Proc glInvertMatrix( var M : TMatrixGL);
14956:  Proc glInvertMatrix1( var M : TAffineMatrix);
14957:  Func glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14958:  Func glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) :Bool
14959:  Func glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14960:  Func glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14961:  Func glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14962:  Func glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14963:  Proc glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14964:  Proc glNormalizePlane( var plane : THmgPlane)
14965:  Func glPlaneEvaluatePoint(const plane:THmgPlane;const point: TAffineVector): Single;
14966:  Func glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14967:  Func glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14968:  Proc glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14969:  Proc glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14970:  Func glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) :Bool;
14971:  Func glPointIsInHalfSpace1(const point,planePoint,planeNormal : TAffineVector) :Bool;
14972:  Func glPointPlaneDistance(const point, planePoint, planeNormal: TVectorGL) : Single;
14973:  Func glPointPlaneDistance1(const point, planePoint, planeNormal: TAffineVector): Single;
14974:  Func glPointSegmentClosestPoint(const point,segmentStart, segmentStop:TAffineVector):TAffineVector
14975:  Func glPointSegmentDistance(const point,segmentStart,segmentStop:TAffineVector): single
14976:  Func glPointLineClosestPoint(const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14977:  Func glPointLineDistance(const point,linePoint,lineDirection:TAffineVector): Single
14978:  Proc SglegmentSegmentClosestPoint(const S0Start,S0Stop,S1Start,S1Stop TAffineVector;
14979:                                        var Segment0Closest,Segment1Closest: TAffineVector)
14980:  Func glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14981:                                   TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14982:  Func glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14983:  Func glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14984:  Func glQuaternionMagnitude( const Q : TQuaternion) : Single
14985:  Proc glNormalizeQuaternion( var Q : TQuaternion)
14986:  Func glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14987:  Proc glQuaternionToPoints(const Q: TQuaternion; var ArcFrom,ArcTo : TAffineVector)
14988:  Func glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14989:  Func glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14990:  Func glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14991:  Func glQuaternionFromAngleAxis(const angle:Single;const axis:TAffineVector) : TQuaternion
14992:  Func glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14993:  Func glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14994:  Func glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14995:  Func glQuaternionSlerp(const QStart,QEnd:TQuaternion;Spin:Int; t: Single): TQuaternion;
14996:  Func glQuaternionSlerp1(const source, dest:TQuaternion; const t: Single) : TQuaternion;
14997:  Func glLnXP1( X : Extended) : Extended
14998:  Func glLog10( X : Extended) : Extended
14999:  Func glLog2( X : Extended) : Extended;
15000:  Func glLog21( X : Single) : Single;
15001:  Func glLogN( Base, X : Extended) : Extended
15002:  Func glIntPower( Base : Extended; Exponent : Int) : Extended
15003:  Func glPower( const Base, Exponent : Single) : Single;
15004:  Func glPower1( Base : Single; Exponent : Int) : Single;
15005:  Func glDegToRad( const Degrees : Extended) : Extended;
15006:  Func glDegToRad1( const Degrees : Single) : Single;
15007:  Func glRadToDeg( const Radians : Extended) : Extended;
15008:  Func glRadToDeg1( const Radians : Single) : Single;
15009:  Func glNormalizeAngle( angle : Single) : Single
15010:  Func glNormalizeDegAngle( angle : Single) : Single
15011:  Proc glSinCos( const Theta : Extended; var Sin, Cos : Extended);
15012:  Proc glSinCos11( const Theta : Double; var Sin, Cos : Double);
15013:  Proc glSinCos0( const Theta : Single; var Sin, Cos : Single);
15014:  Proc glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
15015:  Proc glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
15016:  Proc glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
15017:  Proc glPrepareSinCosCache(var s,c: array of Single; startAngle,stopAngle : Single)
15018:  Func glArcCos( const X : Extended) : Extended;
15019:  Func glArcCos1( const x : Single) : Single;
15020:  Func glArcSin( const X : Extended) : Extended;
15021:  Func glArcSin1( const X : Single) : Single;
15022:  Func glArcTan21( const Y, X : Extended) : Extended;
15023:  Func glArcTan21( const Y, X : Single) : Single;
15024:  Func glFastArcTan2( y, x : Single) : Single
15025:  Func glTan( const X : Extended) : Extended;
15026:  Func glTan1( const X : Single) : Single;
15027:  Func glCoTan( const X : Extended) : Extended;
15028:  Func glCoTan1( const X : Single) : Single;
15029:  Func glSinh( const x : Single) : Single;
15030:  Func glSinh1( const x : Double) : Double;
15031:  Func glCosh( const x : Single) : Single;
15032:  Func glCosh1( const x : Double) : Double;
```

```
Func glRSqrt( v : Single) : Single
Func glRLength( x, y : Single) : Single
Func glISqrt( i : Int) : Int
Func glILength( x, y : Int) : Int;
Func glILength1( x, y, z : Int) : Int;
Proc glRegisterBasedExp
Proc glRandomPointOnSphere( var p : TAffineVector)
Func glRoundInt( v : Single) : Single;
Func glRoundInt1( v : Extended) : Extended;
Func glTrunc( v : Single) : Int;
Func glTrunc64( v : Extended) : Int64;
Func glInt( v : Single) : Single;
Func glInt1( v : Extended) : Extended;
Func glFrac( v : Single) : Single;
Func glFrac1( v : Extended) : Extended;
Func glRound( v : Single) : Int;
Func glRound64( v : Single) : Int64;
Func glRound641( v : Extended) : Int64;
Func glTrunc( X : Extended) : Int64
Func glRound( X : Extended) : Int64
Func glFrac( X : Extended) : Extended
Func glCeil( v : Single) : Int;
Func glCeil64( v : Extended) : Int64;
Func glFloor( v : Single) : Int;
Func glFloor64( v : Extended) : Int64;
Func glScaleAndRound( i : Int; var s : Single) : Int
Func glSign( x : Single) : Int
Func glIsInRange( const x, a, b : Single) :Bool;
Func glIsInRange1( const x, a, b : Double) :Bool;
Func glIsInCube( const p, d : TAffineVector) :Bool;
Func glIsInCube1( const p, d : TVectorGL) :Bool;
//Func MinFloat( values : PSingleArray; nbItems : Int) : Single;
//Func MinFloat1( values : PDoubleArray; nbItems : Int) : Double;
//Func MinFloat2( values : PExtendedArray; nbItems : Int) : Extended;
Func glMinFloat3( const v1, v2 : Single) : Single;
Func glMinFloat4( const v : array of Single) : Single;
Func glMinFloat5( const v1, v2 : Double) : Double;
Func glMinFloat6( const v1, v2 : Extended) : Extended;
Func glMinFloat7( const v1, v2, v3 : Single) : Single;
Func glMinFloat8( const v1, v2, v3 : Double) : Double;
Func glMinFloat9( const v1, v2, v3 : Extended) : Extended;
//Func MaxFloat10( values : PSingleArray; nbItems : Int) : Single;
//Func MaxFloat( values : PDoubleArray; nbItems : Int) : Double;
//Func MaxFloat1( values : PExtendedArray; nbItems : Int) : Extended;
Func glMaxFloat2( const v : array of Single) : Single;
Func glMaxFloat3( const v1, v2 : Single) : Single;
Func glMaxFloat4( const v1, v2 : Double) : Double;
Func glMaxFloat5( const v1, v2 : Extended) : Extended;
Func glMaxFloat6( const v1, v2, v3 : Single) : Single;
Func glMaxFloat7( const v1, v2, v3 : Double) : Double;
Func glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
Func glMinInt9( const v1, v2 : Int) : Int;
Func glMinInt( const v1, v2 :Card) :Card;
Func glMaxInt( const v1, v2 : Int) : Int;
Func glMaxInt1( const v1, v2 :Card) :Card;
Func glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
//Func PolygonArea( const p : PAffineVectorArray; nSides : Int) : Single;
Func glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
//Func PolygonSignedArea( const p : PAffineVectorArray; nSides : Int) : Single;
//Proc ScaleFloatArray( values : PSingleArray; nb : Int; var factor : Single);
Proc glScaleFloatArray( var values : TSingleArray; factor : Single);
//Proc OffsetFloatArray( values : PSingleArray; nb : Int; var delta : Single);
Proc glOffsetFloatArray1( var values : array of Single; delta : Single);
//Proc OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Int);
Func glMaxXYZComponent( const v : TVectorGL) : Single;
Func glMaxXYZComponent1( const v : TAffineVector) : single;
Func glMinXYZComponent( const v : TVectorGL) : Single;
Func glMinXYZComponent1( const v : TAffineVector) : single;
Func glMaxAbsXYZComponent( v : TVectorGL) : Single
Func glMinAbsXYZComponent( v : TVectorGL) : Single
Proc glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
Proc glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
Proc glMinVector( var v : TVectorGL; const v1 : TVectorGL);
Proc glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
Proc glSortArrayAscending( var a : array of Extended)
Func glClampValue( const aValue, aMin, aMax : Single) : Single;
Func glClampValue1( const aValue, aMin : Single) : Single;
Func glGeometryOptimizationMode :Str
Proc glBeginFPUOnlySection
Proc glEndFPUOnlySection
Func glConvertRotation( const Angles : TAffineVector) : TVectorGL
Func glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
Func glMakeDblVector( var v : array of Double) : THomogeneousDblVector
Func glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
Func glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
Func glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
Func glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
Func glPointInPolygon( var xp, yp : array of Single; x, y : Single) :Bool
Proc glDivMod( Dividend : Int; Divisor : Word; var Result, Remainder : Word)
```

```
15122:  Func glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
15123:  Func glTurn1(const Matrix:TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
15124:  Func glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
15125:  Func glPitch1(const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
15126:  Func glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
15127:  Func glRoll1(const Matrix:TMatrixGL;const MasterDirection:TAffineVector;Angle:Single): TMatrixGL;
15128:  Func glRayCastMinDistToPoint(const rayStart,rayVector:TVectorGL;const point:TVectorGL):Single
15129:  Func glRayCastIntersectsSphere(const rayStart,rayVector:TVectorGL;const sphereCenter:TVectorGL;const
        sphereRadius:Single):Bool;
15130:  Func glRayCastSphereIntersect(const rayStart,rayVector:TVectorGL;const sphereCenter:TVectorGL;const
        sphereRadius:Single;var i1,i2:TVectorGL):Int;
15131:  Func glSphereVisibleRadius( distance, radius : Single) : Single
15132:  Func glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
15133:  Func glIsVolumeClipped(const objPos:TVectorGL;const objRadius:Single;const
        rcci:TRenderContextClippingInfo):Bool;
15134:  Func glIsVolumeClipped1(const objPos:TAffineVector;const objRadius:Single;const
        rcci:TRenderContextClippingInfo):Bool;
15135:  Func glIsVolumeClipped2(const min,max:TAffineVector;const rcci:TRenderContextClippingInfo):Bool;
15136:  Func glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const Frustum:TFrustum):Bool;
15137:  Func glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
15138:  Func glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
15139:  Func glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
15140:  Func glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
15141:  Func glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
15142:  'cPI','Single').setExtended( 3.141592654);
15143:  'cPIdiv180','Single').setExtended( 0.017453292);
15144:  'c180divPI','Single').setExtended( 57.29577951);
15145:  'c2PI','Single').setExtended( 6.283185307);
15146:  'cPIdiv2','Single').setExtended( 1.570796326);
15147:  'cPIdiv4','Single').setExtended( 0.785398163);
15148:  'c3PIdiv4','Single').setExtended( 2.35619449);
15149:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
15150:  'cInv360','Single').setExtended( 1 / 360);
15151:  'c180','Single').setExtended( 180);
15152:  'c360','Single').setExtended( 360);
15153:  'cOneHalf','Single').setExtended( 0.5);
15154:  'cLn10','Single').setExtended( 2.302585093);
15155:  {'MinSingle','Extended').setExtended( 1.5e-45);
15156:  'MaxSingle','Extended').setExtended( 3.4e+38);
15157:  'MinDouble','Extended').setExtended( 5.0e-324);
15158:  'MaxDouble','Extended').setExtended( 1.7e+308);
15159:  'MinExtended','Extended').setExtended( 3.4e-4932);
15160:  'MaxExtended','Extended').setExtended( 1.1e+4932);
15161:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
15162:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
15163: end;
15164:
15165: Proc SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
15166: begin
15167:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
15168:   (FindClass('TOBJECT'),'TFaceGroups
15169:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
15170:   TMeshAutoCenterings', 'set of TMeshAutoCentering
15171:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
15172:   SIRegister_TBaseMeshObject(CL);
15173:   (FindClass('TOBJECT'),'TSkeletonFrameList
15174:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
15175:   SIRegister_TSkeletonFrame(CL);
15176:   SIRegister_TSkeletonFrameList(CL);
15177:   (FindClass('TOBJECT'),'TSkeleton
15178:   (FindClass('TOBJECT'),'TSkeletonBone
15179:   SIRegister_TSkeletonBoneList(CL);
15180:   SIRegister_TSkeletonRootBoneList(CL);
15181:   SIRegister_TSkeletonBone(CL);
15182:   (FindClass('TOBJECT'),'TSkeletonColliderList
15183:   SIRegister_TSkeletonCollider(CL);
15184:   SIRegister_TSkeletonColliderList(CL);
15185:   (FindClass('TOBJECT'),'TGLBaseMesh
15186:   TBlendedLerpInfo', 'record frameIndex1 : Int; frameIndex2 : '
15187:    +'Int; lerpFactor : Single; weight : Single; externalPositions : TAffine'
15188:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
15189:    +'QuaternionList; end
15190:   SIRegister_TSkeleton(CL);
15191:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
15192:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
15193:   SIRegister_TMeshObject(CL);
15194:   SIRegister_TMeshObjectList(CL);
15195:   //TMeshObjectListClass', 'class of TMeshObjectList
15196:   (FindClass('TOBJECT'),'TMeshMorphTargetList
15197:   SIRegister_TMeshMorphTarget(CL);
15198:   SIRegister_TMeshMorphTargetList(CL);
15199:   SIRegister_TMorphableMeshObject(CL);
15200:   TVertexBoneWeight', 'record BoneID : Int; Weight : Single; end
15201:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
15202:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
15203:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
15204:   SIRegister_TSkeletonMeshObject(CL);
15205:   SIRegister_TFaceGroup(CL);
15206:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
```

```
15207:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
15208:   SIRegister_TFGVertexIndexList(CL);
15209:   SIRegister_TFGVertexNormalTexIndexList(CL);
15210:   SIRegister_TFGIndexTexCoordList(CL);
15211:   SIRegister_TFaceGroups(CL);
15212:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
15213:   SIRegister_TVectorFile(CL);
15214:   //TVectorFileClass', 'class of TVectorFile
15215:   SIRegister_TGLGLSMVectorFile(CL);
15216:   SIRegister_TGLBaseMesh(CL);
15217:   SIRegister_TGLFreeForm(CL);
15218:   TGLActorOption', '( aoSkeletonNormalizeNormals )
15219:   TGLActorOptions', 'set of TGLActorOption
15220:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
15221:   (FindClass('TOBJECT'),'TGLActor
15222:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
15223:   SIRegister_TActorAnimation(CL);
15224:   TActorAnimationName', 'String
15225:   SIRegister_TActorAnimations(CL);
15226:   SIRegister_TGLBaseAnimationControler(CL);
15227:   SIRegister_TGLAnimationControler(CL);
15228:   TActorFrameInterpolation', '( afpNone, afpLinear )
15229:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
15230:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
15231:   SIRegister_TGLActor(CL);
15232:   SIRegister_TVectorFileFormat(CL);
15233:   SIRegister_TVectorFileFormatsList(CL);
15234:   (FindClass('TOBJECT'),'EInvalidVectorFile
15235:  Func GetVectorFileFormats : TVectorFileFormatsList
15236:  Func VectorFileFormatsFilter :Str
15237:  Func VectorFileFormatsSaveFilter :Str
15238:  Func VectorFileFormatExtensionByIndex( index : Int) :Str
15239:  Proc RegisterVectorFileFormat(const aExtension,aDescription:str;aClass: TVectorFileClass)
15240:  Proc UnregisterVectorFileClass( aClass : TVectorFileClass)
15241: end;
15242:
15243: Proc SIRegister_AxCtrls(CL: TPSPascalCompiler);
15244: begin
15245:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
15246:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
15247:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
15248:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
15249:   SIRegister_TOleStream(CL);
15250:   (FindClass('TOBJECT'),'TConnectionPoints
15251:   TConnectionKind', '( ckSingle, ckMulti )
15252:   SIRegister_TConnectionPoint(CL);
15253:   SIRegister_TConnectionPoints(CL);
15254:   TDefinePropertyPage', 'Proc ( const GUID : TGUID)
15255:   (FindClass('TOBJECT'),'TActiveXControlFactory
15256:   SIRegister_TActiveXControl(CL);
15257:   //TActiveXControlClass', 'class of TActiveXControl
15258:   SIRegister_TActiveXControlFactory(CL);
15259:   SIRegister_TActiveFormControl(CL);
15260:   SIRegister_TActiveForm(CL);
15261:   //TActiveFormClass', 'class of TActiveForm
15262:   SIRegister_TActiveFormFactory(CL);
15263:   (FindClass('TOBJECT'),'TPropertyPageImpl
15264:   SIRegister_TPropertyPage(CL);
15265:   //TPropertyPageClass', 'class of TPropertyPage
15266:   SIRegister_TPropertyPageImpl(CL);
15267:   SIRegister_TActiveXPropertyPage(CL);
15268:   SIRegister_TActiveXPropertyPageFactory(CL);
15269:   SIRegister_TCustomAdapter(CL);
15270:   SIRegister_TAdapterNotifier(CL);
15271:   SIRegister_IFontAccess(CL);
15272:   SIRegister_TFontAdapter(CL);
15273:   SIRegister_IPictureAccess(CL);
15274:   SIRegister_TPictureAdapter(CL);
15275:   SIRegister_TOleGraphic(CL);
15276:   SIRegister_TStringsAdapter(CL);
15277:   SIRegister_TReflectorWindow(CL);
15278:  Proc EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
15279:  Proc GetOleFont( Font : TFont; var OleFont : IFontDisp)
15280:  Proc SetOleFont( Font : TFont; OleFont : IFontDisp)
15281:  Proc GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
15282:  Proc SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
15283:  Proc GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
15284:  Proc SetOleStrings( Strings : TStrings; OleStrings : IStrings)
15285:  Func ParkingWindow : HWND
15286: end;
15287:
15288: Proc SIRegister_synaip(CL: TPSPascalCompiler);
15289: begin
15290:  // TIp6Bytes = array [0..15] of Byte;
15291: {:binary form of IPv6 adress (for string conversion routines)}
15292:  // TIp6Words = array [0..7] of Word;
15293:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;;
15294:   AddTypeS('TIp6Words', 'array [0..7] of Word;;
15295:   AddTypeS(TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end;
```

```
15296:  Func synaIsIP( const Value :Str) :Bool;
15297:  Func synaIsIP6( const Value :Str) :Bool;
15298:  Func synaIPToID( Host :Str) : Ansistr;
15299:  Func synaStrToIp6( value :Str) : TIp6Bytes;
15300:  Func synaIp6ToStr( value : TIp6Bytes) :Str;
15301:  Func synaStrToIp( value :Str) : Int;
15302:  Func synaIpToStr( value : Int) :Str;
15303:  Func synaReverseIP( Value : Ansistr) : Ansistr;
15304:  Func synaReverseIP6( Value : Ansistr) : Ansistr;
15305:  Func synaExpandIP6( Value : Ansistr) : Ansistr;
15306:  Func xStrToIP( const Value :Str) : TIPAdr;
15307:  Func xIPToStr( const Adresse : TIPAdr) :Str;
15308:  Func IPToCardinal( const Adresse : TIPAdr) :Card;
15309:  Func CardinalToIP( const Value :Card) : TIPAdr;
15310:  Func ReverseIP( Value : Ansistr) : Ansistr');
15311:  Func ReverseIP6( Value : Ansistr) : Ansistr');
15312: end;
15313:
15314: Proc SIRegister_synacode(CL: TPSPascalCompiler);
15315: begin
15316:  AddTypeS('TSpecials', 'set of Char;
15317:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_;
15318:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+;
15319:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=;
15320:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=;
15321:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_;
15322:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;
15323:  Func DecodeTriplet( const Value : Ansistr; Delimiter : Char) : Ansistr;
15324:  Func DecodeQuotedPrintable( const Value : Ansistr) : Ansistr;
15325:  Func DecodeURL( const Value : Ansistr) : Ansistr;
15326:  Func EncodeTriplet(const Value:Ansistr;Delimiter:Char;Specials:TSpecials):Ansistr);
15327:  Func EncodeQuotedPrintable( const Value : Ansistr) : Ansistr;
15328:  Func EncodeSafeQuotedPrintable( const Value : Ansistr) : Ansistr;
15329:  Func EncodeURLElement( const Value : Ansistr) : Ansistr;
15330:  Func EncodeURL( const Value : Ansistr) : Ansistr;
15331:  Func Decode4to3( const Value, Table : Ansistr) : Ansistr;
15332:  Func Decode4to3Ex( const Value, Table : Ansistr) : Ansistr;
15333:  Func Encode3to4( const Value, Table : Ansistr) : Ansistr;
15334:  Func synDecodeBase64( const Value : Ansistr) : Ansistr;
15335:  Func synEncodeBase64( const Value : Ansistr) : Ansistr;
15336:  Func DecodeBase64mod( const Value : Ansistr) : Ansistr;
15337:  Func EncodeBase64mod( const Value : Ansistr) : Ansistr;
15338:  Func DecodeUU( const Value : Ansistr) : Ansistr;
15339:  Func EncodeUU( const Value : Ansistr) : Ansistr;
15340:  Func DecodeXX( const Value : Ansistr) : Ansistr;
15341:  Func DecodeYEnc( const Value : Ansistr) : Ansistr;
15342:  Func UpdateCrc32( Value : Byte; Crc32 : Int) : Int;
15343:  Func synCrc32( const Value : Ansistr) : Int;
15344:  Func UpdateCrc16( Value : Byte; Crc16 : Word) : Word;
15345:  Func Crc16( const Value : Ansistr) : Word;
15346:  Func synMD5( const Value : Ansistr) : Ansistr;
15347:  Func HMAC_MD5( Text, Key : Ansistr) : Ansistr;
15348:  Func MD5LongHash( const Value : Ansistr; Len : Int) : Ansistr;
15349:  Func synSHA1( const Value : Ansistr) : Ansistr;
15350:  Func HMAC_SHA1( Text, Key : Ansistr) : Ansistr;
15351:  Func SHA1LongHash( const Value : Ansistr; Len : Int) : Ansistr;
15352:  Func synMD4( const Value : Ansistr) : Ansistr;
15353: end;
15354:
15355: Proc SIRegister_synachar(CL: TPSPascalCompiler);
15356: begin
15357:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
15358:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
15359:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
15360:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
15361:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
15362:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
15363:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
15364:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
15365:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
15366:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
15367:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
15368:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
15369:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
15370:    +', CP864, CP865, CP869, CP1125 );
15371:   AddTypeS('TMimeSetChar', 'set of TMimeChar;
15372:  Func CharsetConversion(const Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiStr;
15373:  Func CharsetConversionEx(const Value:Ansistr;CharFrom:TMimeChar;CharTo:TMimeChar;const
      TransformTable:array of Word):Ansistr;
15374:  Func CharsetConversionTrans(Value:Ansistr;CharFrom:TMimeChar;CharTo:TMimeChar;const TransformTable:array
      of Word;Translit:Boolean):Ansistr;
15375:  Func GetCurCP : TMimeChar;
15376:  Func GetCurOEMCP : TMimeChar;
15377:  Func GetCPFromID( Value : Ansistr) : TMimeChar;
15378:  Func GetIDFromCP( Value : TMimeChar) : Ansistr;
15379:  Func NeedCharsetConversion( const Value : Ansistr) :Bool;
15380:  Func IdealCharsetCoding(const Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
15381:  Func GetBOM( Value : TMimeChar) : Ansistr;
15382:  Func StringToWide( const Value : Ansistr) : WideString;
```

```
15383:  Func WideToString( const Value : WideString) : Ansistr;
15384: end;
15385:
15386: Proc SIRegister_synamisc(CL: TPSPascalCompiler);
15387: begin
15388:   AddTypeS('TProxySetting', 'record Host :Str; Port :Str; Bypass :Str; end;
15389:  Proc WakeOnLan( MAC, IP :Str);
15390:  Func GetDNS :Str;
15391:  Func GetIEProxy( protocol :Str) : TProxySetting;
15392:  Func GetLocalIPs :Str;
15393: end;
15394:
15395: Proc SIRegister_synaser(CL: TPSPascalCompiler);
15396: begin
15397:  AddConstantN('synCR','Char #$0d);
15398:  Const('synLF','Char #$0a);
15399:  Const('cSerialChunk','LongInt'( 8192);
15400:  Const('LockfileDirectory','String '/var/lock;
15401:  Const('PortIsClosed','LongInt'( - 1);
15402:  Const('ErrAlreadyOwned','LongInt'( 9991);
15403:  Const('ErrAlreadyInUse','LongInt'( 9992);
15404:  Const('ErrWrongParameter','LongInt'( 9993);
15405:  Const('ErrPortNotOpen','LongInt'( 9994);
15406:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
15407:  Const('ErrMaxBuffer','LongInt'( 9996);
15408:  Const('ErrTimeout','LongInt'( 9997);
15409:  Const('ErrNotRead','LongInt'( 9998);
15410:  Const('ErrFrame','LongInt'( 9999);
15411:  Const('ErrOverrun','LongInt'( 10000);
15412:  Const('ErrRxOver','LongInt'( 10001);
15413:  Const('ErrRxParity','LongInt'( 10002);
15414:  Const('ErrTxFull','LongInt'( 10003);
15415:  Const('dcb_Binary','LongWord')( $00000001);
15416:  Const('dcb_ParityCheck','LongWord')( $00000002);
15417:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
15418:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
15419:  Const('dcb_DtrControlMask','LongWord')( $00000030);
15420:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
15421:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
15422:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
15423:  Const('dcb_DsrSensivity','LongWord')( $00000040);
15424:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
15425:  Const('dcb_OutX','LongWord')( $00000100);
15426:  Const('dcb_InX','LongWord')( $00000200);
15427:  Const('dcb_ErrorChar','LongWord')( $00000400);
15428:  Const('dcb_NullStrip','LongWord')( $00000800);
15429:  Const('dcb_RtsControlMask','LongWord')( $00003000);
15430:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
15431:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
15432:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
15433:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
15434:  Const('dcb_AbortOnError','LongWord')( $00004000);
15435:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
15436:  Const('synSB1','LongInt'( 0);
15437:  Const('SB1andHalf','LongInt'( 1);
15438:  Const('synSB2','LongInt'( 2);
15439:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
15440:  Const('CS7fix','LongWord')( $0000020);
15441:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
15442:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
15443:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
15444:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end;
15445:   //AddTypeS('PDCB', '^TDCB // will not work;
15446:  //Const('MaxRates','LongInt'( 18);
15447:  //Const('MaxRates','LongInt'( 30);
15448:  //Const('MaxRates','LongInt'( 19);
15449:  Const('O_SYNC','LongWord')( $0080);
15450:  Const('synOK','LongInt'( 0);
15451:  Const('synErr','LongInt'( Int ( - 1 ));
15452:   AddTypeS('THookSerialReason','(HR_SerialClose,HR_Connect,HR_CanRead,HR_CanWrite,HR_ReadCount,
       HR_WriteCount,HR_Wait);
15453:  Type('THookSerialStatus',Procedure(Sender:TObject;Reason:THookSerialReason;const Value:str);
15454:   SIRegister_ESynaSerError(CL);
15455:   SIRegister_TBlockSerial(CL);
15456:  Func GetSerialPortNames :Str;
15457: end;
15458:
15459: Proc SIRegister_synaicnv(CL: TPSPascalCompiler);
15460: begin
15461:  Const('DLLIconvName','String 'libiconv.so;
15462:  Const('DLLIconvName','String 'iconv.dll;
15463:   AddTypeS('size_t', 'Cardinal;
15464:   AddTypeS('iconv_t', 'Int;
15465:   //AddTypeS('iconv_t', 'Pointer;
15466:   AddTypeS('argptr', 'iconv_t;
15467:  Func SynaIconvOpen( const tocode, fromcode : Ansistr) : iconv_t;
15468:  Func SynaIconvOpenTranslit( const tocode, fromcode : Ansistr) : iconv_t;
15469:  Func SynaIconvOpenIgnore( const tocode, fromcode : Ansistr) : iconv_t;
15470:  Func SynaIconv( cd : iconv_t; inbuf : Ansistr; var outbuf : Ansistr) : Int;
```

```
15471:  Func SynaIconvClose( var cd : iconv_t) : Int;
15472:  Func SynaIconvCtl( cd : iconv_t; request : Int; argument : argptr) : Int;
15473:  Func IsIconvloaded :Bool;
15474:  Func InitIconvInterface :Bool;
15475:  Func DestroyIconvInterface :Bool;
15476:  Const('ICONV_TRIVIALP','LongInt'( 0);
15477:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
15478:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
15479:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
15480:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
15481: end;
15482:
15483: Proc SIRegister_pingsend(CL: TPSPascalCompiler);
15484: begin
15485:  Const 'ICMP_ECHO','LongInt'( 8);
15486:  Const('ICMP_ECHOREPLY','LongInt'( 0);
15487:  Const('ICMP_UNREACH','LongInt'( 3);
15488:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
15489:  Const('ICMP6_ECHO','LongInt'( 128);
15490:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
15491:  Const('ICMP6_UNREACH','LongInt'( 1);
15492:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
15493:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
15494:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort );
15495:   SIRegister_TPINGSend(CL);
15496:  Func PingHost( const Host :Str) : Int;
15497:  Func TraceRouteHost( const Host :Str) :Str;
15498: end;
15499:
15500: Proc SIRegister_asn1util(CL: TPSPascalCompiler);
15501: begin
15502:  AddConstantN('synASN1_BOOL','LongWord')( $01);
15503:  Const('synASN1_INT','LongWord')( $02);
15504:  Const('synASN1_OCTSTR','LongWord')( $04);
15505:  Const('synASN1_NULL','LongWord')( $05);
15506:  Const('synASN1_OBJID','LongWord')( $06);
15507:  Const('synASN1_ENUM','LongWord')( $0a);
15508:  Const('synASN1_SEQ','LongWord')( $30);
15509:  Const('synASN1_SETOF','LongWord')( $31);
15510:  Const('synASN1_IPADDR','LongWord')( $40);
15511:  Const('synASN1_COUNTER','LongWord')( $41);
15512:  Const('synASN1_GAUGE','LongWord')( $42);
15513:  Const('synASN1_TIMETICKS','LongWord')( $43);
15514:  Const('synASN1_OPAQUE','LongWord')( $44);
15515:  Func synASNEncOIDItem( Value : Int) : Ansistr;
15516:  Func synASNDecOIDItem( var Start : Int; const Buffer : Ansistr) : Int;
15517:  Func synASNEncLen( Len : Int) : Ansistr;
15518:  Func synASNDecLen( var Start : Int; const Buffer : Ansistr) : Int;
15519:  Func synASNEncInt( Value : Int) : Ansistr;
15520:  Func synASNEncUInt( Value : Int) : Ansistr;
15521:  Func synASNObject( const Data : Ansistr; ASNType : Int) : Ansistr;
15522:  Func synASNItem(var Start:Int;const Buffer:Ansistr;var ValueType:Int):Ansistr;
15523:  Func synMibToId( Mib :Str) : Ansistr;
15524:  Func synIdToMib( const Id : Ansistr) :Str;
15525:  Func synIntMibToStr( const Value : Ansistr) : Ansistr;
15526:  Func ASNdump( const Value : Ansistr) : Ansistr;
15527:  Func GetMailServers(const DNSHost, Domain : Ansistr; const Servers:TStrings):Bool;
15528:  Func LDAPResultDump( const Value : TLDAPResultList) : Ansistr;
15529: end;
15530:
15531: Proc SIRegister_ldapsend(CL: TPSPascalCompiler);
15532: begin
15533:  Const('cLDAPProtocol','String '389;
15534:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
15535:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
15536:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
15537:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
15538:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
15539:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
15540:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
15541:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
15542:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
15543:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
15544:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
15545:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
15546:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
15547:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
15548:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
15549:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
15550:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
15551:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
15552:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
15553:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
15554:   SIRegister_TLDAPAttribute(CL);
15555:   SIRegister_TLDAPAttributeList(CL);
15556:   SIRegister_TLDAPResult(CL);
15557:   SIRegister_TLDAPResultList(CL);
15558:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace );
15559:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree );
```

```
15560:   AddTypeS(TLDAPSearchAliases,(SA_NeverDeref,SA_InSearching,SA_FindingBaseObj,SA_Always);
15561:   SIRegister_TLDAPSend(CL);
15562:  Func LDAPResultDump( const Value : TLDAPResultList) : Ansistr;
15563: end;
15564:
15565:
15566: Proc SIRegister_slogsend(CL: TPSPascalCompiler);
15567: begin
15568:  Const('cSysLogProtocol','String '514;
15569:  Const('FCL_Kernel','LongInt'( 0);
15570:  Const('FCL_UserLevel','LongInt'( 1);
15571:  Const('FCL_MailSystem','LongInt'( 2);
15572:  Const('FCL_System','LongInt'( 3);
15573:  Const('FCL_Security','LongInt'( 4);
15574:  Const('FCL_Syslogd','LongInt'( 5);
15575:  Const('FCL_Printer','LongInt'( 6);
15576:  Const('FCL_News','LongInt'( 7);
15577:  Const('FCL_UUCP','LongInt'( 8);
15578:  Const('FCL_Clock','LongInt'( 9);
15579:  Const('FCL_Authorization','LongInt'( 10);
15580:  Const('FCL_FTP','LongInt'( 11);
15581:  Const('FCL_NTP','LongInt'( 12);
15582:  Const('FCL_LogAudit','LongInt'( 13);
15583:  Const('FCL_LogAlert','LongInt'( 14);
15584:  Const('FCL_Time','LongInt'( 15);
15585:  Const('FCL_Local0','LongInt'( 16);
15586:  Const('FCL_Local1','LongInt'( 17);
15587:  Const('FCL_Local2','LongInt'( 18);
15588:  Const('FCL_Local3','LongInt'( 19);
15589:  Const('FCL_Local4','LongInt'( 20);
15590:  Const('FCL_Local5','LongInt'( 21);
15591:  Const('FCL_Local6','LongInt'( 22);
15592:  Const('FCL_Local7','LongInt'( 23);
15593:   Type(TSyslogSeverity','(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15594:   SIRegister_TSyslogMessage(CL);
15595:   SIRegister_TSyslogSend(CL);
15596:  Func ToSysLog(const SyslogServer:str;Facil:Byte;Sever:TSyslogSeverity;const Content:str):Bool;
15597: end;
15598:
15599: Proc SIRegister_mimemess(CL: TPSPascalCompiler);
15600: begin
15601:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high );
15602:   SIRegister_TMessHeader(CL);
15603:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader;
15604:   SIRegister_TMimeMess(CL);
15605: end;
15606:
15607: Proc SIRegister_mimepart(CL: TPSPascalCompiler);
15608: begin
15609:   (FindClass('TOBJECT'),'TMimePart;
15610:   AddTypeS('THookWalkPart', 'Proc ( const Sender : TMimePart);
15611:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY );
15612:   AddTypeS(TMimeEncoding,(ME_7BIT,ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX );
15613:   SIRegister_TMimePart(CL);
15614:  Const('MaxMimeType','LongInt'( 25);
15615:  Func GenerateBoundary :Str;
15616: end;
15617:
15618: Proc SIRegister_mimeinln(CL: TPSPascalCompiler);
15619: begin
15620:  Func InlineDecode( const Value :Str; CP : TMimeChar) :Str;
15621:  Func InlineEncode( const Value :Str; CP, MimeP : TMimeChar) :Str;
15622:  Func NeedInline( const Value : Ansistr) :Bool;
15623:  Func InlineCodeEx( const Value :Str; FromCP : TMimeChar) :Str;
15624:  Func InlineCode( const Value :Str) :Str;
15625:  Func InlineEmailEx( const Value :Str; FromCP : TMimeChar) :Str;
15626:  Func InlineEmail( const Value :Str) :Str;
15627: end;
15628:
15629: Proc SIRegister_ftpsend(CL: TPSPascalCompiler);
15630: begin
15631:  Const('cFtpProtocol','String '21;
15632:  Const('cFtpDataProtocol','String '20;
15633:  Const('FTP_OK','LongInt'( 255);
15634:  Const('FTP_ERR','LongInt'( 254);
15635:   AddTypeS('TFTPStatus', 'Procedure(Sender:TObject; Response:Boolean; const Value :Str);
15636:   SIRegister_TFTPListRec(CL);
15637:   SIRegister_TFTPList(CL);
15638:   SIRegister_TFTPSend(CL);
15639:  Func FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass :Str) :Bool;
15640:  Func FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass :Str) :Bool;
15641:  Func FtpInterServerTransfer(const FromIP,FromPort,FromFile,FromUser,FromPass:str;const ToIP,ToPort,ToFile,
        ToUser,ToPass:str):Bool;
15642: end;
15643:
15644: Proc SIRegister_httpsend(CL: TPSPascalCompiler);
15645: begin
15646:  Const('cHttpProtocol','String '80;
15647:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED );
```

```
15648:   SIRegister_THTTPSend(CL);
15649:  Func HttpGetText( const URL :Str; const Response : TStrings) :Bool;
15650:  Func HttpGetBinary( const URL :Str; const Response : TStream) :Bool;
15651:  Func HttpPostBinary( const URL :Str; const Data : TStream) :Bool;
15652:  Func HttpPostURL( const URL, URLData :Str; const Data : TStream) :Bool;
15653:  Func HttpPostFile(const URL,FieldName,FileName:str;const Data:TStream;const ResultData:TStrings):Bool;
15654: end;
15655:
15656: Proc SIRegister_smtpsend(CL: TPSPascalCompiler);
15657: begin
15658:  Const('cSmtpProtocol','String '25;
15659:   SIRegister_TSMTPSend(CL);
15660:  Func SendToRaw(const MailFr,MailTo,SMTPHost:str;const MailData:TStrings;const Usrname,Passw:str):Bool;
15661:  Func SendTo(const MailFrom,MailTo,Subject,SMTPHost:str;const MailData:TStrings):Bool;
15662:  Func SendToEx(const MailFrom, MailTo, Subject, SMTPHost:str; const MailData:TStrings; const Username,
        Password :Str):Boolean;
15663: end;
15664:
15665: Proc SIRegister_snmpsend(CL: TPSPascalCompiler);
15666: begin
15667:  Const('cSnmpProtocol','String '161;
15668:  Const('cSnmpTrapProtocol','String '162;
15669:  Const('SNMP_V1','LongInt'( 0);
15670:  Const('SNMP_V2C','LongInt'( 1);
15671:  Const('SNMP_V3','LongInt'( 3);
15672:  Const('PDUGetRequest','LongWord')( $A0);
15673:  Const('PDUGetNextRequest','LongWord')( $A1);
15674:  Const('PDUGetResponse','LongWord')( $A2);
15675:  Const('PDUSetRequest','LongWord')( $A3);
15676:  Const('PDUTrap','LongWord')( $A4);
15677:  Const('PDUGetBulkRequest','LongWord')( $A5);
15678:  Const('PDUInformRequest','LongWord')( $A6);
15679:  Const('PDUTrapV2','LongWord')( $A7);
15680:  Const('PDUReport','LongWord')( $A8);
15681:  Const('ENoError',LongInt 0;
15682:  Const('ETooBig','LongInt')( 1);
15683:  Const('ENoSuchName','LongInt'( 2);
15684:  Const('EBadValue','LongInt'( 3);
15685:  Const('EReadOnly','LongInt'( 4);
15686:  Const('EGenErr','LongInt'( 5);
15687:  Const('ENoAccess','LongInt'( 6);
15688:  Const('EWrongType','LongInt'( 7);
15689:  Const('EWrongLength','LongInt'( 8);
15690:  Const('EWrongEncoding','LongInt'( 9);
15691:  Const('EWrongValue','LongInt'( 10);
15692:  Const('ENoCreation','LongInt'( 11);
15693:  Const('EInconsistentValue','LongInt'( 12);
15694:  Const('EResourceUnavailable','LongInt'( 13);
15695:  Const('ECommitFailed','LongInt'( 14);
15696:  Const('EUndoFailed','LongInt'( 15);
15697:  Const('EAuthorizationError','LongInt'( 16);
15698:  Const('ENotWritable','LongInt'( 17);
15699:  Const('EInconsistentName','LongInt'( 18);
15700:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv );
15701:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 );
15702:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES );
15703:   SIRegister_TSNMPMib(CL);
15704:   AddTypeS(TV3Sync', 'record EngineID : Ansistr; EngineBoots : Int; EngineTime : Int; EngineStamp :Card;
        end;
15705:   SIRegister_TSNMPRec(CL);
15706:   SIRegister_TSNMPSend(CL);
15707:  Func SNMPGet(const OID,Community,SNMPHost:Ansistr; var Value: Ansistr) :Bool;
15708:  Func SNMPSet(const OID,Community,SNMPHost,Value:Ansistr;ValueType:Int):Bool;
15709:  Func SNMPGetNext(var OID:Ansistr;const Community,SNMPHost:Ansistr;var Value:Ansistr):Bool;
15710:  Func SNMPGetTable(const BaseOID,Community,SNMPHost:Ansistr;const Value:TStrings):Bool;
15711:  Func SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15712:  Func SendTrap(const Dest,Source, Enterprise, Community : Ansistr; Generic, Specific, Seconds : Int; const
        MIBName, MIBValue : Ansistr; MIBtype : Int) : Int;
15713:  Func RecvTrap(var Dest,Source,Enterprise,Community:Ansistr;var Generic,Specific,Seconds Int; const
        MIBName,MIBValue : TStringList): Int;
15714: end;
15715:
15716: Proc SIRegister_NetWork(CL: TPSPascalCompiler);
15717: begin
15718:  Func GetDomainName2: Ansistr;
15719:  Func GetDomainController( Domain : Ansistr) : Ansistr;
15720:  Func GetDomainUsers( Controller : Ansistr) : Ansistr;
15721:  Func GetDomainGroups( Controller : Ansistr) : Ansistr;
15722:  Func GetDateTime( Controller : Ansistr) : TDateTime;
15723:  Func GetFullName2( Controller, UserID : Ansistr) : Ansistr;
15724: end;
15725:
15726: Proc SIRegister_wwSystem(CL: TPSPascalCompiler);
15727: begin
15728:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD );
15729:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15730:  Func wwStrToDate( const S :Str) :Bool;
15731:  Func wwStrToTime( const S :Str) :Bool;
15732:  Func wwStrToDateTime( const S :Str) :Bool;
```

```
15733:  Func wwStrToTimeVal( const S :Str) : TDateTime;
15734:  Func wwStrToDateVal( const S :Str) : TDateTime;
15735:  Func wwStrToDateTimeVal( const S :Str) : TDateTime;
15736:  Func wwStrToInt( const S :Str) :Bool;
15737:  Func wwStrToFloat( const S :Str) :Bool;
15738:  Func wwGetDateOrder( const DateFormat :Str) : TwwDateOrder;
15739:  Func wwNextDay( Year, Month, Day : Word) : Int;
15740:  Func wwPriorDay( Year, Month, Day : Word) : Int;
15741:  Func wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) :Bool;
15742:  Func wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) :Bool;
15743:  Func wwGetDateTimeCursorPosition(SelStart:int;Text:str;TimeOnly:Bool):TwwDateTimeSelection;
15744:  Func wwGetTimeCursorPosition( SelStart : Int; Text :Str) : TwwDateTimeSelection;
15745:  Func wwScanDate( const S :Str; var Date : TDateTime) :Bool;
15746:  Func wwScanDateEpoch(const S:Str; var Date : TDateTime; Epoch : Int):Bool;
15747:  Proc wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15748:  Func wwStrToFloat2(const S:Str; var FloatValue:Extended;DisplayFormat:str):bool;
15749: end;
15750:
15751: unit uPSI_Themes;
15752: Func ThemeServices : TThemeServices;
15753: Func ThemeControl( AControl : TControl) :Bool;
15754: Func UnthemedDesigner( AControl : TControl) :Bool;
15755: Proc SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15756: begin
15757:  Func GetBindingkeyAccessPoint(const Operator:Str; const key:Str):Str;
15758: end;
15759: Unit uPSC_menus;
15760:  Func StripHotkey( const Text :Str) :Str;
15761:  Func GetHotkey( const Text :Str) :Str;
15762:  Func AnsiSameCaption( const Text1, Text2 :Str) :Bool;
15763:  Func IsAltGRPressed :Bool;
15764:
15765: Proc SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15766: begin
15767:   TCommandEvent,Procedure(Thread:TIdPeerThread; const Tag,CmdStr:str;var Handled:Bool);
15768:   SIRegister_TIdIMAP4Server(CL);
15769: end;
15770:
15771: Proc SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15772: begin
15773:  'HASH_SIZE','LongInt'( 256);
15774:   CL.FindClass('TOBJECT'),'EVariantSymbolTable;
15775:   TypeS('TSymbolType', '( stInt, stFloat, stDate, stString );
15776:   //TypeS('PSymbol', '^TSymbol // will not work;
15777:   TypeS('TSymbol', 'record Name :Str; BlockLevel : Int; HashValue'
15778:    +' : Int; Value : Variant; end;
15779:   //TypeS('PSymbolArray', '^TSymbolArray // will not work;
15780:   SIRegister_TVariantSymbolTable(CL);
15781: end;
15782:
15783: Proc SIRegister_udf_glob(CL: TPSPascalCompiler);
15784: begin
15785:   SIRegister_TThreadLocalVariables(CL);
15786:  Func MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar;
15787:  //Func MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD;
15788:  Func ThreadLocals : TThreadLocalVariables;
15789:  Proc WriteDebug( sz :Str);
15790:  ConstantN('UDF_SUCCESS','LongInt'( 0);
15791:  'UDF_FAILURE','LongInt'( 1);
15792:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15793:  TypeS('mTByteArray', 'array of byte;;
15794:  Func ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15795:  Func ChangeOEPFromFile(sFile:str; sDestFile:str):Boolean;
15796:  Proc CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget:Str);
15797:  Func IsNetworkConnected:Bool;
15798:  Func IsInternetConnected:Bool;
15799:  Func IsCOMConnected:Bool;
15800:  Func IsNetworkOn:Bool;
15801:  Func IsInternetOn:Bool;
15802:  Func IsCOMOn:Bool;
15803:  Func SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15804:  TmrProc', 'Proc TmrProc(hWnd: HWND; uMsg: Int; idEvent: Int; dwTime: Int);;
15805:  Func SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT;
15806:  Func KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL;
15807:  Func wIsWindowUnicode( hWnd : HWND) : BOOL;
15808:  Func wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL;
15809:  Func wIsWindowEnabled( hWnd : HWND) : BOOL;
15810:  Func GetMenu( hWnd : HWND) : HMENU;
15811:  Func SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL;
15812: end;
15813:
15814: Proc SIRegister_SockTransport(CL: TPSPascalCompiler);
15815: begin
15816:   SIRegister_IDataBlock(CL);
15817:   SIRegister_ISendDataBlock(CL);
15818:   SIRegister_ITransport(CL);
15819:   SIRegister_TDataBlock(CL);
15820:   //TypeS('PIntArray', '^TIntArray // will not work;
15821:   //TypeS('PVariantArray', '^TVariantArray // will not work;
```

```
15822:   TypeS('TVarFlag', '( vfByRef, vfVariant );
15823:   TypeS('TVarFlags', 'set of TVarFlag;
15824:   SIRegister_TCustomDataBlockInterpreter(CL);
15825:   SIRegister_TSendDataBlock(CL);
15826:   'CallSig','LongWord')( $D800);
15827:   'ResultSig','LongWord')( $D400);
15828:   'asMask','LongWord')( $00FF);
15829:   ClassN(CL.FindClass('TOBJECT'),'EInterpreterError;
15830:   ClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError;
15831:  Proc CheckSignature( Sig : Int);
15832: end;
15833:
15834: Proc SIRegister_WinInet(CL: TPSPascalCompiler);
15835: begin
15836:   //TypeS('HINTERNET', '___Pointer;
15837:   TypeS('HINTERNET1', 'THANDLE;
15838:   TypeS('HINTERNET', 'Int;
15839:   TypeS('HINTERNET2', '___Pointer;
15840:   //TypeS('PHINTERNET', '^HINTERNET // will not work;
15841:   //TypeS('LPHINTERNET', 'PHINTERNET;
15842:   TypeS('INTERNET_PORT', 'Word;
15843:   //TypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work;
15844:   //TypeS('LPINTERNET_PORT', 'PINTERNET_PORT;
15845:   Func InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15846:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15847:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15848:  Func InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
15849:  Func InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
        dwUrlLength:DWORD):BOOL;
15850:  Func InternetCloseHandle(hInet : HINTERNET) : BOOL;
15851:  Func
        InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;lpszPassword:PChar;d
15852:  Func InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15853:  ;dwContext:DWORD):HINTERNET;
15854:  Func InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
        lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15855:  Func InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
        dwContext:DWORD):BOOL;
15856:  Func InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL;
15857:  Func
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15858:  Func InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD;
15859:  Func InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL;
15860:  Func InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL;
15861:  Func InternetAutodialHangup( dwReserved : DWORD) : BOOL;
15862:  Func InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL;
15863:  Func InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
        lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15864:  Func InternetCombineUrl(lpszBaseUrl,lpszRelativeUrl:PChar;lpszBuffer:PChar;var lpdwBufferLength:Func
        InternetCloseHandle(hInet:HINTERNET):BOOL;
15865:  Func InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15866:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15867:  Func InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
        dwContext:DWORD):BOOL;
15868:  Func FtpFindFirstFile(hConnect:HINTERNET;lpszSearchFile:PChar;var lpFindFileData:TWin32FindData;dwFlags:
        DWORD; dwContext:DWORD): HINTERNET;
15869:  Func
        WFtpGetFile(hConnect:HINTERNET;lpszRemoteFile:PChar;lpszNewFile:PChar;fFailIfExists:BOOL;dwFlagsAndAttributes:DWORD;dw
15870:  Func WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15871:  Func FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL;
15872:  Func FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15873:  Func FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15874:  Func FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL;
15875:  Func FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL;
15876:  Func FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL;
15877:  Func FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15878:  Func FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15879:  Func IS_GOPHER_FILE( GopherType : DWORD) : BOOL;
15880:  Func IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL;
15881:  Func IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL;
15882:  Func IS_GOPHER_ERROR( GopherType : DWORD) : BOOL;
15883:  Func IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL;
15884:  Func IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL;
15885:  Func IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL;
15886:  Func IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL;
15887:  Func IS_GOPHER_ASK( GopherType : DWORD) : BOOL;
15888:  Func IS_GOPHER_PLUS( GopherType : DWORD) : BOOL;
15889:  Func IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL;
15890:  Func
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorStr:PChar;dwGopherTyp
        lpdwBufferLength:DWORD):BOOL;
15891:  Func GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL;
15892:  Func
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15893:  Func
        HttpOpenRequest(hConnect:HINTERNET;lpszVerb:PChar;lpszObjectName:PChar;lpszVersion:PChar;lpszReferrer:
        PChar;lplpszAcceptTypes:PLPSTR;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15894:  Func HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15895:  Func HttpSendRequest(hRequest:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;lpOptional:Tobject;
        dwOptionalLength:DWORD):BOOL;
```

```
15896:  Func InternetGetCookie(lpszUrl,lpszCookName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15897:  Func InternetAttemptConnect( dwReserved : DWORD) : DWORD;
15898:  Func InternetCheckConnection(lpszUrl:PChar; dwFlags:DWORD; dwReserved:DWORD) : BOOL;
15899:  Func InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15900:  Func InternetConfirmZoneCrossing(hWnd:HWND;szUrlPrev,szUrlNew:LPSTR;bPost:BOOL): DWORD;
15901:  Func CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64;
15902:  Func DeleteUrlCacheGroup(GroupId: Int64; dwFlags : DWORD; lpReserved : TObject) : Bool;
15903:  Func FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15904:  Func FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo;var
        lpdwNextCacheEntryInfoBufferSize:DWORD):BOOL;
15905:  Func FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15906:  Func DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15907:  Func
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15908:  Func InternetSetDialState(lpszConnectoid:PChar; dwState:DWORD;dwReserved: DWORD): BOOL;
15909: end;
15910:
15911: Proc SIRegister_Wwstr(CL: TPSPascalCompiler);
15912: begin
15913:   AddTypeS('strCharSet', 'set of char;
15914:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15915:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption;
15916:  Proc strBreakApart( s :Str; delimeter :Str; parts : TStrings);
15917:  Func strGetToken( s :Str; delimeter :Str; var APos : Int) :Str;
15918:  Proc strStripPreceding( var s :Str; delimeter : strCharSet);
15919:  Proc strStripTrailing( var s :Str; delimeter : strCharSet);
15920:  Proc strStripWhiteSpace( var s :Str);
15921:  Func strRemoveChar( str :Str; removeChar : char) :Str;
15922:  Func wwstrReplaceChar( str :Str; removeChar, replaceChar : char) :Str;
15923:  Func strReplaceCharWithStr(str:str; removeChar:char;replaceStr:Str) :Str;
15924:  Func wwEqualStr( s1, s2 :Str) :Bool;
15925:  Func strCount( s :Str; delimeter : char) : Int;
15926:  Func strWhiteSpace : strCharSet;
15927:  Func wwExtractFileNameOnly( const FileName :Str) :Str;
15928:  Func wwGetWord(s:str;var APos:Int;Options:TwwGetWordOptions;DelimSet:strCharSet):str;
15929:  Func strTrailing( s :Str; delimeter : char) :Str;
15930:  Func strPreceding( s :Str; delimeter : char) :Str;
15931:  Func wwstrReplace( s, Find, Replace :Str) :Str;
15932: end;
15933:
15934:  Ex.:tex2:= 'Finally, we recognize that the 8% is calculated over all states of the world.'
15935:       delimset:= [',','?',';','.',':','-','=','{','}','[',']','(',')','>','<','%'];
15936:       while length(tex2)>0 do begin
15937:         k:=1;
15938:         s:= wwGetWord(tex2, k, [wwgwStripQuotes, wwgwSpacesInWords], delimSet);
15939:         delete(tex2,1,k);
15940:         write(' '+s)
15941:       end;
15942:
15943: Proc SIRegister_DataBkr(CL: TPSPascalCompiler);
15944: begin
15945:   SIRegister_TRemoteDataModule(CL);
15946:   SIRegister_TCRemoteDataModule(CL);
15947:  Proc RegisterPooled(const ClassID:str; Max,Timeout : Int; Singleton :Bool);
15948:  Proc UnregisterPooled( const ClassID :Str);
15949:  Proc EnableSocketTransport( const ClassID :Str);
15950:  Proc DisableSocketTransport( const ClassID :Str);
15951:  Proc EnableWebTransport( const ClassID :Str);
15952:  Proc DisableWebTransport( const ClassID :Str);
15953: end;
15954:
15955: Proc SIRegister_Mathbox(CL: TPSPascalCompiler);
15956: begin
15957:  Func mxArcCos( x : Real) : Real;
15958:  Func mxArcSin( x : Real) : Real;
15959:  Func Comp2Str( N : Comp) :Str;
15960:  Func Int2StrPad0( N : LongInt; Len : Int) :Str;
15961:  Func Int2Str( N : LongInt) :Str;
15962:  Func mxIsEqual( R1, R2 : Double) :Bool;
15963:  Func LogXY( x, y : Real) : Real;
15964:  Func Pennies2Dollars( C : Comp) :Str;
15965:  Func mxPower( X : Int; Y : Int) : Real;
15966:  Func Real2Str( N : Real; Width, Places : Int) :Str;
15967:  Func mxStr2Comp( MyString :Str) : Comp;
15968:  Func mxStr2Pennies( S :Str) : Comp;
15969:  Func Str2Real( MyString :Str) : Real;
15970:  Func XToTheY( x, y : Real) : Real;
15971: end;
15972:
15973: //*************************Cindy Functions!*****************************
15974: Proc SIRegister_cyIndy(CL: TPSPascalCompiler);
15975: begin
15976:  TypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15977:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15978:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification);
15979:  MessagePlainText','String 'text/plain;
15980:  ConstantN('MessagePlainText_Attach','String 'multipart/mixed;
15981:  MessageAlterText_Html','String 'multipart/alternative;
```

```
15982:  MessageHtml_Attach','String 'multipart/mixed;
15983:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html";
15984:  MessageAlterText_Html_Attach','String 'multipart/mixed;
15985:  MessageAlterText_Html_RelatedAttach,String)('multipart/related;type="multipart/alternative"';
15986:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed;
15987:  MessageReadNotificatin String).(multipart/report;report-type="disposition-notification"';
15988:  Func ForceDecodeHeader( aHeader :Str) :Str;
15989:  Func Base64_EncodeString( Value :Str; const aEncoding : TEncoding) :Str;
15990:  Func Base64_DecodeToString(Value:Str; const aBytesEncoding:TEncoding):Str;;
15991:  Func Base64_DecodeToBytes( Value :Str) : TBytes;;
15992:  Func IdHttp_DownloadFile(aSrcUrlFile,aDestFile:str;const OnWorkEvent:TWorkEvent):Bool;
15993:  Func Get_MD5( const aFileName :Str) :Str;
15994:  Func Get_MD5FromString( const aString :Str) :Str;
15995: end;
15996:
15997: Proc SIRegister_cySysUtils(CL: TPSPascalCompiler);
15998: begin
15999:  Func IsFolder( SRec : TSearchrec) :Bool;
16000:  Func isFolderReadOnly( Directory :Str) :Bool;
16001:  Func DirectoryIsEmpty( Directory :Str) :Bool;
16002:  Func DirectoryWithSubDir( Directory :Str) :Bool;
16003:  Proc GetSubDirs( FromDirectory :Str; aList : TStrings);
16004:  Func DiskFreeBytes( Drv : Char) : Int64;
16005:  Func DiskBytes( Drv : Char) : Int64;
16006:  Func GetFileBytes( Filename :Str) : Int64;
16007:  Func GetFilesBytes( Directory, Filter :Str) : Int64;
16008:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege;
16009:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege;
16010:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege;
16011:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege;
16012:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege;
16013:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege;
16014:  SE_TCB_NAME','String 'SeTcbPrivilege;
16015:  SE_SECURITY_NAME','String 'SeSecurityPrivilege;
16016:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege;
16017:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege;
16018:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege;
16019:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege;
16020:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege;
16021:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege;
16022:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege;
16023:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege;
16024:  SE_BACKUP_NAME','String 'SeBackupPrivilege;
16025:  SE_RESTORE_NAME','String 'SeRestorePrivilege;
16026:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege;
16027:  SE_DEBUG_NAME','String 'SeDebugPrivilege;
16028:  SE_AUDIT_NAME','String 'SeAuditPrivilege;
16029:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege;
16030:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege;
16031:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege;
16032:  SE_UNDOCK_NAME','String 'SeUndockPrivilege;
16033:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege;
16034:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege;
16035:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege;
16036: end;
16037:
16038: Proc SIRegister_cyWinUtils(CL: TPSPascalCompiler);
16039: begin
16040:   Type(TWindowsVersion','( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
16041:    +Me, wvWinNt3, wvWinNt4,wvWin2000,wvWinXP,wvWinVista, wvWin7,wvWin8 wvWin8_Or_Upper );
16042:  Func ShellGetExtensionName( FileName :Str) :Str;
16043:  Func ShellGetIconIndex( FileName :Str) : Int;
16044:  Func ShellGetIconHandle( FileName :Str) : HIcon;
16045:  Proc ShellThreadCopy( App_Handle : THandle; fromFile :Str; toFile :Str);
16046:  Proc ShellThreadMove( App_Handle : THandle; fromFile :Str; toFile :Str);
16047:  Proc ShellRenameDir( DirFrom, DirTo:Str);
16048:  Func cyShellExecute(Operation,FileName,Parameters,Directory:str;ShowCmd:Int):Card;
16049:  Proc cyShellExecute1(ExeFilename,Params,ApplicName,ApplicationClass:str;Restore:Bool);
16050:  Proc ShellExecuteAsModal( ExeFilename, ApplicationName, Directory :Str);
16051:  Proc ShellExecuteExAsAdmin( hWnd : HWND; Filename :Str; Parameters :Str);
16052:  Func ShellExecuteEx(aFileName:str;const Parameters:str;const Directory:str;const
       WaitCloseCompletion:bool):Bool;
16053:  Proc RestoreAndSetForegroundWindow( Hnd : Int);
16054:  Func RemoveDuplicatedPathDelimiter( Str :Str) :Str;
16055:  Func cyFileTimeToDateTime( _FT : TFileTime) : TDateTime;
16056:  Func GetModificationDate( Filename :Str) : TDateTime;
16057:  Func GetCreationDate( Filename :Str) : TDateTime;
16058:  Func GetLastAccessDate( Filename :Str) : TDateTime;
16059:  Func FileDelete( Filename :Str) :Bool;
16060:  Func FileIsOpen( Filename :Str) :Bool;
16061:  Proc FilesDelete( FromDirectory :Str; Filter : ShortString);
16062:  Func DirectoryDelete( Directory :Str) :Bool;
16063:  Func GetPrinters( PrintersList : TStrings) : Int;
16064:  Proc SetDefaultPrinter( PrinterName :Str);
16065:  Proc ShowDefaultPrinterWindowProperties( FormParent_Handle : Int);
16066:  Func WinToDosPath( WinPathName :Str) :Str;
16067:  Func DosToWinPath( DosPathName :Str) :Str;
16068:  Func cyGetWindowsVersion : TWindowsVersion;
16069:  Func NTSetPrivilege( sPrivilege :Str; bEnabled :Bool) :Bool;
```

```
16070:  Proc WindowsShutDown( Restart :Bool);
16071:  Proc CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,FileIcon:str;NumIcone:Int);
16072:  Proc GetWindowsFonts( FontsList : TStrings);
16073:  Func GetAvailableFilename( DesiredFileName :Str):Str;
16074: end;
16075:
16076: Proc SIRegister_cyStrUtils(CL: TPSPascalCompiler);
16077: begin
16078:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey );
16079:   Type(TStrLocateOptions', 'set of TStrLocateOption;
16080:   Type(TStringRead', '( srFromLeft, srFromRight );
16081:   Type(TStringReads', 'set of TStringRead;
16082:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive );
16083:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar );
16084:   Type(TWordsOptions', 'set of TWordsOption;
16085:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther );
16086:   Type(TCarTypes', 'set of TCarType;
16087:   //TypeS('TUnicodeCategories', 'set of TUnicodeCategory;
16088:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
16089:  Func Char_GetType( aChar : Char) : TCarType;
16090:  Func SubString_Count( Str :Str; Separator : Char) : Int;
16091:  Func SubString_AtPos( Str :Str; Separator : Char; SubStringIndex : Word) : Int;
16092:  Func SubString_Get( Str :Str; Separator : Char; SubStringIndex : Word) :Str;
16093:  Func SubString_Length( Str :Str; Separator : Char; SubStringIndex : Word) : Int;
16094:  Proc SubString_Add( var Str :Str; Separator : Char; Value :Str);
16095:  Proc SubString_Insert(var Str:str;Separator:Char;SubStrIndex:Word;Value:str);
16096:  Proc SubString_Edit(var Str:str;Separator:Char;SubStringIndex:Word; NewValue:str);
16097:  Func SubString_Remove(var Str:str; Separator:Char; SubStringIndex : Word) :Bool;
16098:  Func SubString_Locate(Str:str;Separator:Char;SubString:Str;Options:TStrLocateOptions):Int;
16099:  Func SubString_Ribbon(Str:str; Separator:Char;Current: Word; MoveBy:Int):Int;;
16100:  Func SubString_Ribbon1(Str:str;Separator:Char;Current:str;MoveBy:Int):str;;
16101:  Func String_Quote( Str :Str) :Str;
16102:  Func String_GetCar(Str:str; Position : Word; ReturnCarIfNotExists : Char) : Char;
16103:  Func String_ExtractCars(fromStr:Str;CarTypes:TCarTypes;IncludeCars,ExcludeCars:Str):str;
16104:  Func String_GetWord( Str :Str; StringRead : TStringRead) :Str;
16105:  Func String_GetInt( Str :Str; StringRead : TStringRead) :Str;
16106:  Func String_ToInt( Str :Str) : Int;
16107:  Func String_Uppercase( Str :Str; Options : TWordsOptions):Str;
16108:  Func String_Lowercase( Str :Str; Options : TWordsOptions):Str;
16109:  Func String_Reverse( Str :Str) :Str;
16110:  Func String_Pos(SubStr:str;Str:str;fromPos:Int;CaseSensitive:TCaseSensitiv)Int;
16111:  Func
       String_Pos1(SubStr:str;Str:str;StringRead:TStringRead;Occurrence:Word;CaseSensitive:TCaseSensitive):Int
16112:  Func String_Copy( Str :Str; fromIndex : Int; toIndex : Int) :Str;;
16113:  Func String_Copy1(Str:str;StringRead:TStringRead;UntilFind:Str;_Inclusive:Boo):str;
16114:  Func String_Copy2(Str:str;Between1:str;Between1MustExist:Boolean;Between2:str;
       Between2MustExist:Boolean;CaseSensitive:TCaseSensitive):str;
16115:  Func String_Delete( Str :Str; fromIndex : Int; toIndex : Int) :Str;;
16116:  Func String_Delete1(Str:str; delStr:str; CaseSensitive:TCaseSensitive):str;;
16117:  Func String_BoundsCut( Str :Str; CutCar : Char; Bounds : TStringReads) :Str;
16118:  Func String_BoundsAdd( Str :Str; AddCar : Char; ReturnLength : Int) :Str;
16119:  Func String_Add(Str:str;StringRead:TStringRead;aCar:Char;ReturnLength:Int):str;
16120:  Func String_End( Str :Str; Cars : Word) :Str;
16121:  Func String_Subst(OldStr:str;NewStr:str;Str:str;
       CaseSensitive:TCaseSensitive;AlwaysFindFromBeginning:Bool):Str;
16122:  Func String_SubstCar( Str :Str; Old, New : Char) :Str;
16123:  Func String_Count(Str:Str; SubStr:Str; CaseSenSitive : TCaseSensitive) : Int;
16124:  Func String_SameCars(Str1,Str2:str;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Int;
16125:  Func String_IsNumbers( Str :Str) :Bool;
16126:  Func SearchPos( SubStr :Str; Str :Str; MaxErrors : Int) : Int;
16127:  Func StringToCsvCell( aStr :Str) :Str;
16128: end;
16129:
16130: Proc SIRegister_cyDateUtils(CL: TPSPascalCompiler);
16131: begin
16132:  Func LongDayName( aDate : TDate) :Str;
16133:  Func LongMonthName( aDate : TDate) :Str;
16134:  Func ShortYearOf( aDate : TDate) : byte;
16135:  Func DateToStrYYYYMMDD( aDate : TDate) :Str;
16136:  Func StrYYYYMMDDToDate( aStr :Str) : TDate;
16137:  Func SecondsToMinutes( Seconds : Int) : Double;
16138:  Func MinutesToSeconds( Minutes : Double) : Int;
16139:  Func MinutesToHours( Minutes : Int) : Double;
16140:  Func HoursToMinutes( Hours : Double) : Int;
16141:  Func ShortTimeStringToTime(ShortTimeStr:str;const ShortTimeFormat:str): TDateTime;
16142:  Proc cyAddMonths( var aMonth, aYear : Word; Months : Int);
16143:  Func MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime;
16144:  Func GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;;
16145:  Func GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:str;const ShortTimeFormat:str):Int64;
16146:  Func GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;;
16147:  Func IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime;var
       RsltBegin:TDateTime;RsltEnd:TDateTime):Bool;
16148:  Func IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
16149:  Func TryToEncodeDate( Year, Month, Day : Int; var RsltDate : TDateTime) :Bool;
16150: end;
16151:
16152: Proc SIRegister_cyObjUtils(CL: TPSPascalCompiler);
16153: begin
16154:   Type(TStringsSortType','( stNone, stStringSensitive, stStringInsensitive, stExtended );
```

```
16155:   Type(TStringsValueKind','( skStringSensitive, skStringInsensitive, skExtended );
16156:   Type(TcyLocateOption','( lCaseInsensitive, lPartialKey );
16157:   Type(TcyLocateOptions','set of TcyLocateOption;
16158:  Func StringsLocate(aList: TStrings; Value:str; Options:TcyLocateOptions): Int;;
16159:  Func StringsLocate1(aList:TStrings; Value:Str; ValueKind:TStringsValueKind):Int;;
16160:  Func StringsAdd(aList:TStrings;Value:str;Unique:Boolean;SortType:TStringsSortType): Int;
16161:  Proc StringsReplace(aList:TStrings;OldStr:str; NewStr:str;ValueKind : TStringsValueKind);
16162:  Proc StringsSort( aList : TStrings; SortType : TStringsSortType);
16163:  Func TreeNodeLocate( ParentNode : TTreeNode; Value :Str) : TTreeNode;
16164:  Func TreeNodeLocateOnLevel(TreeView:TTreeView;OnLevel:Int;Value:str): TTreeNode;
16165:  Func TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Int):TTreeNode;
16166:  Func TreeNodeGetParentOnLevel(ChildNode: TTreeNode; ParentLevel : Int) : TTreeNode;
16167:  Proc TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
16168:  Proc RichEditSetStr( aRichEdit : TRichEdit; FormatedString :Str);
16169:  Proc RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:str;Flags:TReplaceFlags);
16170:  Func GetTopMostControlAtPos(FromControl:TWinControl; aControlPoint : TPoint): TControl;
16171:  Proc cyCenterControl( aControl : TControl);
16172:  Func GetLastParent( aControl : TControl) : TWinControl;
16173:  Func GetControlBitmap( aControl : TWinControl) : TBitmap;
16174:  Func GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap;
16175: end;
16176:
16177: Proc SIRegister_cyBDE(CL: TPSPascalCompiler);
16178: begin
16179:  Func TablePackTable( Tab : TTable) :Bool;
16180:  Func TableRegenIndexes( Tab : TTable) :Bool;
16181:  Func TableShowDeletedRecords( Tab : TTable; Show :Bool) :Bool;
16182:  Func TableUndeleteRecord( Tab : TTable) :Bool;
16183:  Func TableAddIndex(Tab:TTable;FieldName Str;FieldExpression:Str;IOpt:TIndexOptions):Bool;
16184:  Func TableDeleteIndex( Tab : TTable; IndexFieldName :Str) :Bool;
16185:  Func TableEmptyTable( Tab : TTable) :Bool;
16186:  Func TableFindKey( aTable : TTable; Value :Str) :Bool;
16187:  Proc TableFindNearest( aTable : TTable; Value :Str);
16188:  Func
       TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:str;IdxName:ShortString;ReadOnly:Bool):TTable;
16189:  Func
       TableOpen(Tab:TTable;FileName:str;IndexFieldName:str;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
16190:  Func DateToBDESQLDate( aDate : TDate; const DateFormat :Str) :Str;
16191: end;
16192:
16193: Proc SIRegister_cyClasses(CL: TPSPascalCompiler);
16194: begin
16195:   SIRegister_TcyRunTimeDesign(CL);
16196:   SIRegister_TcyShadowText(CL);
16197:   SIRegister_TcyBgPicture(CL);
16198:   SIRegister_TcyGradient(CL);
16199:   SIRegister_tcyBevel(CL);
16200:   //TypeS('TcyBevelClass', 'class of tcyBevel;
16201:   SIRegister_tcyBevels(CL);
16202:   SIRegister_TcyImagelistOptions(CL);
16203:  Proc cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture);
16204: end;
16205:
16206: Proc SIRegister_cyGraphics(CL: TPSPascalCompiler);
16207: begin
16208:  Proc cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; adgradOrientation :
       TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode; Maxdegrade : Byte;'+
16209:   SpeedPercent : Int; const AngleClipRect :Bool; const AngleBuffer : TBitmap);
16210:  Proc cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
16211:  Proc cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromCol,toColor:TColor;MaxDegrad:byte);
16212:  Proc cyGradientFillShape(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:
       Byte;toRect:TRect;OrientationShape:TDgradOrientationShape);
16213:  Proc cyGradientFillAngle(aCanvas:TCanvas;aRect TRect;fromColor,toColor:TColor;MaxDegrad:
       Byte;AngleDegree:Word;const ClipRect:Boolean;const Buffer:TBitmap);
16214:  Proc DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Int);
16215:  Proc cyFrame(aCanvas:TCanvas; var InsideRect:TRect;Color:TColor;const Width:Int);;
16216:  Proc cyFrame1(Canvas:TCanvas;var InsideRect:TRect;LeftColor,TopColor,ightColor,BottomColor:TColor;const
       Width:Int;const RoundRect:bool);;
16217:  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;
16218:  Proc cyDrawButtonFace(Canvas:TCanvas; var Rect:Rect;GradientColor1,
       GradientColor2:TColor;aState:TButtonState;Focused,Hot:Bool);
16219:  Proc cyDrawButton(Canvas:TCanvas;Caption:str;ARect:TRect;radientColor1,GradientColor2:TColor;aState
       TButtonState;Focused,Hot : Bool);
16220:  Proc cyDrawSpeedButtonFace(Canvas:TCanvas;var Rect:TRect;GradientColor1,
       GradientColor2:TColor;aState:TButtonState;Focused,Hot:Bool);
16221:  Proc cyDrawSpeedButton(Canvas:TCanvas;Caption:str;ARect:TRect;GradientColor1,
       GradientColor2:TColor;aState: TButtonState;Focused,Hot:oolean);
16222:  Proc cyDrawCheckBox(Canvas:TCanvas;IsChecked:Boolean;ARect:TRect;const BgColor:TColor;const
       DarkFrameColor:TColor;const LightFrameColor:TColor;const MarkColor:TColor);
16223:  Proc
       cyDrawSingleLineText(Canvas:TCanvas;Text:str;ARect:TRect;Alignment:TAlignment;TextLayout:TTextLayout; const
       IndentX:Int;const IndentY:Int);
16224:  Func DrawTextFormatFlags(aTextFormat:LongInt;Alignment
       TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
16225:  Func DrawTextFormatFlags1(aTextFormat:LongInt;Alignment:TAlignment;Layout:TTextLayout;
       WordWrap:Boolean;CaptionRender:TCaptionRender):LongInt;;
16226:  Proc cyDrawText(CanvasHandl:Card;Text:str;var Rect:TRect;TextFormat:LongInt);
```

```
16227:  Func cyCreateFontIndirect(fromFont : TFont; Angle : Double) : TFont;;
16228:  Func cyCreateFontIndirect1(fromFont: TFont;CaptionOrientation:TCaptionOrientation):TFont;
16229:  Proc cyDrawVerticalText(Canvas : TCanvas; Text :Str; var Rect : TRect; TextFormat : Longint;
        CaptionOrientation:TCaptionOrientation;Alignment:TAlignment;Layout: TextLayout);
16230:  Func DrawLeftTurnPageEffect(Canvas:TCanvas;PageColor:TColor;PageRect:TRect;PercentDone:Int; const
        OnlyCalcFoldLine:Boolean):TLineCoord;
16231:  Func DrawRightTurnPageEffect(Canvas:TCanvas;PageColor:TColor;PageRect:TRect;PercentDone:Int;const
        OnlyCalcFoldLine:Boolean):TLineCoord;
16232:  Func PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean;
16233:  Func IconIsTransparentAtPos(aIcon : TIcon; aPoint : TPoint) :Bool;
16234:  Func MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) :Bool;
16235:  Func PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) :Bool;
16236:  Proc DrawCanvas(Destination:TCanvas;DestRect:TRect;Source:TCanvas;SourceRect:TRect);
16237:  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);
16238:  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;
16239:  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);
16240:  Proc DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap);
16241:  Func ValidGraphic( aGraphic : TGraphic) :Bool;
16242:  Func ColorSetPercentBrightness( Color : TColor; PercentLight : Int) : TColor;
16243:  Func ColorModify( Color : TColor; incR, incG, incB : Int) : TColor;
16244:  Func ColorSetPercentContrast( Color : TColor; IncPercent : Int) : TColor;
16245:  Func ColorSetPercentPale( Color : TColor; IncPercent : Int) : TColor;
16246:  Func MediumColor( Color1, Color2 : TColor) : TColor;
16247:  Func ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect;
16248:  Func ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect;
16249:  Func CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect;
16250:  Proc InflateRectPercent( var aRect : TRect; withPercent : Double);
16251:  Func GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect;
16252:  Func GetProportionalRect( fromRect, InsideRect : TRect) : TRect;
16253:  Func PointInRect( const aPt : TPoint; const aRect : TRect) :Bool;
16254:  Func PointInEllispe( const aPt : TPoint; const aRect : TRect) :Bool;
16255:  Func CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText :Str) : Int;
16256: end;
16257:
16258: Proc SIRegister_cyTypes(CL: TPSPascalCompiler);
16259: begin
16260:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight );
16261:   Type(TGlyphLayout', '( glTop, glCenter, glBottom );
16262:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome );
16263:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis );
16264:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed );
16265:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
16266:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft);
16267:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional );
16268:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext );
16269:   Type(TDgradOrientation','( dgdVertical,dgdHorizontal,dgdAngle,dgdRadial,dgdRectangle );
16270:   Type(TDgradOrientationShape', '( osRadial, osRectangle );
16271:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
         bmInvertReverseFromColor);
16272:   Type(TRunTimeDesignJob','(rjNothing,rjMove,rjResizeTop,rjResizeBottom,rjResizeLeft,rjResizeTopLeft,
         rjResizeBottomLeft,rjResizeRight,rjResizeTopRight,rjResizeBottomRight );
16273:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end;
16274:  cCaptionOrientationWarning,String)(Note text orientation doesnt work with all fonts!;
16275: end;
16276:
16277: Proc SIRegister_WinSvc(CL: TPSPascalCompiler);
16278: begin
16279:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive;
16280:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive;
16281:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA;
16282:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed;
16283:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed;
16284:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA;
16285:  Const SC_GROUP_IDENTIFIERA','String'). '+;
16286:  Const SC_GROUP_IDENTIFIERW','String') '+;
16287:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA;
16288:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
16289:  Const SERVICE_ACTIVE','LongWord')( $00000001);
16290:  Const SERVICE_INACTIVE','LongWord $00000002);
16291:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
16292:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
16293:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
16294:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
16295:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
16296:  Const SERVICE_STOPPED','LongWord $00000001);
16297:  Const SERVICE_START_PENDING','LongWord $00000002);
16298:  Const SERVICE_STOP_PENDING','LongWord $00000003);
16299:  Const SERVICE_RUNNING','LongWord $00000004);
16300:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
16301:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
16302:  Const SERVICE_PAUSED','LongWord $00000007);
```

```
16303:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
16304:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
16305:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
16306:  Const SC_MANAGER_CONNECT','LongWord $0001);
16307:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
16308:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
16309:  Const SC_MANAGER_LOCK','LongWord $0008);
16310:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
16311:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
16312:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
16313:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
16314:  Const SERVICE_QUERY_STATUS','LongWord $0004);
16315:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
16316:  Const SERVICE_START','LongWord $0010);
16317:  Const SERVICE_STOP','LongWord $0020);
16318:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
16319:  Const SERVICE_INTERROGATE','LongWord $0080);
16320:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
16321:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
16322:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
16323:  Const SERVICE_ADAPTER','LongWord $00000004);
16324:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
16325:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
16326:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
16327:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
16328:  Const SERVICE_BOOT_START','LongWord $00000000);
16329:  Const SERVICE_SYSTEM_START','LongWord $00000001);
16330:  Const SERVICE_AUTO_START','LongWord $00000002);
16331:  Const SERVICE_DEMAND_START','LongWord $00000003);
16332:  Const SERVICE_DISABLED','LongWord $00000004);
16333:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
16334:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
16335:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
16336:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
16337:   TypeS('SC_HANDLE', 'THandle;
16338:   //TypeS('LPSC_HANDLE', '^SC_HANDLE // will not work;
16339:   TypeS('SERVICE_STATUS_HANDLE', 'DWORD;
16340:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
16341:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
16342:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end;
16343:   Const SERVICE_STATUS', '_SERVICE_STATUS;
16344:   Const TServiceStatus', '_SERVICE_STATUS;
16345:   TypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
16346:    +'playName : PChar; ServiceStatus : TServiceStatus; end;
16347:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA;
16348:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA;
16349:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA;
16350:   TEnumServiceStatus', 'TEnumServiceStatusA;
16351:   SC_LOCK', '___Pointer;
16352:   _QUERY_SERVICE_LOCK_STATUSA,record fIsLocked:DWORD;lpLockOner:PChar;dwLockDuration:DWORD;end;
16353:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA;
16354:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA;
16355:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA;
16356:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA;
16357:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW;
16358:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA;
16359:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
16360:    +ype : DWORD; dwErrorControl:DWORD;lpBinaryPathName: PChar; lpLoadO'
16361:   +rderGroup:PChar;dwTagId:DWORD;lpDependencies:PChar;lpServiceStartName:PChar;lpDisplayName:PChar;end;
16362:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA;
16363:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA;
16364:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA;
16365:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA;
16366:   TQueryServiceConfig', 'TQueryServiceConfigA;
16367:  Func CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL;
16368:  Func ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
16369:  Func CreateService(hSCManager:SC_HANDLE;lpServiceName,lpDisplayName:PChar;dwDesiredAccess,dwServiceType,
      dwStartType,dwErrorControl:DWORD;lpBinaryPathName,lpLoadOrderGroup:PChar;
16370:  +' lpdwTagId: DWORD;lpDependencies,lpServiceStartName, lpPassword : PChar): SC_HANDLE;
16371:  Func CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
16372:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE;
16373:  Func DeleteService( hService : SC_HANDLE) : BOOL;
16374:  Func EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD; var
      lpServices:TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned: DWORD) : BOOL;
16375:  Func EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState:DWORD;var lpServices:
      TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD):BOOL;
16376:  Func GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var lpcchBuffer:DWORD):BOOL;
16377:  Func GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
      lpcchBuffer:DWORD):BOOL;
16378:  Func LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK;
16379:  Func NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL;
16380:  Func OpenSCManager(lpMachineName,lpDatabaseName:PChar;dwDesiredAccess:DWORD):SC_HANDLE;
16381:  Func OpenService(hSCManager:SC_HANDLE;lpServiceNam:PChar;dwDesiredAccess:DWORD):SC_HANDLE;
16382:  Func QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
      cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL;
16383:  Func QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
16384:  Func SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
16385:  Func StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
```

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

```
16471:   SIRegister_TJclControlPanelSummaryInformation(CL);
16472:   SIRegister_TJclVolumeSummaryInformation(CL);
16473:   SIRegister_TJclShareSummaryInformation(CL);
16474:   SIRegister_TJclLinkSummaryInformation(CL);
16475:   SIRegister_TJclQuerySummaryInformation(CL);
16476:   SIRegister_TJclImageInformation(CL);
16477:   SIRegister_TJclJpegSummaryInformation(CL);
16478: end;
16479:
16480: Proc SIRegister_Jcl8087(CL: TPSPascalCompiler);
16481: begin
16482:   AddTypeS('T8087Precision','( pcSingle, pcReserved, pcDouble, pcExtended );
16483:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate );
16484:   T8087Infinity','( icProjective, icAffine );
16485:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
16486:   TypeS('T8087Exceptions', set of T8087Exception;
16487:  Func Get8087ControlWord : Word;
16488:  Func Get8087Infinity : T8087Infinity;
16489:  Func Get8087Precision : T8087Precision;
16490:  Func Get8087Rounding : T8087Rounding;
16491:  Func Get8087StatusWord( ClearExceptions :Bool) : Word;
16492:  Func Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity;
16493:  Func Set8087Precision( const Precision : T8087Precision) : T8087Precision;
16494:  Func Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding;
16495:  Func Set8087ControlWord( const Control : Word) : Word;
16496:  Func ClearPending8087Exceptions : T8087Exceptions;
16497:  Func GetPending8087Exceptions : T8087Exceptions;
16498:  Func GetMasked8087Exceptions : T8087Exceptions;
16499:  Func SetMasked8087Exceptions(Exceptions:T8087Exceptions;ClearBefore:Bool):T8087Exceptions;
16500:  Func Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions;
16501:  Func Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore:Bool):T8087Exceptions;
16502: end;
16503:
16504: Proc SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
16505: begin
16506:  Proc BoxMoveSelectedItems( SrcList, DstList : TWinControl);
16507:  Proc BoxMoveAllItems( SrcList, DstList : TWinControl);
16508:  Proc BoxDragOver(List:TWinControl;Sorce:TObject;X,Y:Int;State:TDragState;var Accept:Bool;Sorted:Bool;
16509:  Proc BoxMoveFocusedItem( List : TWinControl; DstIndex : Int);
16510:  Proc BoxMoveSelected( List : TWinControl; Items : TStrings);
16511:  Proc BoxSetItem( List : TWinControl; Index : Int);
16512:  Func BoxGetFirstSelection( List : TWinControl) : Int;
16513:  Func BoxCanDropItem(List:TWinControl; X,Y:Int; var DragIndex : Int) :Bool;
16514: end;
16515:
16516: Proc SIRegister_UrlMon(CL: TPSPascalCompiler);
16517: begin
16518:  //ConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context;
16519:  //ConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee;
16520:  ConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
16521:   type ULONG', 'Cardinal; LPCWSTR', 'PChar;  TypeS('LPWSTR', 'PChar;
16522:   LPSTR', 'PChar;
16523:   TBindVerb', 'ULONG;
16524:   TBindInfoF', 'ULONG;
16525:   TBindF', 'ULONG;
16526:   TBSCF', 'ULONG;
16527:   TBindStatus', 'ULONG;
16528:   TCIPStatus', 'ULONG;
16529:   TBindString', 'ULONG;
16530:   TPiFlags', 'ULONG;
16531:   TOIBdgFlags', 'ULONG;
16532:   TParseAction', 'ULONG;
16533:   TPSUAction', 'ULONG;
16534:   TQueryOption', 'ULONG;
16535:   TPUAF', 'ULONG;
16536:   TSZMFlags', 'ULONG;
16537:   TUrlZone', 'ULONG;
16538:   TUrlTemplate', 'ULONG;
16539:   TZAFlags', 'ULONG;
16540:   TUrlZoneReg', 'ULONG;
16541:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
16542:  ConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
16543:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
16544:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
16545:  const 'CF_NULL','LongInt').SetInt( 0);
16546:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
16547:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain;
16548:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext;
16549:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap;
16550:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript;
16551:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff;
16552:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic;
16553:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav;
16554:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav;
16555:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif;
16556:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg;
16557:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg;
16558:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff;
16559:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png;
```

```
16560:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp;
16561:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg;
16562:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf;
16563:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf;
16564:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi;
16565:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg;
16566:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals;
16567:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream;
16568:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream;
16569:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf;
16570:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff;
16571:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio;
16572:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm;
16573:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime;
16574:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo;
16575:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie;
16576:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html;
16577:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
16578:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
16579:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
16580:   Interface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding;
16581:   SIRegister_IPersistMoniker(CL);
16582:   SIRegister_IBindProtocol(CL);
16583:   SIRegister_IBinding(CL);
16584:  const BINDVERB_GET','LongWord').SetUInt( $00000000);
16585:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
16586:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
16587:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
16588:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
16589:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
16590:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
16591:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
16592:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
16593:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
16594:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
16595:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
16596:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16597:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16598:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16599:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
16600:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
16601:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
16602:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
16603:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
16604:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
16605:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
16606:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
16607:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
16608:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
16609:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
16610:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
16611:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16612:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16613:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16614:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16615:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
16616:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16617:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16618:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16619:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16620:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16621:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16622:  const BINDSTATUS_BEGINDOWNLOADCOMPONENTS,LongInt).SetInt(BINDSTATUS_ENDDOWNLOADDATA + 1);
16623:  const 'BINDSTATUS_INSTALLINGCOMPONENTS',LongInt).SetInt(BINDSTATUS_BEGINDOWNLOADCOMPONENTS+1);
16624:  const BINDSTATUS_ENDDOWNLOADCOMPONENTS,LongInt).SetInt(BINDSTATUS_INSTALLINGCOMPONENTS+1);
16625:  const BINDSTATUS_USINGCACHEDCOPY,LongInt).SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16626:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16627:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16628:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16629:  const BINDSTATUS_CACHEFILENAMEAVAILABLE,LongInt).SetInt(BINDSTATUS_MIMETYPEAVAILABLE+1);
16630:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt(BINDSTATUS_CACHEFILENAMEAVAILABLE+1);
16631:  const BINDSTATUS_ENDSYNCOPERATION',LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16632:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16633:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16634:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16635:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16636:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16637:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16638:  const BINDSTATUS_CLASSINSTALLLOCATION,LongInt).SetInt(BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE+1);
16639:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16640:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16641:  const BINDSTATUS_CONTENTDISPOSITIONATTACH,LongInt).SetInt(BINDSTATUS_LOADINGMIMEHANDLER+1);
16642:  const 'BINDSTATUS_FILTERREPORTMIMETYPE',LongInt).SetInt(BINDSTAT_CONTENTDISPOSITIONATTACH+1);
16643:  const BINDSTATUS_CLSIDCANINSTANTIATE,LongInt).SetInt(BINDSTATUS_FILTERREPORTMIMETYPE+ 1);
16644:  const BINDSTATUS_IUNKNOWNAVAILABLE,LongInt).SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16645:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16646:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16647:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16648:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
```

```
16649:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16650:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt(BINDSTATUS_COOKIE_SENT + 1);
16651:  const BINDSTATUS_COOKIE_SUPPRESSED,LongInt).SetInt(BINDSTATUS_COMPACT_POLICY_RECEIVED+1);
16652:  const BINDSTATUS_COOKIE_STATE_UNKNOWN,LongInt).SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16653:  const BINDSTATUS_COOKIE_STATE_ACCEPT,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_UNKNOWN+ 1);
16654:  const BINDSTATUS_COOKIE_STATE_REJECT,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16655:  const BINDSTATUS_COOKIE_STATE_PROMPT,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_REJECT + 1);
16656:  const BINDSTATUS_COOKIE_STATE_LEASH,LongInt).SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16657:  const BINDSTATUS_COOKIE_STATE_DOWNGRADE,LongInt).SetInt(BINDSTATUS_COOKIE_STATE_LEASH+1);
16658:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16659:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16660:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16661: const'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED',LongInt).SetInt(BINDSTATUS_SESSION_COOKIE_RECEIVED+1);
16662: const'BINDSTATUS_SESSION_COOKIES_ALLOWED',LongInt).SetInt(BINDSTATUS_PERSISTENT_COOKIE_RECEIVED+1);
16663:  const BINDSTATUS_CACHECONTROL,LongInt).SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16664:  const BINDSTATUS_CONTENTDISPOSITIONFILENAME,LongInt).SetInt(BINDSTATUS_CACHECONTROL + 1);
16665: const'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt(BINDSTATUS_CONTENTDISPOSITIONFILENAME+1);
16666:  const BINDSTATUS_PUBLISHERAVAILABLE,LongInt).SetInt(BINDSTATUS_MIMETEXTPLAINMISMATCH+1);
16667:  const BINDSTATUS_DISPLAYNAMEAVAILABLE,LongInt).SetInt(BINDSTATUS_PUBLISHERAVAILABLE + 1);
16668:  // PBindInfo', '^TBindInfo // will not work;
16669:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16670:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16671:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16672:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16673:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end;
16674:   TBindInfo', '_tagBINDINFO;
16675:   BINDINFO', '_tagBINDINFO;}
16676:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16677:    +'criptor : DWORD; bInheritHandle : BOOL; end;
16678:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES;
16679:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES;
16680:   //PRemBindInfo', '^TRemBindInfo // will not work;
16681:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16682:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16683:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16684:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16685:    +'n; dwReserved : DWORD; end;
16686:   TRemBindInfo', '_tagRemBINDINFO;
16687:   RemBINDINFO', '_tagRemBINDINFO;}
16688:   //PRemFormatEtc', '^TRemFormatEtc // will not work;
16689: tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end;
16690:   TRemFormatEtc', 'tagRemFORMATETC;
16691:   RemFORMATETC', 'tagRemFORMATETC;
16692:   SIRegister_IBindStatusCallback(CL);
16693:   SIRegister_IAuthenticate(CL);
16694:   SIRegister_IHttpNegotiate(CL);
16695:   SIRegister_IWindowForBindingUI(CL);
16696:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16697:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16698:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16699:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16700:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16701:  const CIP_TRUST_VERIFICATION_COMPONENT_MISSING,LongInt).SetInt( CIP_NAME_CONFLICT + 1);
16702:  const CIP_EXE_SELF_REGISTERATION_TIMEOUT,LongInt(CIP_TRUST_VERIFICATION_COMPONET_MISSING+1);
16703:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16704:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16705:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16706:   SIRegister_ICodeInstall(CL);
16707:   SIRegister_IWinInetInfo(CL);
16708:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16709:   SIRegister_IHttpSecurity(CL);
16710:   SIRegister_IWinInetHttpInfo(CL);
16711:   SIRegister_IBindHost(CL);
16712:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16713:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16714:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16715:  Func URLOpenStream(p1:IUnknown;p2:PChar; p3:DWORD; p4:IBindStatusCallback): HResult;
16716:  Func URLOpenPullStream(p1:IUnknown;p2:PChar;p3:DWORD; BSC:IBindStatusCallback):HResult;
16717:  Func
        URLDownloadToFile(Caller:IUnknown;URL:PChar;FileName:PChar;Reserved:DWORD;StatusCB:IBindStatusCallback):
        HResult;
16718:  Func URLDownloadToCacheFile(p1:IUnknown;p2:PChar;p3:PChar;p4:DWORD;p5:DWORD;p6:IBindStatusCallback):
        HResult;
16719:  Func URLOpenBlockingStream(p1:IUnknown;p2:PChar;out p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult;
16720:  Func HlinkGoBack( unk : IUnknown) : HResult;
16721:  Func HlinkGoForward( unk : IUnknown) : HResult;
16722:  Func HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult;
16723: // Func HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult;
16724:  SIRegister_IInternet(CL);
16725:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16726:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16727:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16728:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16729:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16730:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16731:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16732:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16733:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16734:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
```

```
16735:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16736:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16737:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16738:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16739:   //POLEStrArray', '^TOLESTRArray // will not work;
16740:   SIRegister_IInternetBindInfo(CL);
16741:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16742:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16743:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16744:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16745:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16746:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16747:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16748:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16749:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16750:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16751:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16752:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16753:   //PProtocolData', '^TProtocolData // will not work;
16754:   _tagPROTOCOLDATA,record grfFlags:DWORD; dwState:DWORD; pData:TObject;cbData:ULONG; end;
16755:   TProtocolData', '_tagPROTOCOLDATA;
16756:   PROTOCOLDATA', '_tagPROTOCOLDATA;
16757:   Interface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink,'IInternetProtocolSink;
16758:   SIRegister_IInternetProtocolRoot(CL);
16759:   SIRegister_IInternetProtocol(CL);
16760:   SIRegister_IInternetProtocolSink(CL);
16761:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16762:   SIRegister_IInternetSession(CL);
16763:   SIRegister_IInternetThreadSwitch(CL);
16764:   SIRegister_IInternetPriority(CL);
16765:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16766:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16767:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16768:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16769:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16770:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16771:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16772:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16773:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16774:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16775:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16776:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16777:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16778:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16779:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16780:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16781:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16782:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16783:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16784:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16785:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16786:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16787:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16788:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16789:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16790:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16791:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16792:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16793:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16794:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16795:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16796:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16797:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16798:   SIRegister_IInternetProtocolInfo(CL);
16799:   IOInet', 'IInternet;
16800:   IOInetBindInfo', 'IInternetBindInfo;
16801:   IOInetProtocolRoot', 'IInternetProtocolRoot;
16802:   IOInetProtocol', 'IInternetProtocol;
16803:   IOInetProtocolSink', 'IInternetProtocolSink;
16804:   IOInetProtocolInfo', 'IInternetProtocolInfo;
16805:   IOInetSession', 'IInternetSession;
16806:   IOInetPriority', 'IInternetPriority;
16807:   IOInetThreadSwitch', 'IInternetThreadSwitch;
16808:  Func
       CoInternetParseUrl(pwzUrl:LPCWSTR;ParseAction:TParseAction;dwFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD;
       var pcchResult:DWORD;dwReserved:DWORD):HResult;
16809:  Func CoInternetCombineUrl(pwzBaseUrl,
       pwzRelativeUrl:LPCWSTR;dwCombineFlags:DWORD;pszResult:LPWSTR;cchResult: DWORD;var
       pcchResult:DWORD;dwReserved:DWORD):HResult;
16810:  Func CoInternetCompareUrl(pwzUrl1, pwzUrl2 LPCWSTR;dwFlags DWORD): HResult;
16811:  Func CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD;dwReserved:DWORD):HResult;
16812:  Func
       CoInternetQueryInfo(pwzUrl:LPCWSTR;QueryOptions:TQueryOption;dwQueryFlags:DWORD;pvBuffer:TObject;cbBuffer
       :DWORD;var pcbBuffer:DWORD;dwReserved DWORD):HResult;
16813:  Func CoInternetGetSession(dwSessionMode:DWORD;var
       pIInternetSession:IInternetSes;dwReserved:DWORD):HResult;
16814:  Func CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16815:  Func
       OInetParseUrl(pwzUrl:LPCWSTR;ParseAction:TParseAction;dwFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD; var
       pcchResult:DWORD;dwReserved:DWORD):HResult;
```

```
16816:  Func OInetCombineUrl(pwzBaseUrl,
        pwzRelativeUrl:LPCWSTR;dwCombineFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD;var
        pcchResult:DWORD;dwReserved:DWORD):HResult;
16817:  Func OInetCompareUrl(pwzUrl1,pwzUrl2  LPCWSTR;dwFlags: DWORD): Hresult;
16818:  Func OInetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags:DWORD;dwReserved:DWORD): HResult;
16819:  Func
        OInetQueryInfo(pwzUrl:LPCWSTR;QueryOptions:TQueryOption;dwQueryFlags:DWORD;pvBuffer:TObject;cbBuffer: DWORD;var
        pcbBuffer:DWORD;dwReserved:DWORD):HResult;
16820:  Func OInetGetSession(dwSessionMode:DWORD;var pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16821:   //Func CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult;
16822:  //Proc ReleaseBindInfo( const bindinfo : TBindInfo);
16823: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16824: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16825:  //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16826:  //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16827:  //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16828:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16829:   SIRegister_IInternetSecurityMgrSite(CL);
16830:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16831:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16832:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16833:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16834:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16835:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16836:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16837:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16838:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16839:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16840:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16841:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16842:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16843:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16844:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16845:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16846:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16847:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16848:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16849:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16850:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16851:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16852:  const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16853:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16854:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16855:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16856:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16857:   SIRegister_IInternetSecurityManager(CL);
16858:   SIRegister_IInternetHostSecurityManager(CL);
16859:   SIRegister_IInternetSecurityManagerEx(CL);
16860:  const URLACTION_MIN','LongWord').SetUInt( $00001000);
16861:  const URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16862:  const URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16863:  const URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16864:  const URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16865:  const URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16866:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16867:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16868:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16869:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16870:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16871:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16872:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16873:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16874:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16875:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16876:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16877:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16878:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16879:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16880:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16881:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16882:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16883:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16884:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16885:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16886:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16887:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16888:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16889:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16890:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16891:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16892:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16893:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16894:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16895:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16896:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16897:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16898:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16899:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16900:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
```

```
16901:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16902:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16903:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16904:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16905:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16906:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16907:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16908:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16909:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16910:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16911:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16912:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16913:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16914:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16915:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16916:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16917:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16918:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16919:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16920:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16921:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16922:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16923:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16924:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16925:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16926:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16927:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16928:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16929:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16930:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16931:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16932:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16933:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16934:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16935:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16936:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16937:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16938:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16939:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16940:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16941:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16942:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16943:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16944:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16945:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16946:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16947:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16948:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16949:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16950:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16951:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16952:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16953:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16954:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16955:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16956:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16957:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16958:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16959:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16960:  Func GetUrlPolicyPermissions( dw : DWORD) : DWORD;
16961:  Func SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD;
16962:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16963:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16964:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16965:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16966:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16967:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16968:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16969:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16970:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16971:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16972:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16973:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16974:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16975:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16976:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16977:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16978:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16979:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16980:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16981:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16982:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16983:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16984:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16985:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16986:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16987:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16988:   //PZoneAttributes', '^TZoneAttributes // will not work;
16989:  _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;
```

```
16990:   { _ZONEATTRIBUTES = packed record
16991:     cbSize: ULONG;
16992:     szDisplayName: array [0..260 - 1] of WideChar;
16993:     szDescription: array [0..200 - 1] of WideChar;
16994:     szIconPath: array [0..260 - 1] of WideChar;
16995:     dwTemplateMinLevel: DWORD;
16996:     dwTemplateRecommended: DWORD;
16997:     dwTemplateCurrentLevel: DWORD;
16998:     dwFlags: DWORD;
16999:   end; }
17000:   TZoneAttributes', '_ZONEATTRIBUTES;
17001:   ZONEATTRIBUTES', '_ZONEATTRIBUTES;
17002:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
17003:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
17004:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
17005:   SIRegister_IInternetZoneManager(CL);
17006:   SIRegister_IInternetZoneManagerEx(CL);
17007:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
17008:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
17009:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
17010:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
17011:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
17012:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
17013:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
17014:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
17015:   //PCodeBaseHold', '^TCodeBaseHold // will not work;
17016:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
17017:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle:DWORD; end;
17018:   TCodeBaseHold', '_tagCODEBASEHOLD;
17019:   CODEBASEHOLD', '_tagCODEBASEHOLD;
17020:   //PSoftDistInfo', '^TSoftDistInfo // will not work;
17021:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
17022:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
17023:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
17024:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
17025:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end;
17026:   TSoftDistInfo', '_tagSOFTDISTINFO;
17027:   SOFTDISTINFO', '_tagSOFTDISTINFO;
17028:   SIRegister_ISoftDistExt(CL);
17029:  Func GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult;
17030:  Func SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
      dwAdvertisedVersionLS : DWORD) : HResult;
17031:   SIRegister_IDataFilter(CL);
17032:  // PProtocolFilterData', '^TProtocolFilterData // will not work;
17033:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
17034:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
17035:    +'terFlags : DWORD; end;
17036:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA;
17037:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA;
17038:   //PDataInfo', '^TDataInfo // will not work;
17039:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
17040:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end;
17041:   TDataInfo', '_tagDATAINFO;
17042:   DATAINFO', '_tagDATAINFO;
17043:   SIRegister_IEncodingFilterFactory(CL);
17044:  Func IsLoggingEnabled( pszUrl : PChar) : BOOL;
17045:  //Func IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL;
17046:  //Func IsLoggingEnabledW( pszUrl : PWideChar) : BOOL;
17047:   //PHitLoggingInfo', '^THitLoggingInfo // will not work;
17048:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
17049:    +rlName:LPSTR;StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo:LPSTR; end;
17050:   THitLoggingInfo', '_tagHIT_LOGGING_INFO;
17051:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO;
17052:  Func WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL;
17053: end;
17054:
17055: Proc SIRegister_DFFUtils(CL: TPSPascalCompiler);
17056: begin
17057:  Proc reformatMemo( const m : TCustomMemo);
17058:  Proc SetMemoMargins( m : TCustomMemo; const L, T, R, B : Int);
17059:  Proc MoveToTop( memo : TMemo);
17060:  Proc ScrollToTop( memo : TMemo);
17061:  Func LineNumberClicked( memo : TMemo) : Int;
17062:  Func MemoClickedLine( memo : TMemo) : Int;
17063:  Func ClickedMemoLine( memo : TMemo) : Int;
17064:  Func MemoLineClicked( memo : TMemo) : Int;
17065:  Func LinePositionClicked( Memo : TMemo) : Int;
17066:  Func ClickedMemoPosition( memo : TMemo) : Int;
17067:  Func MemoPositionClicked( memo : TMemo) : Int;
17068:  Proc AdjustGridSize( grid : TDrawGrid);
17069:  Proc DeleteGridRow( Grid : TStringGrid; const ARow : Int);
17070:  Proc InsertgridRow( Grid : TStringGrid; const ARow : Int);
17071:  Proc Sortgrid( Grid : TStringGrid; const SortCol : Int);;
17072:  Proc Sortgrid1(Grid:TStringGrid; const SortCol : Int; sortascending :Bool);;
17073:  Proc sortstrDown( var s :Str);
17074:  Proc sortstrUp( var s :Str);
17075:  Proc rotatestrleft( var s :Str);
17076:  Func dffstrtofloatdef( s :Str; default : extended) : extended;
17077:  Func deblank( s :Str) :Str;
```

```
17078:  Func IntToBinaryString( const n : Int; MinLength : Int) :Str;
17079:  Proc FreeAndClearListBox( C : TListBox);;
17080:  Proc FreeAndClearMemo( C : TMemo);;
17081:  Proc FreeAndClearStringList( C : TStringList);;
17082:  Func dffgetfilesize( f : TSearchrec) : int64;
17083: end;
17084:
17085: Proc SIRegister_MathsLib(CL: TPSPascalCompiler);
17086: begin
17087:   TypeS('intset', set of byte;
17088:   TPoint64', 'record x : int64; y : int64; end;
17089:  Func GetNextPandigital( size : Int; var Digits : array of Int) :Bool;
17090:  Func IsPolygonal( T : int64; var rank : array of Int) :Bool;
17091:  Func GeneratePentagon( n : Int) : Int;
17092:  Func IsPentagon( p : Int) :Bool;
17093:  Func isSquare( const N : int64) :Bool;
17094:  Func isCube( const N : int64) :Bool;
17095:  Func isPalindrome( const n : int64) :Bool;;
17096:  Func isPalindrome1( const n : int64; var len : Int) :Bool;;
17097:  Func GetEulerPhi( n : int64) : int64;
17098:  Func dffIntPower( a, b : int64) : int64;;
17099:  Func IntPower1( a : extended; b : int64) : extended;;
17100:  Func gcd2( a, b : int64) : int64;
17101:  Func GCDMany( A : array of Int) : Int;
17102:  Func LCMMany( A : array of Int) : Int;
17103:  Proc ContinuedFraction(A: array of int64;const wholepart:int;var numerator,denominator:int64);
17104:  Func dffFactorial( n : int64) : int64;
17105:  Func digitcount( n : int64) : Int;
17106:  Func nextpermute( var a : array of Int) :Bool;
17107:  Func convertfloattofractionstring(N:extended;maxdenom:Int;multipleof:boolean):str;
17108:  Func convertStringToDecimal( s :Str; var n : extended) :Bool;
17109:  Func InttoBinaryStr( nn : Int) :Str;
17110:  Func StrtoAngle( const s :Str; var angle : extended) :Bool;
17111:  Func AngleToStr( angle : extended) :Str;
17112:  Func AngleToStr2( angle : extended; decimal:boolean) :Str;
17113:  Func deg2rad( deg : extended) : extended;
17114:  Func rad2deg( rad : extended) : extended;
17115:  Func GetLongToMercProjection( const long : extended) : extended;
17116:  Func GetLatToMercProjection( const Lat : Extended) : Extended;
17117:  Func GetMercProjectionToLong( const ProjLong : extended) : extended;
17118:  Func GetMercProjectionToLat( const ProjLat : extended) : extended;
17119:   SIRegister_TPrimes(CL);
17120:   //RIRegister_TPrimes(CL);
17121:  //ConstantN('deg','LongInt').SetInt( char( 176));
17122:  ConstantN('minmark','LongInt').SetInt(( 180));
17123:  Func Random64( const N : Int64) : Int64;;
17124:  Proc Randomize64;
17125:  Func Random641 : extended;;
17126:  Proc GetParens( Variables:Str; OpChar:char; var list:TStringlist)//DFFUtils
17127:  add 4.7.4.62
17128:  Func MercScaling(Long0,lat0, long1, lat1:extended; x0,y0,x1,y1:integer) :TMercScalingRec');
17129:  Func LonglatToPlotPt(Long,lat:Extended; Scalerec:TMercScalingrec):TPoint');
17130:  Func PlotPtToLonglat(PlotPt:TPoint; Scalerec:TMercScalingrec):TRealPoint');
17131:  Func SphericalEarthDistance(lat1,lon1,lat2,lon2:extended; Units:integer):extended');
17132: end;
17133:
17134: Proc SIRegister_UGeometry(CL: TPSPascalCompiler);
17135: begin
17136:   TrealPoint', 'record x : extended; y : extended; end;
17137:   Tline', 'record p1 : TPoint; p2 : TPoint; end;
17138:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end;
17139:   TCircle', 'record cx : Int; cy : Int; r : Int; end;
17140:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end;
17141:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError );
17142:  Func realpoint( x, y : extended) : TRealPoint;
17143:  Func dist( const p1, p2 : TrealPoint) : extended;
17144:  Func intdist( const p1, p2 : TPoint) : Int;
17145:  Func dffLine( const p1, p2 : TPoint) : Tline;;
17146:  Func Line1( const p1, p2 : TRealPoint) : TRealline;;
17147:  Func dffCircle( const cx, cy, R : Int) : TCircle;;
17148:  Func Circle1( const cx, cy, R : extended) : TRealCircle;;
17149:  Func GetTheta( const L : TLine) : extended;;
17150:  Func GetTheta1( const p1, p2 : TPoint) : extended;;
17151:  Func GetTheta2( const p1, p2 : TRealPoint) : extended;;
17152:  Proc Extendline( var L : TLine; dist : Int);;
17153:  Proc Extendline1( var L : TRealLine; dist : extended);;
17154:  Func Linesintersect( line1, line2 : TLine) :Bool;
17155:  Func ExtendedLinesIntersect(Line1,Line2:TLine;const extendlines:bool;var IP:TPoint):bool;
17156:  Func ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
17157:  Func Intersect(L1,L2: TLine; var pointonborder:boolean; var IP : TPoint) :Bool;
17158:  Func PointPerpendicularLine( L : TLine; P : TPoint) : TLine;
17159:  Func PerpDistance( L : TLine; P : TPoint) : Int;
17160:  Func AngledLineFromLine(L:TLine;P:TPoint;Dist:extended; alpha : extended) : TLine;;
17161:  Func AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordiates:bool):TLine;
17162:  Func PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult;
17163:  Func PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var Clockwise:bool):Int;
17164:  Proc InflatePolygon(const points:array of Tpoint;var points2:array of TPoint;var area:Int;const
       screenCoordinates:bool; const inflateby : Int);
17165:  Func PolyBuiltClockwise(const points:array of TPoint;const screencoordies:bool):bool;
```

```
17166:  Func DegtoRad( d : extended) : extended;
17167:  Func RadtoDeg( r : extended) : extended;
17168:  Proc TranslateLeftTo( var L : TLine; newend : TPoint);;
17169:  Proc TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);;
17170:  Proc RotateRightEndBy( var L : TLine; alpha : extended);
17171:  Proc RotateRightEndTo( var L : TLine; alpha : extended);;
17172:  Proc RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);;
17173:  Func CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) :Bool;;
17174:  Func CircleCircleIntersect1(c1,c2:TRealCircle; var IP1,Ip2  TRealPoint) :Bool;;
17175:  Func PointCircleTangentLines(const C:TCircle;const P:TPoint;var L1,L2:TLine):bool;
17176: Func CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
17177: end;
17178:
17179:
17180: Proc SIRegister_UAstronomy(CL: TPSPascalCompiler);
17181: begin
17182:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat );
17183:   TDTType', '( ttLocal, ttUT, ttGST, ttLST );
17184:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end;
17185:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl:TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
17186:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
17187:    + TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
17188:    +arth : extended; Phase : extended; end;
17189:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
17190:    +Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end;
17191:   TEclipseRec', 'record Msg :Str; Status : Int; FirstConta'
17192:    +ct: TDatetime;LastContact:TDateTime;Magnitude:Extended;MaxeclipseUTime:TDateTime;end;
17193:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO );
17194:   TPlanetRec', 'record AsOf : TDateTime; Name :Str; MeanLon : '
17195:    +extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
17196:    +ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
17197:    +jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end;
17198:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended;
       UncorrectedEarthDistance:extended;GeoEclLonLat:TRpoint;CorrectedEarthDistance:extended;ApparentRaDecl:TRPoint;end;
17199:   SIRegister_TAstronomy(CL);
17200:  Func AngleToStr( angle : extended) :Str;
17201:  Func AngleToStr2( angle : extended; decimal:boolean) :Str;
17202:  Func StrToAngle( s :Str; var angle : extended) :Bool;
17203:  Func HoursToStr24( t : extended) :Str;
17204:  Func RPoint( x, y : extended) : TRPoint;
17205:  Func getStimename( t : TDTType) :Str;
17206: end;
17207:
17208: Proc SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
17209: begin
17210:   TCardValue', 'Int;
17211:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts );
17212:   TShortSuit', '( cardS, cardD, cardC, cardH );
17213: Type(TDecks,Stand1,Standrd2,Fises1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
17214:   SIRegister_TCard(CL);
17215:   SIRegister_TDeck(CL);
17216: end;
17217:
17218: Proc SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
17219: begin
17220:   tMethodCall', 'Procedure;tVerboseCall', 'Proc ( s :Str);
17221:  // PTEdge', '^TEdge // will not work;
17222:   TEdge', 'record FromNodeIndex : Int; ToNodeIndex : Int; '
17223:    +Weight : Int; work : Int; Len : Int; Highlight :Bool; end;
17224:   SIRegister_TNode(CL);
17225:   SIRegister_TGraphList(CL);
17226: end;
17227:
17228: Proc SIRegister_UParser10(CL: TPSPascalCompiler);
17229: begin
17230:   ParserFloat', 'extended;
17231:   //PParserFloat', '^ParserFloat /will not work;
17232:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
17233:    +ulo, IntDiv,IntDIVZ,Intpower,realpower,square,third,fourth,FuncOneVar,FuncTwoVar );
17234:   //POperation', '^TOperation // will not work;
17235:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
17236:    +est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
17237:    +; Token : TDFFToken; end;
17238:   TMathProcedure','procedure(AnOperation: TDFFOperation);
17239:   (TOBJECT'),'EMathParserError;
17240:   TOBJECT'),'ESyntaxError;
17241:   'TOBJECT'),'EExpressionHasBlanks;
17242:   ('TOBJECT'),'EExpressionTooComplex;
17243:   (TOBJECT'),'ETooManyNestings;
17244:   ('TOBJECT'),'EMissMatchingBracket;
17245:   (TOBJECT'),'EBadName;
17246:   (CL.FindClass('TOBJECT'),'EParserInternalError;
17247:   (TExParserExceptionEvent', 'Proc ( Sender : TObject; E : Exception);
17248:   SIRegister_TCustomParser(CL);
17249:   SIRegister_TExParser(CL);
17250: end;
```

```
17251:
17252: Func isService:Bool;
17253: begin
17254:   result:= NOT(Application is TApplication);
17255:   {result:= Application is TServiceApplication;}
17256: end;
17257: Func isApplication:Bool;
17258: begin
17259:   result:= Application is TApplication;
17260: end;
17261: //SM_REMOTESESSION = $1000
17262: Func isTerminalSession:Bool;
17263: begin
17264:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
17265: end;
17266: writeln(inttostr(getmemoryload))
17267:
17268: Proc SIRegister_cyIEUtils(CL: TPSPascalCompiler);
17269: begin
17270:  TypeS('TwbPageSetup', record font :Str; footer :Str; header : '
17271:    +String; margin_bottom :Str; margin_left :Str; margin_right : Strin'
17272:    +g; margin_top :Str; Print_Background :Str; Shrink_To_Fit:str; end;
17273:  Func cyURLEncode( const S :Str) :Str;
17274:  Func MakeResourceURL(const ModulName:str;const ResName:PChar;const ResType:PChar):str;
17275:  Func MakeResourceURL1(const Modul:HMODULE;const ResName:PChar;const ResType:PChar):str;
17276:  Func cyColorToHtml( aColor : TColor) :Str;
17277:  Func HtmlToColor( aHtmlColor :Str) : TColor;
17278:  //Func GetStreamEncoding( aStream : TStream) : TEncoding;
17279:  // Func IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) :Bool;
17280:  Func AddHtmlUnicodePrefix( aHtml :Str) :Str;
17281:  Func RemoveHtmlUnicodePrefix( aHtml :Str) :Str;
17282:  Proc GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup);
17283:  Proc SetPageSetupToRegistry( awbPageSetup : TwbPageSetup);
17284:  ConstantN('IEBodyBorderless','String').SetString( 'none;
17285:   IEBodySingleBorder','String').SetString( ';
17286:   IEDesignModeOn','String').SetString( 'On;
17287:   IEDesignModeOff','String').SetString( 'Off;
17288:   cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
17289:   cHtmlUnicodePrefixChar','Char').SetString( #$FE);
17290: end;
17291:
17292:
17293: Proc SIRegister_UcomboV2(CL: TPSPascalCompiler);
17294: begin
17295:  ConstantN('UMaxEntries','LongInt').SetInt( 600);
17296:   TypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
17297:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
17298:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
17299:    +'inationsRepeat, CombinationsRepeatDown );
17300:   TypeS('TByteArray64', 'array[0..600 + 1] of int64;;
17301:   SIRegister_TComboSet(CL);
17302: end;
17303:
17304:  Proc SIRegister_cyBaseComm(CL: TPSPascalCompiler);
17305: begin
17306:   TcyCommandType', '( ctDeveloperDefined, ctUserDefined );
17307:   TStreamContentType', '( scDeveloperDefined, scUserDefined, scString );
17308:   TProcOnReceiveCommand', 'Proc (Sender: TObject; aCommand: Word; userParam : Int);
17309:   TProcOnReceiveString', 'Proc ( Sender : TObject; fromBaseCo'
17310:    +'mmHandle : THandle; aString :Str; userParam : Int);
17311:   TProcOnReceiveMemoryStream', 'Proc ( Sender : TObject; from'
17312:    +BaseCommHandle : THandle; aStream : TMemoryStream; userParam: Int);
17313:   SIRegister_TcyBaseComm(CL);
17314:  ConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
17315:  ConstantN('MsgResultOk','LongInt').SetInt( 99);
17316:  Func ValidateFileMappingName( aName :Str) :Str;
17317:  Proc makeCaption(leftSide, Rightside:str; form:TForm);
17318: end;
17319:
17320: Proc SIRegister_cyDERUtils(CL: TPSPascalCompiler);
17321: begin
17322:   TypeS('DERString', 'String;
17323:   TypeS('DERChar', 'Char;
17324:   TypeS('TElementsType', ( etText, etExpressionKeyWord, etNumbers, etInt'
17325:    +eger,etFloat,etPercentage,etwebSite,etWebMail,etMoney,etDate,etTextLine,etParagraph);
17326:   TypeS('TElementsTypes', 'set of TElementsType;
17327:   TypeS('DERNString', 'String;
17328:  const DERDecimalSeparator','String').SetString( '.;
17329:  const DERDefaultChars,String)(+°@/%-_.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ;
17330:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz;
17331:  Func isValidWebSiteChar( aChar : Char) :Bool;
17332:  Func isValidWebMailChar( aChar : Char) :Bool;
17333:  Func isValidwebSite( aStr :Str) :Bool;
17334:  Func isValidWebMail( aStr :Str) :Bool;
17335:  Func ValidateDate( aDERStr : DERString; var RsltFormat :Str) :Bool;
17336:  Func DERStrToDate( aDERStr, aFormat :Str) : TDate;
17337:  Func IsDERChar( aChar : Char) :Bool;
17338:  Func IsDERDefaultChar( aChar : Char) :Bool;
17339:  Func IsDERMoneyChar( aChar : Char) :Bool;
```

```
17340:  Func IsDERExceptionCar( aChar : Char) :Bool;
17341:  Func IsDERSymbols( aDERString :Str) :Bool;
17342:  Func StringToDERCharSet( aStr :Str) : DERString;
17343:  Func IsDERNDefaultChar( aChar : Char) :Bool;
17344:  Func IsDERNChar( aChar : Char) :Bool;
17345:  Func DERToDERNCharset( aDERStr : DERString) : DERNString;
17346:  Func DERExtractwebSite( aDERStr : DERString; SmartRecognition :Bool) :Str;
17347:  Func DERExtractWebMail( aDERStr : DERString) :Str;
17348:  Func DERExtractPhoneNr( aDERStr : DERString) :Str;
17349:  Func DERExecute(aDERStr:DERString;SmartNumbersRecognition,SmartWebsiteRecognition:Bool):TElementsType;
17350:  Func DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInt, RsltFloat, RsltPercentage, RsltwebSite,
        RsltWebMail,RsltMoney,RsltDate:str;SmartNumbersRecognition,SmartWebsiteRecognition:Bool):TElementsType;
17351:  Func RetrieveElementValue(aStr:str;SmartNumbersRecognition,
        SmartWebsiteRecognition:Bool;aElementsType:TElementsType):str;;
17352:  Proc RetrieveElementValue1(aStr:Str;SmartNumbersRecognition,SmartWebsiteRecognition:Bool;var RsltDERStr:
        DERString;var RsltElementType:TElementsType);;
17353: end;
17354:
17355: Proc SIRegister_cyImage(CL: TPSPascalCompiler);
17356: begin
17357:   pRGBQuadArray', '^TRGBQuadArray // will not work;
17358:  Func BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Int);
17359:  Proc BitmapSetPercentBrightness(Bmp: TBitmap; IncPercent:Int; RefreshBmp :Bool);
17360:  Proc BitmapSetPixelsBrightness(Bmp:TBitmap; IncPixels : Int; RefreshBmp:Bool);
17361:  Proc BitmapSetPercentContrast(Bmp:TBitmap; IncPercent : Int; RefreshBmp:Bool);
17362:  Proc BitmapSetPixelsContrast(Bmp: TBitmap; IncPixels : Int; RefreshBmp:Bool);
17363:  Proc BitmapNegative( Bmp : TBitmap; RefreshBmp :Bool);
17364:  Proc BitmapModifyRGB(Bmp:TBitmap;IncRed:Int;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
17365:  Proc BitmapReplaceColor(Bmp:TBitmap;OldColor:TColor;NewColor:TColor;RangeRed,RangeGreen,
        RangeBlue:Word;SingleDestinationColor:Bool;RefreshBmp:Bool);
17366:  Proc BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,PercentRange1Green,
         PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
        PercentRange2Blue:Double;SingleDestinationColor:Bool;RefreshBmp:Bool;;
17367:  Proc BitmapReplaceColors(Bmp:TBitmap; Array_OldPalette,Array_NewPalette:array of
        TColor;SingleDestinationColor,RefreshBmp:Bool);
17368: Proc BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp:Bool
17369:  Proc BitmapRotate(Bmp:TBitmap;AngleDegree: Word;AdjustSize:Bool;BkColor: TColor);
17370:  Proc BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Int;RefreshBmp:Bool;
17371:  Proc GraphicMirror(Source:TGraphic;Dest:TCanvas;Left,Top:Int;Horizontal,Vertical:Bool;
17372:  Proc GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Bool;
17373:  Func BitmapCreate(BmpWidth:Int;BmpHeight:Int;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
17374:  Proc BitmapSaveToJpegFile(Bmp : TBitmap; FileName :Str; QualityPercent : Word);
17375:  Proc JpegSaveToBitmapFile(JPEG : TJPEGImage; FileName :Str);
17376: end;
17377:
17378: Proc SIRegister_WaveUtils(CL: TPSPascalCompiler);
17379: begin
17380:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA );
17381:   TPCMChannel', '( cMono, cStereo );
17382:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz );
17383:   TPCMBitsPerSample', '( bs8Bit, bs16Bit );
17384:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
17385:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
17386:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
17387:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
17388:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
17389:    +'it48000Hz, Stereo16bit48000Hz );
17390:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
17391:    +nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end;
17392:    tWaveFormatEx', 'PWaveFormatEx;  HMMIO', 'Int;
17393:   TWaveDeviceFormats', 'set of TPCMFormat;
17394:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
17395:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound );
17396:   TypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport;
17397:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate );
17398:   TWaveOutOptions', 'set of TWaveOutOption;
17399:   TStreamOwnership2', '( soReference, soOwned );
17400:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx );
17401:  // PRawWave', '^TRawWave // will not work;
17402:   TRawWave', 'record pData : TObject; dwSize : DWORD; end;
17403:   ClassN(CL.FindClass('TOBJECT'),'EWaveAudioError;
17404:   ClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError;
17405:   ClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation;
17406:   TWaveAudioEvent', 'Proc (Sender : TObject);
17407:   TWaveAudioGetFormatEvent', 'Proc (Sender : TObject; var pW'
17408:    +aveFormat : PWaveFormatEx; var FreeIt : Bool);
17409:   TWaveAudioGetDataEvent', 'Func (Sender : TObject; const Buf'
17410:    +fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD;
17411:   TWaveAudioGetDataPtrEvent', 'Func ( Sender : TObject; var Bu'
17412:    +ffer : TObject; var NumLoops : DWORD; var FreeIt :Bool) : DWORD;
17413:   TWaveAudioDataReadyEvent', 'Proc ( Sender : TObject; const '
17414:    +Buffer : TObject; BufferSize : DWORD; var FreeIt :Bool);
17415:   TWaveAudioLevelEvent', 'Proc ( Sender : TObject; Level : Int);
17416:   TWaveAudioFilterEvent,Procedure(Sender:TObject;const Buffer:TObject; BufferSize:DWORD);
17417:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
17418:  CreateWaveAudio(mmIO:HMMIO;const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
17419:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo);
17420: GetStreamWaveAudioInfo(Stream:TStream;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
17421:  CreateStreamWaveAudio(Stream:TStream;const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):HMMIO;
```

```
17422:  OpenStreamWaveAudio( Stream : TStream) : HMMIO;
17423:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD;
17424:  GetAudioFormat( FormatTag : Word) :Str;
17425:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) :Str;
17426:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD;
17427:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD;
17428:  GetWaveAudioPeakLevel(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx): Int;
17429:  InvertWaveAudio(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx):Bool;
17430:  SilenceWaveAudio(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx):Bool;
17431: ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
17432:  MixWaveAudio(const RawWaves:TRawWave;Count:Int;const
       pWaveFormat:PWaveFormatEx;Buffer:TObject;BufferSize:DWORD):Bool;
17433:  ConvertWaveFormat(const srcFormat:PWaveFormatEx;srcData:TObject;srcDataSize:DWORD;const
       dstFormat:PWaveFormatEx; var dstData:TObject;var dstDataSize:DWORD):Bool;
17434:  SetPCMAudioFormat(const pWaveFormat:PWaveFormatEx;Channels:TPCMChannel;SamplesPerSec:TPCMSamplesPerSec;
       BitsPerSample:TPCMBitsPerSample);
17435:  Proc SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat:TPCMFormat);
17436:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat;
17437:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position:DWORD):DWORD;
17438:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) :Str;
17439:  WaitForSyncObject(SyncObject : THandle; Timeout : DWORD) : DWORD;
17440:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT;
17441: end;
17442:
17443: Proc SIRegister_NamedPipes(CL: TPSPascalCompiler);
17444: begin
17445:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
17446:  ConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
17447:  'PIPE_NAMING_SCHEME','String).SetString( '\\%s\pipe\%s;
17448:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
17449:  WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
17450:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
17451:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
17452:   TypeS('TPipeDirection', ( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient );
17453:   TypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg );
17454:   TypeS('TOverlappedResult', '(ov_Failed, ov_Pending, ov_MoreData, ov_Complete );
17455:   SIRegister_TNamedPipe(CL);
17456:   SIRegister_TServerPipe(CL);
17457:   SIRegister_TClientPipe(CL);
17458:  Func CalculateTimeout( aBasis : DWORD) : DWORD;
17459:  Func HasOverlappedIoCompleted( const ov : OVERLAPPED) :Bool;
17460:  Func GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var
       dwBytes:DWORD;bWait:Bool):TOverlappedResult;
17461:  Func GetStreamAsText( stm : TStream) :Str;
17462:  Proc SetStreamAsText(const aTxt:Str; stm : TStream);
17463: end;
17464:
17465: Proc SIRegister_DPUtils(CL: TPSPascalCompiler);
17466: begin
17467:  // TypeS('pRGBTripleArray', '^TRGBTripleArray // will not work;
17468:   SIRegister_TThumbData(CL);
17469:  'PIC_BMP','LongInt').SetInt( 0);
17470:  'PIC_JPG','LongInt').SetInt( 1);
17471:  'THUMB_WIDTH','LongInt').SetInt( 60);
17472:  'THUMB_HEIGHT','LongInt').SetInt( 60);
17473:  Func IsEqualFile(Filename:str; Size:Int; LastWriteTime : TDateTime):Bool;
17474:  Proc GetFileInfo(Filename:str; var Size : Int; var LastWriteTime : TDateTime);
17475:  Func ReadBitmap( Filehandle : Int; Width, Height : Int) : TBitmap;
17476:  Proc WriteBitmap( Filehandle : Int; bmp : TBitmap);
17477:  Func OpenPicture( fn :Str; var tp : Int) : Int;
17478:  Func ConvertPicture( pi : Int; tp : Int) : TBitmap;
17479:  Func LoadPicture( fn :Str; var w, h : Int) : TBitmap;
17480:  Func TurnBitmap( bmp : TBitmap; ang : Int) : TBitmap;
17481:  Func RotateBitmap( Bitmap : TBitmap; Direction : Int) : TBitmap;
17482:  Func StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect;
17483:  Func ThumbBitmap( Bitmap : TBitmap; Width, Height : Int) : TBitmap;
17484:  Proc ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Int);
17485:  Proc FindFiles( path, mask :Str; items : TStringList);
17486:  Func LetFileName( s :Str) :Str;
17487:  Func LetParentPath( path :Str) :Str;
17488:  Func AddBackSlash( path :Str) :Str;
17489:  Func CutBackSlash( path :Str) :Str;
17490: end;
17491:
17492: Proc SIRegister_CommonTools(CL: TPSPascalCompiler);
17493: begin
17494:  //'BYTES','LongInt').SetInt( 1);
17495:  'KBYTES','LongInt').SetInt( 1024);
17496:  'DBG_ALIVE','LongWord').SetUInt( Int ( $11BABE11 ));
17497:  'DBG_DESTROYING','LongWord').SetUInt( Int ( $44FADE44 ));
17498:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
17499:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D};
17500:   SIRegister_MakeComServerMethodsPublic(CL);
17501:   TypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application );
17502:  Func IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) :Bool;
17503:  Proc TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet :Bool);
17504:  Func TBGetTempFolder :Str;
17505:  Func TBGetTempFile :Str;
17506:  Func TBGetModuleFilename :Str;
```

```
17507:  Func FormatModuleVersionInfo( const aFilename :Str) :Str;
17508:  Func GetVersionInfoString( const aFile, aEntry :Str; aLang : WORD) :Str;
17509:  Func TBGetFileSize( aFile :Str; aMultipleOf : Int) : Int;
17510:  Func FormatAttribString( aAttr : Int) :Str;
17511:  Func GetSomeFileInfo( aFile :Str; aWhatInfo : TSomeFileInfo) :Str;
17512:  Func ShellRecycle( aWnd : HWND; aFileOrFolder :Str) :Bool;
17513:  Func IsDebuggerPresent : BOOL;
17514:  Func TBNotImplemented : HRESULT;
17515: end;
17516:
17517: Proc SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
17518: begin
17519:   //TypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA;
17520:   //TypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA;
17521:   TypeS('TOSVersionInfoEx', 'TOSVersionInfo;
17522:   TypeS('TDrivesProperty', 'array[1..26] of boolean;;
17523:   //TDrivesProperty = array['A'..'Z'] of boolean;
17524:  Func TBSetSystemTime( DateTime : TDateTime; DOW : word) :Bool;
17525:  Func IsElevated :Bool;
17526: Proc CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
17527:   TypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined );
17528:  Func TrimNetResource( UNC :Str) :Str;
17529:  Proc GetFreeDrives( var FreeDrives : TDrivesProperty);
17530:  Proc GetMappedDrives( var MappedDrives : TDrivesProperty);
17531:  Func MapDrive(UNCPath:str;Drive:char;PasswordUsage:TPasswordUsage;Password:str;
       UserUsage:TPasswordUsage;User:str;Comment:str):bool;
17532:  Func UnmapDrive( Drive : char; Force :Bool):Bool;
17533:  Func TBIsWindowsVista :Bool;
17534:  Proc SetVistaFonts( const AForm : TForm);
17535:  Proc SetVistaContentFonts( const AFont : TFont);
17536:  Func GetProductType( var sType :Str) :Bool;
17537:  Func lstrcmp( lpString1, lpString2 : PChar) : Int;
17538:  Func lstrcmpi( lpString1, lpString2 : PChar) : Int;
17539:  Func lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Int) : PChar;
17540:  Func lstrcpy( lpString1, lpString2 : PChar) : PChar;
17541:  Func lstrcat( lpString1, lpString2 : PChar) : PChar;
17542:  Func lstrlen( lpString : PChar) : Int;
17543:  Func
       GetTokenInformation(TokenHandle:THandle;TokenInformationClass:TTokenInformationClass;TokenInformation:___Pointer;Token
       ReturnLength:DWORD):BOOL;
17544:  Func
       SetTokenInformation(TokenHandle:THandle;TokenInformationClass:TTokenInformationClass;TokenInformation:___Pointer;Token
17545: end;
17546:
17547: Proc SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
17548: begin
17549:  'cLineTerminator','Char').SetString( #10);
17550:  'cLineTerminators','String').SetString( #13#10);
17551:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
17552:   SIRegister_TFixedCriticalSection(CL);
17553:   SIRegister_TMultiReadSingleWrite(CL);
17554:  Proc SetDecimalSeparator( c : Char);
17555:  Func GetDecimalSeparator : Char;
17556:  Func GetDecimalSeparator : Char;
17557:  Proc SetDateSeparator( c : Char);
17558:  Func GetDateSeparator : Char;
17559:  Proc setTimeSeparator( c : Char);
17560:  Func getTimeSeparator : Char;
17561:  Proc setShortDateFormat( c :Str);
17562:  Func getShortDateFormat :Str;
17563:  Proc setlongDateFormat( c :Str);
17564:  Func getlongDateFormat :Str;
17565:  Proc setShorttimeFormat( c :Str);
17566:  Func getShorttimeFormat :Str;
17567:  Proc setlongtimeFormat( c :Str);
17568:  Func getlongtimeFormat :Str;
17569:   //getTimeSeparator
17570:  Proc setThousandSeparator(c : Char);
17571:  Func getThousandSeparator : Char;
17572:  Proc setListSeparator( c : Char);
17573:  Func getListSeparator : Char;
17574:  'LOGON32_LOGON_INTERACTIVE','LongInt').SetInt( 2);
17575:  'LOGON32_LOGON_NETWORK','LongInt').SetInt( 3);
17576:  'LOGON32_LOGON_BATCH','LongInt').SetInt( 4);
17577:  'LOGON32_LOGON_SERVICE','LongInt').SetInt( 5);
17578:  'LOGON32_PROVIDER_DEFAULT','LongInt').SetInt( 0);
17579:  'LOGON32_PROVIDER_WINNT35','LongInt').SetInt( 1);
17580:  'LOGON32_PROVIDER_WINNT40','LongInt').SetInt( 2);
17581:  'LOGON32_PROVIDER_WINNT50','LongInt').SetInt( 3);
17582:
17583:  TCollectFileProgressEvent', 'Proc ( const directory:Str; var skipScan:Bool);
17584:  Proc CollectFiles(const directory,fileMask:UnicodeString; list:TStrings; recurseSubdirectories:Boolean;
       onProgress : TCollectFileProgressEvent);
17585:   TypeS('NativeInt', 'Int;
17586:   //TypeS('PNativeInt', '^NativeInt // will not work;
17587:   TypeS('NativeUInt', 'Cardinal;
17588:   //TypeS('PNativeUInt', '^NativeUInt // will not work;
17589:   //TypeS('TBytes', 'array of Byte;
17590:   TypeS('RawByteString', 'UnicodeString;
```

```
17591:   //TypeS('PNativeInt', '^NativeInt // will not work;
17592:   //TypeS('PUInt64', '^UInt64 // will not work;
17593:   SIRegister_TPath(CL);
17594:   SIRegister_TFile(CL);
17595:   SIRegister_TdwsThread(CL);
17596:  Func GetSystemMilliseconds : Int64;
17597:  Func UTCDateTime : TDateTime;
17598:  Func UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString;
17599:  Func UnicodeCompareStr( const S1, S2 : UnicodeString) : Int;
17600:  Func dwsAnsiCompareText( const S1, S2 : UnicodeString) : Int;
17601:  Func dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Int;
17602:  Func UnicodeComparePChars(p1:PChar; n1 : Int; p2 : PChar; n2 : Int): Int;;
17603:  Func UnicodeComparePChars1( p1, p2 : PChar; n : Int) : Int;;
17604:  Func UnicodeLowerCase( const s : UnicodeString) : UnicodeString;
17605:  Func UnicodeUpperCase( const s : UnicodeString) : UnicodeString;
17606:  Func ASCIICompareText( const s1, s2 : UnicodeString) : Int;
17607:  Func ASCIISameText( const s1, s2 : UnicodeString) :Bool;
17608:  Func InterlockedIncrement( var val : Int) : Int;
17609:  Func InterlockedDecrement( var val : Int) : Int;
17610:  Proc FastInterlockedIncrement( var val : Int);
17611:  Proc FastInterlockedDecrement( var val : Int);
17612:  Func InterlockedExchangePointer(var target:___Pointer; val:___Pointer):___Pointer;
17613:  Proc SetThreadName( const threadName : Char; threadID :Card);
17614:  Proc dwsOutputDebugString( const msg : UnicodeString);
17615: Proc WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeStr;const logRawData:Str)
17616: Func TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
17617:  Proc VarCopy( out dest : Variant; const src : Variant);
17618:  Func VarToUnicodeStr( const v : Variant) : UnicodeString;
17619:  Func LoadTextFromBuffer( const buf : TBytes) : UnicodeString;
17620:  Func LoadTextFromStream( aStream : TStream) : UnicodeString;
17621:  Func LoadTextFromFile( const fileName : UnicodeString) : UnicodeString;
17622:  Proc SaveTextToUTF8File( const fileName, text : UnicodeString);
17623:  Func OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle;
17624:  Func OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle;
17625:  Proc CloseFileHandle( hFile : THandle);
17626:  Func FileCopy(const existing, new : UnicodeString; failIfExists :Bool):Boolean;
17627:  Func FileMove( const existing, new : UnicodeString) :Bool;
17628:  Func dwsFileDelete( const fileName :Str) :Bool;
17629:  Func FileRename( const oldName, newName :Str) :Bool;
17630:  Func dwsFileSize( const name :Str) : Int64;
17631:  Func dwsFileDateTime( const name :Str) : TDateTime;
17632:  Func DirectSet8087CW( newValue : Word) : Word;
17633:  Func DirectSetMXCSR( newValue : Word) : Word;
17634:  Func TtoObject( const T: byte) : TObject;
17635:  Func TtoPointer( const T: byte) : ___Pointer;
17636:  Proc GetMemForT(var T: byte; Size : Int);
17637:  Func FindDelimiter( const Delimiters, S :Str; StartIdx : Int) : Int;
17638: end;
17639:
17640: Proc SIRegister_AdSocket(CL: TPSPascalCompiler);
17641: begin
17642:  'IPStrSize','LongInt').SetInt( 15);
17643:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17644:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
17645:  'ADWSBASE','LongInt').SetInt( 9000);
17646:   TypeS('TCMAPDSocketMessage', 'record Msg :Card; Socket : TSocket; '
17647:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end;
17648:   SIRegister_EApdSocketException(CL);
17649:   TWsMode', '( wsClient, wsServer );
17650:   TWsNotifyEvent', 'Proc ( Sender : TObject; Socket : TSocket;
17651:   TWsSocketErrorEvent', 'Proc ( Sender : TObject; Socket : TSocket;ErrCode : Int);
17652:   SIRegister_TApdSocket(CL);
17653: end;
17654:
17655: Proc SIRegister_AdPort(CL: TPSPascalCompiler);
17656: begin
17657:   SIRegister_TApdCustomComPort(CL);
17658:   SIRegister_TApdComPort(CL);
17659:  Func ComName( const ComNumber : Word) : ShortString;
17660:  Func SearchComPort( const C : TComponent) : TApdCustomComPort;
17661: end;
17662:
17663: Proc SIRegister_PathFunc(CL: TPSPascalCompiler);
17664: begin
17665:  Func inAddBackslash( const S :Str) :Str;
17666:  Func PathChangeExt( const Filename, Extension :Str) :Str;
17667:  Func PathCharCompare( const S1, S2 : PChar) :Bool;
17668:  Func PathCharIsSlash( const C : Char) :Bool;
17669:  Func PathCharIsTrailByte( const S :Str; const Index : Int) :Bool;
17670:  Func PathCharLength( const S :Str; const Index : Int) : Int;
17671:  Func inPathCombine( const Dir, Filename :Str) :Str;
17672:  Func PathCompare( const S1, S2 :Str) : Int;
17673:  Func PathDrivePartLength( const Filename :Str) : Int;
17674:  Func PathDrivePartLengthEx(const Filename:str;const IncludeSignificantSlash:Bool):Int;
17675:  Func inPathExpand( const Filename :Str) :Str;
17676:  Func PathExtensionPos( const Filename :Str) : Int;
17677:  Func PathExtractDir( const Filename :Str) :Str;
17678:  Func PathExtractDrive( const Filename :Str) :Str;
17679:  Func PathExtractExt( const Filename :Str) :Str;
```

```
17680:  Func PathExtractName( const Filename :Str) :Str;
17681:  Func PathExtractPath( const Filename :Str) :Str;
17682:  Func PathIsRooted( const Filename :Str) :Bool;
17683:  Func PathLastChar( const S :Str) : PChar;
17684:  Func PathLastDelimiter( const Delimiters, S :Str) : Int;
17685:  Func PathLowercase( const S :Str) :Str;
17686:  Func PathNormalizeSlashes( const S :Str) :Str;
17687:  Func PathPathPartLength(const Filename:str;const IncludeSlashsAfterPath:Bool):Int;
17688:  Func PathPos( Ch : Char; const S :Str) : Int;
17689:  Func PathStartsWith( const S, AStartsWith :Str) :Bool;
17690:  Func PathStrNextChar( const S : PChar) : PChar;
17691:  Func PathStrPrevChar( const Start, Current : PChar) : PChar;
17692:  Func PathStrScan( const S : PChar; const C : Char) : PChar;
17693:  Func inRemoveBackslash( const S :Str) :Str;
17694:  Func RemoveBackslashUnlessRoot( const S :Str) :Str;
17695:  Proc PathFuncRunTests( const AlsoTestJapaneseDBCS :Bool);
17696: end;
17697:
17698: Proc SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17699: begin
17700:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion;
17701:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer;
17702:  NEWREGSTR_PATH_SPECIAL_FOLDERS,String').SetStr(NEWREGSTR_PATH_EXPLORER +'\Shell Folders;
17703:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall;
17704:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName;
17705:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString;
17706:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17707:   //TypeS('PLeadByteSet', '^TLeadByteSet // will not work;
17708:   TypeS('TLeadByteSet', 'set of Char;
17709:   SIRegister_TOneShotTimer(CL);
17710:   TypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit );
17711:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17712:  Func NewFileExists( const Name :Str) :Bool;
17713:  Func inDirExists( const Name :Str) :Bool;
17714:  Func FileOrDirExists( const Name :Str) :Bool;
17715:  Func IsDirectoryAndNotReparsePoint( const Name :Str) :Bool;
17716:  Func GetIniString(const Section,Key:str; Default:str;const Filename:str):str;
17717:  Func GetIniInt(const Section,Key:Str;const Default,Min,Max:Longint;const Filenam:Str):Longint;
17718:  Func GetIniBool(const Section,Key:str;const Default:Bool;const Filename:str):Bool;
17719:  Func IniKeyExists( const Section, Key, Filename :Str) :Bool;
17720:  Func IsIniSectionEmpty( const Section, Filename :Str) :Bool;
17721:  Func SetIniString( const Section, Key, Value, Filename :Str) :Bool;
17722:  Func SetIniInt(const Section,Key:str;const Value:Longint;const Filename:str):Bool;
17723:  Func SetIniBool(const Section,Key:str;const Value:Bool;const Filename:Str):Bool;
17724:  Proc DeleteIniEntry( const Section, Key, Filename :Str);
17725:  Proc DeleteIniSection( const Section, Filename :Str);
17726:  Func GetEnv( const EnvVar :Str) :Str;
17727:  Func GetCmdTail :Str;
17728:  Func GetCmdTailEx( StartIndex : Int) :Str;
17729:  Func NewParamCount : Int;
17730:  Func NewParamStr( Index : Int) :Str;
17731:  Func AddQuotes( const S :Str) :Str;
17732:  Func RemoveQuotes( const S :Str) :Str;
17733:  Func inGetShortName( const LongName :Str) :Str;
17734:  Func inGetWinDir :Str;
17735:  Func inGetSystemDir :Str;
17736:  Func GetSysWow64Dir :Str;
17737:  Func GetSysNativeDir( const IsWin64 :Bool) :Str;
17738:  Func inGetTempDir :Str;
17739:  Func StringChange( var S :Str; const FromStr, ToStr :Str) : Int;
17740:  Func StringChangeEx(var S:str;const FromStr,ToStr:str;const SupportDBCS:Bool):Int;
17741:  Func AdjustLength( var S :Str; const Res :Card) :Bool;
17742:  Func UsingWinNT :Bool;
17743:  Func ConvertConstPercentStr( var S :Str) :Bool;
17744:  Func ConvertPercentStr( var S :Str) :Bool;
17745:  Func ConstPos( const Ch : Char; const S :Str) : Int;
17746:  Func SkipPastConst( const S :Str; const Start : Int) : Int;
17747:  Func RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr :Str) :Bool;
17748:  Func RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr:str):Bool;
17749:  Func RegValueExists( H : HKEY; Name : PChar) :Bool;
17750:  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;
17751:  Func RegOpenKeyExView(const
        RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17752:  Func RegDeleteKeyView( const RegView:TRegView;const Key:HKEY;const Name:PChar): Longint;
17753:  Func RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17754:  Func RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyNam:PChar):Longint;
17755:  Func GetShellFolderPath( const FolderID : Int) :Str;
17756:  Func IsAdminLoggedOn :Bool;
17757:  Func IsPowerUserLoggedOn :Bool;
17758:  Func IsMultiByteString( const S : Ansistr) :Bool;
17759:  Func FontExists( const FaceName :Str) :Bool;
17760:  //Proc FreeAndNil( var Obj);
17761:  Func SafeLoadLibrary( const Filename :Str; ErrorMode : UINT) : HMODULE;
17762:  Func GetUILanguage : LANGID;
17763:  Func RemoveAccelChar( const S :Str) :Str;
17764:  Func GetTextWidth(const DC : HDC; S :Str; const Prefix:Boolean):Int;
17765:  Func AddPeriod( const S :Str) :Str;
```

```
17766:  Func GetExceptMessage :Str;
17767:  Func GetPreferredUIFont :Str;
17768:  Func IsWildcard( const Pattern :Str) :Bool;
17769:  Func WildcardMatch( const Text, Pattern : PChar) :Bool;
17770:  Func IntMax( const A, B : Int) : Int;
17771:  Func Win32ErrorString( ErrorCode : Int) :Str;
17772:  Proc GetLeadBytes( var ALeadBytes : TLeadByteSet);
17773:  Func inCompareMem( P1, P2 : TObject; Length : Int) :Bool;
17774:  Func DeleteDirTree( const Dir :Str) :Bool;
17775:  Func SetNTFSCompression(const FileOrDir:Str; Compress :Bool) :Bool;
17776:  Proc AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT);
17777:  // TypeS('TSysCharSet', 'set of AnsiChar;
17778:  Func inCharInSet( C : Char; const CharSet : TSysCharSet) :Bool;
17779:  Func ShutdownBlockReasonCreate( Wnd : HWND; const Reason :Str) :Bool;
17780:  Func ShutdownBlockReasonDestroy( Wnd : HWND) :Bool;
17781:  Func TryStrToBoolean( const S :Str; var BoolResult :Bool) :Bool;
17782:  Proc WaitMessageWithTimeout( const Milliseconds : DWORD);
17783:  Func MoveFileReplace(const ExistingFileName, NewFileName :Str) :Bool;
17784:  Proc TryEnableAutoCompleteFileSystem( Wnd : HWND);
17785: end;
17786:
17787: Proc SIRegister_CmnFunc(CL: TPSPascalCompiler);
17788: begin
17789:   SIRegister_TWindowDisabler(CL);
17790:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError );
17791:   TMsgBoxCallbackFunc,procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17792:  Proc UpdateHorizontalExtent( const ListBox : TCustomListBox);
17793:  Func MinimizePathName(const Filename:str; const Font:TFont;MaxLen:Int):str;
17794:  Func AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Int;
17795:  Func MsgBoxP(const Text,Caption:PChar;const Typ: TMsgBoxType;const Buttons:Card):Int;
17796:  Func inMsgBox(const Text,Caption:str;const Typ:TMsgBoxType;const Buttons:Card):Int;
17797:  Func MsgBoxFmt(const Text:str;const Args:array of const;const Caption:str;const Typ:TMsgBoxType;const
        Buttons:Card):Int;
17798:  Proc ReactivateTopWindow;
17799:  Proc SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar);
17800:  Proc SetMessageBoxRightToLeft( const ARightToLeft :Bool);
17801:  Proc SetMessageBoxCallbakFunc(const AFunc:TMsgBoxCallbackFunc;const AParam:LongInt);
17802: end;
17803:
17804: Proc SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17805: begin
17806:   SIRegister_TImageGrabber(CL);
17807:   SIRegister_TCaptureDrivers(CL);
17808:   SIRegister_TCaptureDriver(CL);
17809: end;
17810:
17811: Proc SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17812: begin
17813:  Func GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:str;const
        Entries:TGrantPermissionEntry; const EntryCount:Int):Boolean;
17814:  Func GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:str;const Entries:
        TGrantPermissionEntry; const EntryCount:Int):Boolean;
17815: end;
17816:
17817: Proc SIRegister_RedirFunc(CL: TPSPascalCompiler);
17818: begin
17819:   TypeS(TPreviousFsRedirectionState,record DidDisable:Boolean;OldValue: ___Pointer;end;
17820:  Func AreFsRedirectionFunctionsAvailable :Bool;
17821:  Func DisableFsRedirectionIf(const Disable:Bool;var PreviousState:TPreviousFsRedirectionState):Boo;
17822:  Proc RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState);
17823:  Func CreateDirectoryRedir(const DisableFsRedir:Boolean; const Filename:Str) : BOOL;
17824:  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;
17825:  Func CopyFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:str;const
        FailIfExists:BOOL):BOOL;
17826:  Func DeleteFileRedir(const DisableFsRedir:Boolean; const Filename:Str): BOOL;
17827:  Func DirExistsRedir(const DisableFsRedir:Boolean; const Filename:Str):Bool;
17828:  Func FileOrDirExistsRedir( const DisableFsRedir:Bool; const Filename:str):Bool;
17829:  Func FindFirstFileRedir(const DisableFsRedir:Boolean;const Filename:str;var
        FindData:TWin32FindData):THandle;
17830:  Func GetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:str): DWORD;
17831:  Func GetShortNameRedir(const DisableFsRedir:Boolean; const Filename:str):str;
17832:  Func GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:str;var
        VersionNumbers:TFileVersionNumbers): Bool;
17833: Func IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Bool;const Name:str):Bool;
17834: Func MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilenme,NewFilename:str):BOOL;
17835:  Func MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilen,NewFilename:str;const
        Flags:DWORD):BOOL;
17836:  Func NewFileExistsRedir(const DisableFsRedir:Boolean;const Filename:str):Bool;
17837:  Func RemoveDirectoryRedir(const DisableFsRedir:Boolean;const Filename :Str) : BOOL;
17838:  Func SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:str;const Attrib:DWORD):BOOL;
17839: Func SetNTFSCompressionRedir(const DisableFsRedir:Bool;const FileOrDir:Str;Compress:Bool:Bool;
17840:   SIRegister_TFileRedir(CL);
17841:   SIRegister_TTextFileReaderRedir(CL);
17842:   SIRegister_TTextFileWriterRedir(CL);
17843: end;
```

```
17844:
17845: Proc SIRegister_Int64Em(CL: TPSPascalCompiler);
17846: begin
17847:   //TypeS('LongWord', 'Cardinal;
17848:   TypeS('Int64', 'record Lo : LongWord; Hi : LongWord; end;
17849:  Func Compare64( const N1, N2 : Int64) : Int;
17850:  Proc Dec64( var X : Int64; N : LongWord);
17851:  Proc Dec6464( var X : Int64; const N : Int64);
17852:  Func Div64( var X : Int64; const Divisor : LongWord) : LongWord;
17853:  Func Inc64( var X : Int64; N : LongWord) :Bool;
17854:  Func Inc6464( var X : Int64; const N : Int64) :Bool;
17855:  Func Int64ToStr( X : Int64) :Str;
17856:  Func Mod64( const X : Int64; const Divisor : LongWord) : LongWord;
17857:  Func Mul64( var X : Int64; N : LongWord) :Bool;
17858:  Proc Multiply32x32to64( N1, N2 : LongWord; var X : Int64);
17859:  Proc Shr64( var X : Int64; Count : LongWord);
17860:  Func StrToInt64( const S :Str; var X : Int64) :Bool;
17861: end;
17862:
17863: Proc SIRegister_InstFunc(CL: TPSPascalCompiler);
17864: begin
17865:   //TypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work;
17866:   SIRegister_TSimpleStringList(CL);
17867:   TypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle);
17868:   TypeS('TDetermineDefaultLanguageResult','(ddNoMatch,ddMatch,ddMatchLangParameter );
17869:   TypeS('TMD5Digest', 'array[0..15] of Byte;;
17870:   TypeS('TSHA1Digest', 'array[0..19] of Byte;;
17871:  // TMD5Digest = array[0..15] of Byte; TSHA1Digest = array[0..19] of Byte;
17872:  Func CheckForMutexes( Mutexes :Str) :Bool;
17873:  Func CreateTempDir :Str;
17874:  Func DecrementSharedCount(const RegView:TRegView;const Filename:str):Bool;
17875:  Proc DelayDeleteFile( const DisableFsRedir :Bool; const Filename :Str; const MaxTries, FirstRetryDelayMS,
       SubsequentRetryDelayMS : Int);
17876:  //Func DelTree( const DisableFsRedir :Bool; const Path :Str; const IsDir, DeleteFiles, DeleteSubdirsAlso,
       BreakOnError :Bool; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc : TDeleteFileProc; const
       Param : Pointer) :Bool;
17877:  //Func DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
       TSetupLanguageDetectionMethod;const LangParameter:str;var ResultIndex:Int):
       TDetermineDefaultLanguageResult;
17878:  //Proc EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17879:  Func GenerateNonRandomUniqueFilename(Path:Str; var Filename:Str):Bool;
17880:  Func GenerateUniqueName(const DisableFsRedir:Bool;Path:Str;const Extension:str):str;
17881:  Func GetComputerNameString :Str;
17882: Func GetFileDateTime(const DisableFsRedir:Bool;const Filename:Str;var DateTime:TFileTime):Bool;
17883:  Func GetMD5OfFile(const DisableFsRedir:Boolean; const Filename :Str): TMD5Digest;
17884:  Func GetMD5OfAnsistr( const S : Ansistr) : TMD5Digest;
17885: // Func GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest;
17886:  Func GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:str):TSHA1Digest;
17887:  Func GetSHA1OfAnsistr( const S : Ansistr) : TSHA1Digest;
17888: // Func GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest;
17889:  Func GetRegRootKeyName( const RootKey : HKEY) :Str;
17890:  Func GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:str;var FreeBytes,TotBytes:Int64):Bool;
17891:  Func GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:str;var FreeBytes,
       TotalBytes:Int64):Bool;
17892:  Func GetUserNameString :Str;
17893:  Proc IncrementSharedCount(const RegView:TRegView;const Filename:str;const AlreadyExisted:Bool;
17894:  //Func InstExec( const DisableFsRedir :Bool; const Filename, Params :Str; WorkingDir :Str; const
       Wait:TExecWait;const ShowCmd:Int;const ProcessMessagesProc:TProcedure;var ResultCode:Int) :Bool;
17895: // Func InstShellExec( const Verb, Filename, Params :Str; WorkingDir :Str; const Wait:TExecWait;const
       ShowCmd:Int;const ProcessMessagesProc:TProcedure;var ResultCode:Int):Bool;
17896:  Proc InternalError( const Id :Str);
17897:  Proc InternalErrorFmt( const S :Str; const Args : array of const);
17898:  Func IsDirEmpty( const DisableFsRedir :Bool; const Dir :Str) :Bool;
17899:  Func IsProtectedSystemFile(const DisableFsRedir:Boolean;const Filename:str):Bool;
17900:  Func MakePendingFileRenameOperationsChecksum : TMD5Digest;
17901:  Func ModifyPifFile( const Filename :Str; const CloseOnExit :Bool) :Bool;
17902:  Proc RaiseFunctionFailedError( const FunctionName :Str);
17903:  Proc RaiseOleError( const FunctionName :Str; const ResultCode : HRESULT);
17904:  Proc RefreshEnvironment;
17905:  Func ReplaceSystemDirWithSysWow64( const Path :Str) :Str;
17906:  Func ReplaceSystemDirWithSysNative( Path :Str; const IsWin64 :Bool) :Str;
17907:  Proc UnregisterFont( const FontName, FontFilename :Str);
17908:  Func RestartComputer :Bool;
17909:  Proc RestartReplace( const DisableFsRedir :Bool; TempFile, DestFile :Str);
17910:  Proc SplitNewParamStr( const Index : Int; var AName, AValue :Str);
17911:  Proc Win32ErrorMsg( const FunctionName :Str);
17912:  Proc Win32ErrorMsgEx( const FunctionName :Str; const ErrorCode : DWORD);
17913:  Func inForceDirectories(const DisableFsRedir:Boolean;Dir:str):Bool;
17914:  //from Func2
17915:  //Func inCreateShellLink(const Filename,Description,ShortcutTo,Parameters, WorkingDir:Str;
       IconFilename:Str; const IconIndex, ShowCmd : Int; const HotKey : Word; FolderShortcut :Bool; '
17916:   //+'const AppUserModelID:Str;const ExcludeFromShowInNewInstall,PreventPinning:Bool):Str;
17917:  Proc RegisterTypeLibrary( const Filename :Str);
17918:  //Proc UnregisterTypeLibrary( const Filename :Str);
17919:  //Func UnpinShellLink( const Filename :Str) :Bool;}
17920:  Func getVersionInfoEx3: TOSVersionInfoEx;;
17921:  Func GetVersionEx3(out verinfo: TOSVersionInfoEx):Bool;;
17922:  Proc InitOle;;
17923:  Func ExpandConst( const S :Str) :Str;
```

```
17924:  Func ExpandConstEx( const S :Str; const CustomConsts : array of String) :Str;
17925:  Func ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndivdConst:Bool):Str;
17926:  Func ExpandConstIfPrefixed( const S :Str) :Str;
17927:  Proc LogWindowsVersion;
17928:  Func EvalCheck( const Expression :Str) :Bool;
17929: end;
17930:
17931: Proc SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17932: begin
17933: ClassN(CL.FindClass('TOBJECT'),'TResourceDetails;
17934: //TypeS('TResourceDetailsClass', 'class of TResourceDetails;
17935:  SIRegister_TResourceModule(CL);
17936:  SIRegister_TResourceDetails(CL);
17937:  SIRegister_TAnsiResourceDetails(CL);
17938:  SIRegister_TUnicodeResourceDetails(CL);
17939:  Proc RegisterResourceDetails( resourceClass : TResourceDetailsClass);
17940:  Proc UnRegisterResourceDetails( resourceClass : TResourceDetailsClass);
17941:  Func ResourceWideCharToStr( var wstr : PWideChar; codePage : Int) :Str;
17942:  Proc ResourceStrToWideChar( const s :Str; var p : PWideChar; codePage : Int);
17943:  Func ResourceNameToInt( const s :Str) : Int;
17944:  Func CompareDetails( p1, p2 : TObject(Pointer)) : Int;
17945: end;
17946:
17947:
17948: Proc SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17949: begin
17950:  //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17951:   with ClassN(CL.FindClass('TObject'),TSimpleComPort') do begin
17952:     RegisterMethod(Constructor Create;
17953:     RegisterMethod(Proc Free;
17954:     RegisterMethod(Proc Open( PortNumber : Int; const Parameters:str);
17955:     RegisterMethod(Proc WriteString( const S :Str);
17956:     RegisterMethod(Proc ReadString( var S :Str);
17957:   end;
17958:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17959:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1;
17960:       SimpleComPort.WriteString(AsciiChar);
17961: end;
17962:
17963: Proc SIRegister_Console(CL: TPSPascalCompiler);
17964: begin
17965:  ConstantN('White','LongInt').SetInt( 15); }
17966: // ConstantN('Blink','LongInt').SetInt( 128);
17967:  ('conBW40','LongInt').SetInt( 0);
17968:  ('conCO40','LongInt').SetInt( 1);
17969:  ('conBW80','LongInt').SetInt( 2);
17970:  ('conCO80','LongInt').SetInt( 3);
17971:  ('conMono','LongInt').SetInt( 7);
17972:  ConstantN('conFont8x8','LongInt').SetInt( 256);
17973:  //ConstantN('C40','').SetString( CO40);
17974:  //ConstantN('C80','').SetString( CO80);
17975:  Func conReadKey : Char;
17976:  Func conKeyPressed :Bool;
17977:  Proc conGotoXY( X, Y : Smallint);
17978:  Func conWhereX : Int;
17979:  Func conWhereY : Int;
17980:  Proc conTextColor( Color : Byte);;
17981:  Func conTextColor1 : Byte;;
17982:  Proc conTextBackground( Color : Byte);;
17983:  Func conTextBackground1 : Byte;;
17984:  Proc conTextMode( Mode : Word);
17985:  Proc conLowVideo;
17986:  Proc conHighVideo;
17987:  Proc conNormVideo;
17988:  Proc conClrScr;
17989:  Proc conClrEol;
17990:  Proc conInsLine;
17991:  Proc conDelLine;
17992:  Proc conWindow( Left, Top, Right, Bottom : Int);
17993:  Func conScreenWidth : Smallint;
17994:  Func conScreenHeight : Smallint;
17995:  Func conBufferWidth : Smallint;
17996:  Func conBufferHeight : Smallint;
17997:  Proc InitScreenMode;;
17998: end;
17999:
18000: (*----------------------------------------------------------------------------*)
18001: Proc SIRegister_testutils(CL: TPSPascalCompiler);
18002: begin
18003:   SIRegister_TNoRefCountObject(CL);
18004:  Proc FreeObjects( List : TFPList);
18005:  Proc GetMethodList( AObject : TObject; AList : TStrings);;
18006:  Proc GetMethodList1( AClass : TClass; AList : TStrings);;
18007: end;
18008:
18009: Proc SIRegister_ToolsUnit(CL: TPSPascalCompiler);
18010: begin
18011:  'MaxDataSet','LongInt').SetInt( 35);
18012:   //TypeS('TDBConnectorClass', 'class of TDBConnector;
```

```
18013:   SIRegister_TDBConnector(CL);
18014:   SIRegister_TDBBasicsTestSetup(CL);
18015:   SIRegister_TTestDataLink(CL);
18016:  'testValuesCount',LongInt).SetInt( 25);
18017:  Proc InitialiseDBConnector;
18018:  Proc FreeDBConnector;
18019:  Func DateTimeToTimeString( d : tdatetime) :Str;
18020:  Func TimeStringToDateTime( d :Str) : TDateTime;
18021: end;
18022:
18023: Proc SIRegister_fpcunit(CL: TPSPascalCompiler);
18024: begin
18025:   SIRegister_EAssertionFailedError(CL);
18026:   TypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing );
18027:   TypeS('TRunMethod', 'Procedure;
18028:   ClassN(CL.FindClass('TOBJECT'),'TTestResult;
18029:   SIRegister_TTest(CL);
18030:   SIRegister_TAssert(CL);
18031:   SIRegister_TTestFailure(CL);
18032:   SIRegister_ITestListener(CL);
18033:   SIRegister_TTestCase(CL);
18034:   //TypeS('TTestCaseClass', 'class of TTestCase;
18035:   SIRegister_TTestSuite(CL);
18036:   SIRegister_TTestResult(CL);
18037:  Func ComparisonMsg( const aExpected:Str; const aActual :Str) :Str;
18038: end;
18039:
18040: Proc SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
18041: begin
18042:   TOBJECT'),'ETCPBuffer;
18043:   TTCPBuffer', 'record Ptr : TObject; Size : Int; Max : Intege'
18044:    +r; Head : Int; Used : Int; end;
18045:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
18046:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
18047:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
18048:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
18049: Proc TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSiz:Int;const TCPBufSize:Int;
18050:  Proc TCPBufferFinalise( var TCPBuf : TTCPBuffer);
18051:  Proc TCPBufferPack( var TCPBuf : TTCPBuffer);
18052:  Proc TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Int);
18053:  Proc TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Int);
18054:  Proc TCPBufferShrink( var TCPBuf : TTCPBuffer);
18055:  Func TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Int) : Pointer;
18056:  Proc TCPBufferAddBuf(var TCPBuf: TTCPBuffer; const Buf:str; const Size : Int);
18057:  Func TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Int;
18058:  Func TCPBufferPeek(var TCPBuf:TTCPBuffer;var Buf:Str; const Size:Int): Int;
18059:  Func TCPBufferRemove(var TCPBuf:TTCPBuffer;var Buf:str;const Size:Int): Int;
18060:  Proc TCPBufferClear( var TCPBuf : TTCPBuffer);
18061:  Func TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Int) : Int;
18062:  Func TCPBufferUsed( const TCPBuf : TTCPBuffer) : Int;
18063:  Func TCPBufferEmpty( const TCPBuf : TTCPBuffer) :Bool;
18064:  Func TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Int;
18065:  Func TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer;
18066:  Proc TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Int);
18067: end;
18068:
18069: Proc SIRegister_Glut(CL: TPSPascalCompiler);
18070: begin
18071:   //TypeS('PInt', '^Int // will not work;
18072:   //TypeS('PPChar', '^PChar // will not work;
18073:  ConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
18074:  ConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
18075:  ConstantN('GLUT_RGB','LongInt').SetInt( 0);
18076:  ConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
18077:  ConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
18078:  Proc LoadGlut( const dll :Str);
18079:  Proc FreeGlut;
18080: end;
18081:
18082: Proc SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
18083: begin
18084:   TypeS('TLEDColor',( ledcGreen, ledcRed, ledcGray );
18085:  Func GetLEDBitmapHandle( Color : TLEDColor; State:Boolean): THandle;
18086: end;
18087:
18088: Proc SIRegister_SwitchLed(CL: TPSPascalCompiler);
18089: begin
18090:   TLedColor, ( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black );
18091:   TTLedState, ( LedOn, LedOff, LedDisabled );
18092:   SIRegister_TSwitchLed(CL);
18093:  //Proc Register;
18094: end;
18095:
18096: Proc SIRegister_FileClass(CL: TPSPascalCompiler);
18097: begin
18098:   TypeS(TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
18099:    +'xisting, fdOpenAlways, fdTruncateExisting );
18100:   TypeS('TIFileAccess', '( faRead, faWrite, faReadWrite );
18101:   TypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite );
```

```
18102:   SIRegister_TCustomFile(CL);
18103:   SIRegister_TIFile(CL);
18104:   SIRegister_TMemoryFile(CL);
18105:   SIRegister_TTextFileReader(CL);
18106:   SIRegister_TTextFileWriter(CL);
18107:   SIRegister_TFileMapping(CL);
18108:   SIRegister_EFileError(CL);
18109: end;
18110:
18111: Proc SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
18112: begin
18113:   TypeS(TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
18114:    +, ffaDirectory, ffaArchive, ffaAnyFile );
18115:   TypeS('TAttributeSet', 'set of TFileAttributes;
18116:   SIRegister_TFileSearch(CL);
18117: end;
18118:
18119: Proc SIRegister_uColorFunctions(CL: TPSPascalCompiler);
18120: begin
18121:  TRGBType', 'record RedHex :Str; GreenHex :Str; BlueHex :'
18122:    + string; Red : Int; Green : Int; Blue : Int; end;
18123:  Func FadeColor( aColor : Longint; aFade : Int) : Tcolor;
18124:  Func CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:Int):Tcolor;
18125: end;
18126:
18127: Proc SIRegister_uSettings(CL: TPSPascalCompiler);
18128: begin
18129:  Proc SaveOscSettings;
18130:  Proc GetOscSettings;
18131: end;
18132:
18133: Proc SIRegister_cyDebug(CL: TPSPascalCompiler);
18134: begin
18135:   TProcProcessEvent', 'Proc ( Sender : TObject; Index : Int);
18136:   RecProcess', 'record Name : ShortString; DurationMs :Card; '
18137:    FirstDurationMs :Card; LastDurationMs :Card; MinDurationMs : Int'
18138:    64; MaxDurationMs :Card; MarksCount : Int; ArrayMarks : array of '
18139:    Cardinal; EnterCount : Int; ExitCount : Int; end;
18140:   SIRegister_TcyDebug(CL);
18141: end;
18142:
18143: (*----------------------------------------------------------------------------*)
18144: Proc SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
18145: begin
18146:   TCopyFileResult,(cfCreate, cfOverwrite,cfNoNeed, cfForceDirectoryError, cfCopyError );
18147:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent );
18148:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir );
18149:   SIRegister_TDestinationOptions(CL);
18150:   TProcCustomCopyFileEvent,Procedure(Sender:TObject;var CopyFileResult:TCopyFileResult);
18151: TProcOnCopyFileProgressEvent,Procedure(Sender:TObject;FileBytes,TransferedBytes:int64;PercentDone:Int64);
18152:   TProcOnCustomSetFileDestination','Proc (Sender:TObject;var FileName:str);
18153:   SIRegister_TcyCopyFiles(CL);
18154: Func cyCopyFile(FromFile,ToFile:Str; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
       ResetAttr:boolean): TCopyFileResult;
18155:  Func cyCopyFileEx(FromFile,ToFile:str;FileExists:TCopyFileExists;FileNotExists
       TCopyFileNotExists;ResetAttr:bool;aProgressBar:TProgressBar):TCopyFileResult;
18156:  Func cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters :Str; ArchiveFiles,
       ReadOnlyFiles, HiddenFiles,
       SystemFiles:TcyFileAttributeMode;FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;SubFolders,
       ResetAttributes:Boolean):Int;
18157: end;
18158:
18159: Proc SIRegister_cySearchFiles(CL: TPSPascalCompiler);
18160: begin
18161:   TypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth );
18162:   SIRegister_TcyFileAttributes(CL);
18163:   SIRegister_TSearchRecInstance(CL);
18164:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude );
18165:   TOptions', 'set of TOption;
18166:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting );
18167:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
       Accept:bool;
18168:   TProcOnValidateDirectoryEvent,Procedure(Sender:TObject;Directry:str;var Accept:bool);
18169:   SIRegister_TcyCustomSearchFiles(CL);
18170:   SIRegister_TcySearchFiles(CL);
18171:  Func FileNameRespondToMask( aFileName :Str; aMask :Str) :Bool;
18172:  Func IscyFolder( aSRec : TSearchrec) :Bool;
18173: end;
18174:
18175: Proc SIRegister_jcontrolutils(CL: TPSPascalCompiler);
18176: begin
18177:  Func jCountChar( const s :Str; ch : char) : Int;
18178:  Proc jSplit( const Delimiter : char; Input :Str; Strings : TStrings);
18179:  Func jNormalizeDate(const Value:str;theValue: TDateTime;const theFormat:str):str;
18180:  Func jNormalizeTime(const Value:str;theValue:TTime;const theFormat:str) :Str;
18181:  Func jNormalizeDateTime(const Val:str;theValue:TDateTime;const theFormat:str):str;
18182:  Func jNormalizeDateSeparator( const s :Str) :Str;
18183:  Func jIsValidDateString( const Value :Str) :Bool;
18184:  Func jIsValidTimeString( const Value :Str) :Bool;
```

```
18185:  Func jIsValidDateTimeString( const Value :Str) :Bool;
18186: end;
18187:
18188: Proc SIRegister_kcMapViewer(CL: TPSPascalCompiler);
18189: begin
18190:  ClassN(CL.FindClass('TOBJECT'),'TMapViewer;
18191:   TypeS('TMapSource', ( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
18192:    +gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
18193:    +, msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
18194:    +EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
18195:    +hooSatellite, msYahooHybrid,msOviNormal,msOviSatellite, msOviHybrid, msOviPhysical );
18196:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end;
18197:   TRealArea', 'record top: Extended; left:Extended; bottom: Extended; right: Extended; end;
18198:   TIntPoint', 'record X : Int64; Y : Int64; end;
18199:   TkcRealPoint', 'record X : Extended; Y : Extended; end;
18200:   TOnBeforeDownloadEvent','Procedure(Url:Str; str:TStream; var CanHandle:Bool);
18201:   TOnAfterDownloadEvent', 'Procedure( Url :Str; str : TStream);
18202:   SIRegister_TCustomDownloadEngine(CL);
18203:   SIRegister_TCustomGeolocationEngine(CL);
18204:   SIRegister_TMapViewer(CL);
18205: end;
18206:
18207: Proc SIRegister_cparserutils(CL: TPSPascalCompiler);
18208: begin
18209:  (*Func isFunc( name : TNamePart) :Bool;*)
18210:  Func isUnnamedFunc( name : TNamepart) :Bool;
18211:  Func isPtrToFunc( name : TNamePart) :Bool;
18212:  Func isFuncRetFuncPtr( name : TNamePart) :Bool;
18213:  Func isPtrToFuncRetFuncPtr( name : TNamePart) :Bool;
18214:  Func GetFuncParam( name : TNamePart) : TNamePart;
18215:  Func isArray( name : TNamePart) :Bool;
18216:  Func GetArrayPart( name : TNamePart) : TNamePart;
18217:  Func GetIdFromPart( name : TNamePart) : Ansistr;
18218:  Func GetIdPart( name : TNamePart) : TNamePart;
18219:  Func isNamePartPtrToFunc( part : TNamePart) :Bool;
18220:  Func isAnyBlock( part : TNamePart) :Bool;*)
18221:   TypeS('TLineInfo', record linestart : Int; lineend : Int; end;
18222:   SIRegister_TLineBreaker(CL);
18223:   TypeS('TNameKind', 'Int;
18224:   ClassN(CL.FindClass('TOBJECT'),'TNamePart;
18225:   //TypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end;
18226:  Func SphericalMod( X : Extended) : Extended;
18227:  Func cSign( Value : Extended) : Extended;
18228:  Func LimitFloat( const eValue, eMin, eMax : Extended) : Extended;
18229:  Func AngleToRadians( iAngle : Extended) : Extended;
18230:  Func RadiansToAngle( eRad : Extended) : Extended;
18231:  Func Cross180( iLong : Double) :Bool;
18232:  Func Mod180( Value : Int) : Int;
18233:  Func Mod180Float( Value : Extended) : Extended;
18234:  Func MulDivFloat( a, b, d : Extended) : Extended;
18235:  Func LongDiff( iLong1, iLong2 : Double) : Double;
18236:  Proc Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap);
18237:  Func Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap;
18238:  Func FixFilePath( const Inpath, CheckPath :Str) :Str;
18239:  Func UnFixFilePath( const Inpath, CheckPath :Str) :Str;
18240:  Proc FillStringList( sl : TStringList; const aText :Str);
18241: end;
18242:
18243: Proc SIRegister_LedNumber(CL: TPSPascalCompiler);
18244: begin
18245:   TLedSegmentSize', 'Int;
18246:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised );
18247:   SIRegister_TCustomLEDNumber(CL);
18248:   SIRegister_TLEDNumber(CL);
18249: end;
18250:
18251: Proc SIRegister_StStrL(CL: TPSPascalCompiler);
18252: begin
18253:   TypeS('LStrRec',record AllocSize:Longint;RefCount : Longint; Length : Longint; end;
18254:    TypeS(AnsiChar', 'Char;
18255:     TypeS('BTable', array[0..255] of Byte;       //!!!
18256:   Func HexBL( B : Byte) : Ansistr;
18257:  Func HexWL( W : Word) : Ansistr;
18258:  Func HexLL( L : LongInt) : Ansistr;
18259:  Func HexPtrL( P : ___Pointer) : Ansistr;
18260:  Func BinaryBL( B : Byte) : Ansistr;
18261:  Func BinaryWL( W : Word) : Ansistr;
18262:  Func BinaryLL( L : LongInt) : Ansistr;
18263:  Func OctalBL( B : Byte) : Ansistr;
18264:  Func OctalWL( W : Word) : Ansistr;
18265:  Func OctalLL( L : LongInt) : Ansistr;
18266:  Func Str2Int16L( const S : Ansistr; var I : SmallInt) :Bool;
18267:  Func Str2WordL( const S : Ansistr; var I : Word) :Bool;
18268:  Func Str2LongL( const S : Ansistr; var I : LongInt) :Bool;
18269:  Func Str2RealL( const S : Ansistr; var R : Double) :Bool;
18270:  Func Str2RealL( const S : Ansistr; var R : Real) :Bool;
18271:  Func Str2ExtL( const S : Ansistr; var R : Extended) :Bool;
18272:  Func Long2StrL( L : LongInt) : Ansistr;
18273:  Func Real2StrL( R : Double; Width : Byte; Places : ShortInt) : Ansistr;
```

```
18274:  Func Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : Ansistr;
18275:  Func ValPrepL( const S : Ansistr) : Ansistr;
18276:  Func CharStrL( C : Char; Len :Card) : Ansistr;
18277:  Func PadChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18278:  Func PadLL( const S : Ansistr; Len :Card) : Ansistr;
18279:  Func LeftPadChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18280:  Func LeftPadL( const S : Ansistr; Len :Card) : Ansistr;
18281:  Func TrimLeadL( const S : Ansistr) : Ansistr;
18282:  Func TrimTrailL( const S : Ansistr) : Ansistr;
18283:  Func TrimL( const S : Ansistr) : Ansistr;
18284:  Func TrimSpacesL( const S : Ansistr) : Ansistr;
18285:  Func CenterChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18286:  Func CenterL( const S : Ansistr; Len :Card) : Ansistr;
18287:  Func EntabL( const S : Ansistr; TabSize : Byte) : Ansistr;
18288:  Func DetabL( const S : Ansistr; TabSize : Byte) : Ansistr;
18289:  Func ScrambleL( const S, Key : Ansistr) : Ansistr;
18290:  Func SubstituteL( const S, FromStr, ToStr : Ansistr) : Ansistr;
18291:  Func FilterL( const S, Filters : Ansistr) : Ansistr;
18292:  Func CharExistsL( const S : Ansistr; C : AnsiChar) :Bool;
18293:  Func CharCountL( const S : Ansistr; C : AnsiChar) :Card;
18294:  Func WordCountL( const S, WordDelims : Ansistr) :Card;
18295:  Func WordPositionL(N:Card;const S,WordDelims:Ansistr;var Pos:Card):Bool;
18296:  Func ExtractWordL( N :Card; const S, WordDelims : Ansistr) : Ansistr;
18297:  Func AsciiCountL( const S, WordDelims : Ansistr; Quote : AnsiChar) :Card;
18298:  Func AsciiPositionL(N:Card;const S,WordDelims:AnsiStr;Quote:AnsiChr;var Pos:Card):Bool;
18299:  Func ExtractAsciiL(N:Card;const S,WordDelims:Ansistr;Quote:AnsiChar):Ansistr;
18300:  Proc WordWrapL(const InSt:Ansistr;var OutSt,Overlap:Ansistr;Margin:Card;PadToMargin:Bool;
18301:  Proc WordWrap(const InSt:AnsiStr;var OutSt,Overlap:AnsiStr;Margin:Card;PadToMargin:Bool;
18302:  Func CompStringL( const S1, S2 : Ansistr) : Int;
18303:  Func CompUCStringL( const S1, S2 : Ansistr) : Int;
18304:  Func SoundexL( const S : Ansistr) : Ansistr;
18305:  Func MakeLetterSetL( const S : Ansistr) : Longint;
18306:  Proc BMMakeTableL( const MatchString : Ansistr; var BT : BTable);
18307: Func BMSearchL(var Buffer,BufLength:Card;var BT:BTable;const MatchString:Ansistr;var Pos:Card):Bool;
18308:  Func BMSearchUCL( var Buffer, BufLength :Card; var BT: BTable; const MatchString: Ansistr; var Pos:Card)
       :Bool;
18309:  Func DefaultExtensionL( const Name, Ext : Ansistr) : Ansistr;
18310:  Func ForceExtensionL( const Name, Ext : Ansistr) : Ansistr;
18311:  Func JustFilenameL( const PathName : Ansistr) : Ansistr;
18312:  Func JustNameL( const PathName : Ansistr) : Ansistr;
18313:  Func JustExtensionL( const Name : Ansistr) : Ansistr;
18314:  Func JustPathnameL( const PathName : Ansistr) : Ansistr;
18315:  Func AddBackSlashL( const DirName : Ansistr) : Ansistr;
18316:  Func CleanPathNameL( const PathName : Ansistr) : Ansistr;
18317:  Func HasExtensionL( const Name : Ansistr; var DotPos :Card) :Bool;
18318:  Func CommaizeL( L : LongInt) : Ansistr;
18319:  Func CommaizeChL( L : Longint; Ch : AnsiChar) : Ansistr;
18320: Func FloatFormL(const Mask:Ansistr;R:TstFloat;const LtCurr,RtCurr:Ansistr;Sep,DecPt:Char):Ansistr;
18321: Func LongIntFormL(const Mask:Ansistr;L:LongInt;const LtCurr,RtCurr:AnsiStr;Sep:Char):Ansistr);
18322:  Func StrChPosL( const P : Ansistr; C : AnsiChar; var Pos :Card) :Bool;
18323:  Func StrStPosL( const P, S : Ansistr; var Pos :Card) :Bool;
18324:  Func StrStCopyL( const S : Ansistr; Pos, Count :Card) : Ansistr;
18325:  Func StrChInsertL( const S : Ansistr; C : AnsiChar; Pos :Card) : Ansistr;
18326:  Func StrStInsertL( const S1, S2 : Ansistr; Pos :Card) : Ansistr;
18327:  Func StrChDeleteL( const S : Ansistr; Pos :Card) : Ansistr;
18328:  Func StrStDeleteL( const S : Ansistr; Pos, Count :Card) : Ansistr;
18329:  Func ContainsOnlyL( const S, Chars : Ansistr; var BadPos :Card) :Bool;
18330:  Func ContainsOtherThanL(const S,Chars: Ansistr; var BadPos :Card):Bool;
18331:  Func CopyLeftL( const S : Ansistr; Len :Card) : Ansistr;
18332:  Func CopyMidL( const S : Ansistr; First, Len :Card) : Ansistr;
18333:  Func CopyRightL( const S : Ansistr; First :Card) : Ansistr;
18334:  Func CopyRightAbsL( const S : Ansistr; NumChars :Card) : Ansistr;
18335:  Func CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Card;var SubString:AString):Bool;
18336:  Func CopyFromToWordL(const S,WordDelims,Word1,Word2:Ansistr;N1,N2:Card;var SubString:Ansistr):Bool;
18337:  Func CopyWithinL( const S, Delimiter : Ansistr; Strip :Bool) : Ansistr;
18338:  Func DeleteFromNthWordL(const S, WordDelims:Ansistr;const AWord: Ansistr; N:Card; var SubString:
       Ansistr): Bool;
18339:  Func DeleteFromToWordL(const S,WordDelims,Word1,Word2:Ansistr;N1,N2:Card;var SubString:Ansistr):Bool;
18340:  Func DeleteWithinL( const S, Delimiter : Ansistr) : Ansistr;
18341: Func ExtractTokensL(const S,Delims:AnsiStr;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Card;
18342:  Func IsChAlphaL( C : AnsiChar) :Bool;
18343:  Func IsChNumericL( C : AnsiChar; const Numbers : Ansistr) :Bool;
18344:  Func IsChAlphaNumericL( C : AnsiChar; const Numbers : Ansistr) :Bool;
18345:  Func IsStrAlphaL( const S : Ansistr) :Bool;
18346:  Func IsStrNumericL( const S, Numbers : Ansistr) :Bool;
18347:  Func IsStrAlphaNumericL( const S, Numbers : Ansistr) :Bool;
18348:  Func KeepCharsL( const S, Chars : Ansistr) : Ansistr;
18349:  Func LastWordL(const S,WordDelims, AWord:Ansistr; var Position:Card):Bool;
18350:  Func LastWordAbsL( const S, WordDelims:Ansistr;var Position:Card):Bool;
18351:  Func LastStringL( const S, AString : Ansistr; var Position :Card):Bool;
18352:  Func LeftTrimCharsL( const S, Chars : Ansistr) : Ansistr;
18353: Func ReplaceWordL(const S,WordDelims,OldWord,NewWord:Ansistr;N:Card;var Replacements:Card):Ansistr;
18354: Func ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:Ansistr;var Replacements:Card):Ansistr;
18355: Func ReplaceStringL(const S,OldString,NewString:Ansistr;N:Card;var Replacements:Card):Ansistr;
18356: Func ReplaceStringAllL(const S,OldString,NewString:Ansistr; var Replacements:Card):Ansistr;
18357: Func RepeatStringL(const RepeatString:Ansistr;var Repetitions:Card;MaxLen:Card):Ansistr;
18358:  Func RightTrimCharsL( const S, Chars : Ansistr) : Ansistr;
18359:  Func StrWithinL(const S,SearchStr:str;Start:Card;var Position:Card): bool;
18360:  Func TrimCharsL( const S, Chars : Ansistr) : Ansistr;
```

```
18361:  Func WordPosL(const S,WordDelims,AWord:Ansistr;N:Card;var Position:Card):Bool;
18362:  Func WordPos(const S,WordDelims,AWord:Ansistr;N:Card;var Position:Card):Bool;
18363:  end;
18364:
18365: Proc SIRegister_pwnative_out(CL: TPSPascalCompiler);
18366: begin
18367:  ConstantN('STDIN','LongInt').SetInt( 0);
18368:  ('STDOUT','LongInt').SetInt( 1);
18369:  ('STDERR','LongInt').SetInt( 2);
18370:  Proc NativeWrite( s : astr);;
18371:  Proc NativeWrite1( PString : PChar);;
18372:  Proc NativeWrite2( Buffer : PChar; NumChars :Card);;
18373:  Proc NativeWriteLn( s : astr);;
18374:  Proc NativeWriteLn1;;
18375: end;
18376:
18377: Proc SIRegister_synwrap1(CL: TPSPascalCompiler);
18378: begin
18379:   TypeS(TSynwInfo', 'record Err : byte; UrlHtml : Ansistr; ErrResponse
18380:    : Int; UltimateURL : Ansistr; Headers : Ansistr; end;
18381:   TypeS('TUrlInfo', 'record Err : byte; UltimateURL :Str; end;
18382:  Func GetHttpFile(const Url,UserAgent,Outfile:str; verbose :Bool):TSynwInfo;
18383:  Func GetHttpFile1( const Url, UserAgent, Outfile :Str) : TSynwInfo;
18384:  Func GetHttpFile2( const Url, Outfile :Str) : TSynwInfo;;
18385:  Func GetHttpFile3( const Url, outfile :Str; verbose :Bool) : TSynwInfo;
18386:  Func GetHtm( const Url :Str) :Str;;
18387:  Func GetHtm1( const Url, UserAgent :Str) :Str;;
18388:  Func GetUrl( const Url :Str; verbose :Bool) : TSynwInfo;;
18389:  Func GetUrl1( const Url, useragent :Str) : TSynwInfo;;
18390:  Func GetUrl2( const Url :Str) : TSynwInfo;;
18391:  Func GetUrl3(const Url:str;const http THTTPSend; verbose :Bool): TUrlInfo;
18392:  Func GetUrl4( const Url :Str; const http : THTTPSend) : TUrlInfo;;
18393:  Proc StrToStream( s :Str; strm : TMemoryStream);
18394:  Func StrLoadStream( strm : TStream) :Str;
18395: end;
18396:
18397: Proc SIRegister_HTMLUtil(CL: TPSPascalCompiler);
18398: begin
18399:  Func GetVal( const tag, attribname_ci :Str) :Str;
18400:  Func GetTagName( const Tag :Str) :Str;
18401:  Func GetUpTagName( const tag :Str) :Str;
18402:  Func GetNameValPair( const tag, attribname_ci :Str) :Str;
18403:  Func GetValFromNameVal( const namevalpair :Str) :Str;
18404:  Func GetNameValPair_cs( const tag, attribname :Str) :Str;
18405:  Func GetVal_JAMES( const tag, attribname_ci :Str) :Str;
18406:  Func GetNameValPair_JAMES( const tag, attribname_ci :Str) :Str;
18407:  Func CopyBuffer( StartIndex : PChar; Len : Int) :Str;
18408:  Func Ucase( s :Str) :Str;
18409: end;
18410:
18411: Proc SIRegister_pwmain(CL: TPSPascalCompiler);
18412: begin
18413:  ConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT;
18414:  EXPIRED_COOKIE','String').SetString( 'Mon, 01 Jan 2001 12:00:00 GMT;
18415:  'SECURE_OFF','LongInt').SetInt( 0);
18416:  'SECURE_ON','LongInt').SetInt( 2);
18417:  'SECURE_FILTER','LongInt').SetInt( 3);
18418:   THandle or DWord!
18419:  //  astr = Ansistr;
18420:   TypeS('pastr', 'Ansistr;
18421:   TypeS('TFilterFunc', 'function(const s: pastr): pastr;;
18422:   uses pwinit at begin
18423:   //type  TFilterFunc = function(const s: astr): astr;
18424:  Demo: ..\maxbox3\examples2\519_powtils.txt
18425:  //ConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false);
18426:  //ConstantN('CASE_IGNORE','Boolean')BoolToStr( false);
18427:  Proc pwInit;
18428:  Proc OffReadln;
18429:  Func Lcase( const s : pastr) : pastr;
18430:  Func Ucase( const s : pastr) : pastr;
18431:  Func CountPostVars : longword;
18432:  Func GetPostVar( const name : pastr) : pastr;;
18433:  Func GetPostVar1( const name : pastr; filter : TFilterFunc) : pastr;;
18434:  Func GetPostVar_S( const name : pastr; Security : Int) : pastr;
18435:  Func GetPostVar_SF( const name : pastr; Security : Int) : pastr;
18436:  Func GetPostVarAsFloat( const name : pastr) : double;
18437:  Func GetPostVarAsInt( const name : pastr) : longint;
18438:  Func GetPostVar_SafeHTML( const name : pastr) : pastr;
18439:  Func FetchPostVarName( idx : longword) : pastr;
18440:  Func FetchPostVarVal( idx : longword) : pastr;;
18441:  Func FetchPostVarVal1( idx : longword; filter : TFilterFunc) : pastr;;
18442:  Func FetchPostVarName_S( idx : longword; Security : Int) : pastr;
18443:  Func FetchPostVarVal_S( idx : longword; Security : Int) : pastr;
18444:  Func IsPostVar( const name : pastr) :Bool;
18445:  Func CountAny : longword;
18446:  Func GetAny( const name : pastr) : pastr;;
18447:  Func GetAny1( const name : pastr; filter : TFilterFunc) : pastr;;
18448:  Func GetAny_S( const name : pastr; Security : Int) : pastr;
18449:  Func GetAnyAsFloat( const name : pastr) : double;
```

```
Func GetAnyAsInt( const name : pastr) : longint;
Func IsAny( const name : pastr) : byte;
Func CountCookies : longword;
Func FetchCookieName( idx : longword) : pastr;
Func FetchCookieVal( idx : longword) : pastr;;
Func FetchCookieVal1( idx : longword; filter : TFilterFunc) : pastr;;
Func GetCookie( const name : pastr) : pastr;;
Func GetCookie1( const name : pastr; filter : TFilterFunc) : pastr;;
Func GetCookieAsFloat( const name : pastr) : double;
Func GetCookieAsInt( const name : pastr) : longint;
Func IsCookie( const name : pastr) :Bool;
Func SetCookie( const name, value : pastr) :Bool;
Func SetCookieAsFloat( const name : pastr; value : double) :Bool;
Func SetCookieAsInt( const name : pastr; value : longint) :Bool;
Func SetCookieEx( const name, value, path, domain, expiry : pastr) :Bool;
Func SetCookieAsFloatEx(const name:pastr;value:double;const path,domai,expiry:pastr):bool;
Func SetCookieAsIntEx(const name:pastr;value:longint;const path,domain,expiry:pastr):bool;
Func UnsetCookie( const name : pastr) :Bool;
Func UnsetCookieEx( const name, path, domain : pastr) :Bool;
Func FilterHtml( const input : pastr) : pastr;
Func FilterHtml_S( const input : pastr; security : Int) : pastr;
Func TrimBadChars( const input : pastr) : pastr;
Func TrimBadFile( const input : pastr) : pastr;
Func TrimBadDir( const input : pastr) : pastr;
Func TrimBad_S( const input : pastr; security : Int) : pastr;
Func CountHeaders : longword;
Func FetchHeaderName( idx : longword) : pastr;
Func FetchHeaderVal( idx : longword) : pastr;
Func GetHeader( const name : pastr) : pastr;
Func IsHeader( const name : pastr) :Bool;
Func SetHeader( const name, value : pastr) :Bool;
Func UnsetHeader( const name : pastr) :Bool;
Func PutHeader( const header : pastr) :Bool;
Proc Out1( const s : pastr);
Proc OutLn( const s : pastr);
Proc OutA( args : array of const);
Proc OutF( const s : pastr);
Proc OutLnF( const s : pastr);
Proc OutFF( const s : pastr);
Proc OutF_FI( const s : pastr; HTMLFilter :Bool);
Proc OutLnFF( const s : pastr);
Proc OutLnF_FI( const s : pastr; HTMLFilter :Bool);
Func FileOut( const fname : pastr) : word;
Func ResourceOut( const fname : pastr) : word;
Proc BufferOut( const buff, len : LongWord);
Func TemplateOut( const fname : pastr; HtmlFilter :Bool) : word;;
Func TemplateOut1( const fname : pastr) : word;;
Func TemplateOut2( const fname : pastr; filter : TFilterFunc) : word;;
Func TemplateOut3( const fname : pastr; HtmlFilter :Bool) : word;;
Func TemplateRaw( const fname : pastr) : word;
Func Fmt( const s : pastr) : pastr;;
Func Fmt1( const s : pastr; filter : TFilterFunc) : pastr;;
Func FmtFilter( const s : pastr) : pastr;
Func Fmt_SF(const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecur,TrimSecurity:int):pastr;
Func Fmt_SF1(const s:pastr;HTMLFilter:bool;FilterSecurity,TrimSecurity:Int):pastr;;
Func CountRtiVars : longword;
Func FetchRtiName( idx : longword) : pastr;
Func FetchRtiVal( idx : longword) : pastr;
Func GetRti( const name : pastr) : pastr;
Func GetRtiAsFloat( const name : pastr) : double;
Func GetRtiAsInt( const name : pastr) : longint;
Func IsRti( const name : pastr) :Bool;
Proc SetRTI( const name, value : pastr);
Func FetchUpfileName( idx : longword) : pastr;
Func GetUpfileName( const name : pastr) : pastr;
Func GetUpfileSize( const name : pastr) : longint;
Func GetUpfileType( const name : pastr) : pastr;
Func CountUpfiles : longword;
Func IsUpfile( const name : pastr) :Bool;
Func SaveUpfile( const name, fname : pastr) :Bool;
Func CountVars : longword;
Func FetchVarName( idx : longword) : pastr;
Func FetchVarVal( idx : longword) : pastr;;
Func FetchVarVal1( idx : longword; filter : TFilterFunc) : pastr;;
Func GetVar( const name : pastr) : pastr;;
Func GetVar1( const name : pastr; filter : TFilterFunc) : pastr;;
Func GetVar_S( const name : pastr; security : Int) : pastr;
Func GetVarAsFloat( const name : pastr) : double;
Func GetVarAsInt( const name : pastr) : longint;
Proc SetVar( const name, value : pastr);
Proc SetVarAsFloat( const name : pastr; value : double);
Proc SetVarAsInt( const name : pastr; value : longint);
Func IsVar( const name : pastr) : byte;
Proc UnsetVar( const name : pastr);
Func LineEndToBR( const s : pastr) : pastr;
Func RandomStr( len : longint) : pastr;
Func XorCrypt( const s : pastr; key : byte) : pastr;
Func CountCfgVars : longword;
Func FetchCfgVarName( idx : longword) : pastr;
```

```
18539:  Func FetchCfgVarVal( idx : longword) : pastr;
18540:  Func IsCfgVar( const name : pastr) :Bool;
18541:  Func SetCfgVar( const name, value : pastr) :Bool;
18542:  Func GetCfgVar( const name : pastr) : pastr;
18543:  Proc ThrowErr( const s : pastr);
18544:  Proc ThrowWarn( const s : pastr);
18545:  Proc ErrWithHeader( const s : pastr);
18546: TypeS(TWebVar', 'record name : pastr; value : pastr; end;
18547: TypeS(TWebVars', 'array of TWebVar;
18548:  Func iUpdateWebVar(var webv:TWebVars; const name,value:pastr; upcased:boolean):boolean;
18549:  Func iAddWebCfgVar( const name, value : pastr) :Bool;
18550:  Proc iAddWebVar( var webv : TWebVars; const name, value : pastr);
18551:  Proc iSetRTI( const name, value : pastr);
18552:  Func iCustomSessUnitSet :Bool;
18553:  Func iCustomCfgUnitSet :Bool;
18554: end;
18555:
18556: Proc SIRegister_W32VersionInfo(CL: TPSPascalCompiler);
18557: begin
18558:   SIRegister_TProjectVersionInfo(CL);
18559:  Func MSLanguageToHex( const s :Str) :Str;
18560:  Func MSHexToLanguage( const s :Str) :Str;
18561:  Func MSCharacterSetToHex( const s :Str) :Str;
18562:  Func MSHexToCharacterSet( const s :Str) :Str;
18563:  Func MSLanguages : TStringList;
18564:  Func MSHexLanguages : TStringList;
18565:  Func MSCharacterSets : TStringList;
18566:  Func MSHexCharacterSets : TStringList;
18567: end;
18568:
18569: Proc SIRegister_IpUtils(CL: TPSPascalCompiler);
18570: begin
18571: TIpHandle', 'Cardinal;
18572: TIpMD5StateArray', 'array[0..3] of DWORD;;
18573: TypeS('TIpMD5CountArray', 'array[0..1] of DWORD;;
18574: TIpMD5ByteBuf', 'array[0..63] of Byte;;
18575: TIpMD5LongBuf', 'array[0..15] of DWORD;;
18576: TIpMD5Digest', 'array[0..15] of Byte;;
18577: TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther );
18578: TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5'
18579:  +'CountArray; ByteBuf : TIpMD5ByteBuf; end;
18580: ClassN(CL.FindClass('TOBJECT'),'EIpBaseException;
18581: ClassN(CL.FindClass('TOBJECT'),'EIpAccessException;
18582: ClassN(CL.FindClass('TOBJECT'),'EIpHtmlException;
18583: SIRegister_TIpBaseAccess(CL);
18584: SIRegister_TIpBasePersistent(CL);
18585: //TIpComponentClass', 'class of TIpBaseComponent;
18586: SIRegister_TIpBaseComponent(CL);
18587: SIRegister_TIpBaseWinControl(CL);
18588:  Func InClassA( Addr : LongInt) :Bool;
18589:  Func InClassB( Addr : LongInt) :Bool;
18590:  Func InClassC( Addr : LongInt) :Bool;
18591:  Func InClassD( Addr : LongInt) :Bool;
18592:  Func InMulticast( Addr : LongInt) :Bool;
18593:  Func IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar) : DWORD;
18594:  Func IpCompStruct( const S1, S2, Size :Card) : Int;
18595:  Func IpMaxInt( A, B : Int) : Int;
18596:  Func IpMinInt( A, B : Int) : Int;
18597:  Proc IpSafeFree( var Obj: TObject);
18598:  Func IpShortVersion :Str;
18599:  Func IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD) : DWORD;
18600:  Func IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD) : DWORD;
18601:  Func IpInternetSumOfFile( const FileName :Str) : DWORD;
18602:  Func MD5SumOfFile( const FileName :Str) :Str;
18603:  Func MD5SumOfStream( Stream : TStream) :Str;
18604:  Func MD5SumOfStreamDigest( Stream : TStream) : TIpMD5Digest;
18605:  Func MD5SumOfString( const S :Str) :Str;
18606:  Func MD5SumOfStringDigest( const S :Str) : TIpMD5Digest;
18607:  Func SafeYield : LongInt;
18608:  Func AllTrimSpaces( Strng :Str) :Str;
18609:  Func IpCharPos( C : AnsiChar; const S :Str) : Int;
18610:  Func CharPosIdx( C : AnsiChar; const S :Str; Idx : Int) : Int;
18611:  Func NthCharPos( C : AnsiChar; const S :Str; Nth : Int) : Int;
18612:  Func RCharPos( C : AnsiChar; const S :Str) : Int;
18613:  Func RCharPosIdx( C : AnsiChar; const S :Str; Idx : Int) : Int;
18614:  Func RNthCharPos( C : AnsiChar; const S :Str; Nth : Int) : Int;
18615:  Func IpRPos( const Substr :Str; const S :Str) : Int;
18616:  Func IpPosIdx( const SubStr, S :Str; Idx : Int) : Int;
18617:   ACharSet', 'set of AnsiChar;
18618:   TIpAddrRec', 'record Scheme:str; UserName :Str; Password string; Authority :Str;
18619:    Port:str;Path:str; Fragment:Str; Query :Str; QueryDelim : AnsiChar; end;
18620:  Proc Initialize( var AddrRec : TIpAddrRec);
18621:  Proc Finalize( var AddrRec : TIpAddrRec);
18622:  Func ExtractEntityName( const NamePath :Str) :Str;
18623:  Func ExtractEntityPath( const NamePath :Str) :Str;
18624:  Func IpParseURL( const URL :Str; var Rslt : TIpAddrRec) :Bool;
18625:  Func BuildURL( const OldURL, NewURL :Str) :Str;
18626:  Func PutEscapes( const S :Str; EscapeSet : ACharSet) :Str;
18627:  Func RemoveEscapes( const S :Str; EscapeSet : ACharSet) :Str;
```

```
18628:  Proc SplitParams( const Parms :Str; Dest : TStrings);
18629:  Func NetToDOSPath( const PathStr :Str) :Str;
18630:  Func DOSToNetPath( const PathStr :Str) :Str;
18631:  Proc SplitHttpResponse( const S :Str; var V, MsgID, Msg :Str);
18632:  Proc FieldFix( Fields : TStrings);
18633:  Func AppendSlash( APath :Str) :Str;
18634:  Func RemoveSlash( APath :Str) :Str;
18635:  Func GetParentPath( const Path :Str) :Str;
18636:  Func GetLocalContent( const TheFileName :Str) :Str;
18637:  Func IPDirExists( Dir :Str) :Bool;
18638:  Func GetTemporaryFile( const Path :Str) :Str;
18639:  Func GetTemporaryPath :Str;
18640:  Func AppendBackSlash( APath :Str) :Str;
18641:  Func IpRemoveBackSlash( APath :Str) :Str;
18642:  Func INetDateStrToDateTime( const DateStr :Str) : TDateTime;
18643:  Func DateTimeToINetDateTimeStr( DateTime : TDateTime) :Str;
18644:  Func IpTimeZoneBias : Int;
18645:  Proc SplitCookieFields( const Data :Str; Fields : TStrings);
18646: end;
18647:
18648: Proc SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18649: begin
18650:   TypeS('TPOStyle', '( postStandard, postPropName, postFull );
18651:  Proc Lrt2Po( const LRTFile :Str; POStyle : TPOStyle);
18652:  Proc CombinePoFiles( SL : TStrings; const FName :Str);
18653: end;
18654:
18655: Proc SIRegister_GPS(CL: TPSPascalCompiler);
18656: begin
18657:  ConstantN('MAX_SATS','LongInt').SetInt( 12);
18658:  GPSMSG_START','String').SetString( '$;
18659:  GPSMSG_STOP','String').SetString( '*;
18660:  SEC_BETWEEN_SEG','LongInt').SetInt( 5);
18661:   TypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor'
18662:    +'tint; Azimut : Smallint; SignLevel : Smallint; end;
18663:   TypeS('TSatellites', 'array[1..12] of TSatellite;
18664:  //TSatellites = array[1..MAX_SATS] of TSatellite;
18665:   TGPSSatEvent,Procedure(Sender: TObject; NbSat, NbSatUse: Shortint; Sats: TSatellites);
18666:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Heigh'
18667:    tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid :Bool; '
18668:    NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end;
18669:   TypeS('TGPSDatasEvent', Proc ( Sender : TObject; GPSDatas : TGPSDatas);
18670:   TypeS('TMsgGP',( msgGP,msgGPGGA,msgGPGLL,msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA );
18671:   TypeS('TSpeedUnit',( suKilometre, suMile, suNauticalMile );
18672:   SIRegister_TGPSLink(CL);
18673:   SIRegister_TCustomGPS(CL);
18674:   SIRegister_TGPS(CL);
18675:   SIRegister_TGPStoGPX(CL);
18676:   SIRegister_TGPSSpeed(CL);
18677:   SIRegister_TGPSSatellitesPosition(CL);
18678:   SIRegister_TGPSSatellitesReception(CL);
18679:   SIRegister_TGPSCompass(CL);
18680:  //Proc Register( );
18681:  Func IndexMsgGP( StrMsgGP :Str) : TMsgGP;
18682:  Func StrCoordToAngle( Point : Char; Angle :Str) : Double;
18683:  Func StrTimeToTime( const Time :Str) : TDateTime;
18684:  Func StrToInt( const Str :Str) : Int;
18685:  Func StrToReal( const Str :Str) : Extended;
18686:  Func GPSRotatePoint( Angle : Double; Ct, Pt : TPoint) : TPoint;
18687:  Proc LoadRessource( RessourceName :Str; ImageList : TImageList);
18688: end;
18689:
18690: Proc SIRegister_NMEA(CL: TPSPascalCompiler);
18691: begin
18692:  NMEADataArray', 'array of string;
18693:  Proc TrimNMEA( var S :Str);
18694:  Proc ExpandNMEA( var S :Str);
18695:  Func ParseNMEA( S :Str) : NMEADataArray;
18696:  Func ChkValidNMEA( S :Str) :Bool;
18697:  Func IdNMEA( S :Str) :Str;
18698:  Func ChkSumNMEA( const S :Str) :Str;
18699:  Func PosInDeg( const PosStr :Str) : Double;
18700:  Func DateTimeNMEA( const StrD, StrT :Str) : TDateTime;
18701:  Func SysClockSet( const StrD, StrT :Str) :Bool;
18702:  Func Ticks2Secs(Ticks : LongInt) : LongInt;;
18703:  Func Secs2Ticks(Secs : LongInt) : LongInt;;
18704:  Func MSecs2Ticks(MSecs : LongInt) : LongInt;;
18705: end;
18706:
18707: Proc SIRegister_SortUtils(CL: TPSPascalCompiler);
18708: begin
18709:   TypeS('SortType1', 'Byte;
18710:   TypeS('SortType2', 'Double;
18711:   TypeS('SortType3', 'DWord;
18712:   //TypeS('PDWordArray', '^DWordArray // will not work;
18713:   TypeS('TDataRecord4', 'record Value : Int; Data : Int; end;
18714:  Function('Proc QuickSort( var List : array of SortType1; Min, Max : Int);
18715:  Proc QuickSortDWord( var List : array of SortType3; Min, Max : Int);
18716:  Proc QuickSortDataRecord4( var List : array of TDataRecord4; Count : Int);
```

```
18717:  Proc HeapSort( var List : array of SortType1; Count : DWord; FirstNeeded : DWord);
18718:  Func QuickSelect(var List: array of SortType1; Min, Max, Wanted : Int) : SortType1;
18719:  Func QuickSelectDouble( var List : array of SortType2; Min, Max, Wanted : Int) : SortType2;
18720:  Func QuickSelectDWord( var List : array of SortType3; Min,Max,Wanted: Int):SortType3;
18721: end;
18722:
18723: Proc SIRegister_BitmapConversion(CL: TPSPascalCompiler);
18724: begin
18725:  // TMatrix3x3 = array[1..3,1..3] of Double;
18726:  // TMatrix4x4 = array[1..4,1..4] of Double;
18727:  TypeS('TMatrix3x31', 'array[1..3] of Double;
18728:  TypeS('TMatrix3x3', 'array[1..3] of TMatrix3x31;
18729:  TypeS('TMatrix4x41', 'array[1..4] of Double;
18730:  TypeS('TMatrix4x4', 'array[1..4] of TMatrix4x41;
18731:  Proc ColorTransform( A, B, C : Byte; out X, Y, Z : Byte; const T : TMatrix4x4);
18732:  Proc ColorTransform1( A, B, C : Byte; out X, Y, Z : Float; const T : TMatrix4x4);
18733:  Proc ColorTransform2(const A,B,C:Float; out X, Y, Z : Byte; const T: TMatrix4x4);
18734:  Proc ColorTransformHSI2RGB( H, S, I : Byte; out R, G, B : Byte);
18735:  Proc ColorTransformRGB2HSI( R, G, B : Byte; out H, S, I : Byte);
18736:  Proc ColorTransformRGB2Lab( R, G, B : Byte; out L, a_, b_ : Byte);
18737:  Proc ColorTransformLab2RGB( L, a_, b_ : Byte; out R, G, B : Byte);
18738:  Proc ColorTransformRGB2LOCO( R, G, B : Byte; out S0, S1, S2 : Byte);
18739:  Proc ColorTransformLOCO2RGB( S0, S1, S2 : Byte; out R, G, B : Byte);
18740:  Proc ConvertColorSpace(Image:TLinarBitmap;const T:TMatrix4x4;NewImage:TLinarBitmap);
18741: //Proc ConvertColorSpace1(Image:TLinarBitmap;ColorTransform:TColorTransformProc;NewImage:TLinarBitmap);
18742:  Proc ConvertToGrayscale( const Image, GrayImage : TLinarBitmap);;
18743:  Proc ConvertToGrayscale1( const Image : TLinarBitmap);;
18744: end;
18745:
18746: Proc SIRegister_ZDbcUtils(CL: TPSPascalCompiler);
18747: begin
18748:  { TZSQLType = (zsqlstUnknown, zsqlstBoolean, zsqlstByte, zsqlstShort, zsqlstInt, zsqlstLong,
18749:     zsqlstFloat,zsqlstDouble,zsqlstBigDecimal, zsqlstString, zsqlstUnicodeString, zsqlstBytes,
18750:     zsqlstDate, zsqlstTime, zsqlstTimestamp, zsqlstDataSet, zsqlstGUID,
18751:     stAsciiStream, stUnicodeStream, stBinaryStream);}
18752:  Func ResolveConnectionProtocol(Url:str; SupportedProtocols:TStringDynArray):Str;
18753:  //Proc ResolveDatabaseUrl( const Url:Str; Info:TStrings; var HostName:str; var Port:Int;var
       Database:str;var UserName:str; var Password :Str; ResultInfo : TStrings);
18754:  Func CheckConvertion( InitialType : TZSQLType; ResultType : TZSQLType) :Bool;
18755:  Func DefineColumnTypeName( ColumnType : TZSQLType) :Str;
18756:  Proc RaiseSQLException( E : Exception);
18757:  //Proc CopyColumnsInfo( FromList : TObjectList; ToList : TObjectList);
18758:  Func ToLikeString( const Value :Str) :Str;
18759:  Func GetSQLHexWideString( Value : PChar; Len : Int; ODBC :Bool) : WideString;
18760:  Func GetSQLHexAnsistr( Value : PChar; Len : Int; ODBC :Bool) : RawByteString;
18761:  Func GetSQLHexString( Value : PChar; Len : Int; ODBC :Bool) :Str;
18762:  Func WideStringStream( const AString : WideString) : TStream;
18763:  Func ConvertAdoToTypeName(FieldType: SmallInt):Str;
18764:  Func GetTableName(const AField: TField):Str;;
18765:  Func GetFieldName(const AField: TField):Str;;
18766: end;
18767:
18768: Proc SIRegister_JclTD32(CL: TPSPascalCompiler);
18769: TypeS('TSymbolInfo', 'record Size : Word; SymbolType : Word; end;
18770: //TypeS('PSymbolInfos', '^TSymbolInfos // will not work;
18771: SIRegister_TJclModuleInfo(CL);
18772: SIRegister_TJclLineInfo(CL);
18773: SIRegister_TJclSourceModuleInfo(CL);
18774: SIRegister_TJclSymbolInfo(CL);
18775: SIRegister_TJclProcSymbolInfo(CL);
18776: ClassN(CL.FindClass('TOBJECT'),'TJclLocalProcSymbolInfo;
18777: ClassN(CL.FindClass('TOBJECT'),'TJclGlobalProcSymbolInfo;
18778: SIRegister_TJclTD32InfoParser(CL);
18779: SIRegister_TJclTD32InfoScanner(CL);
18780: SIRegister_TJclPeBorTD32Image(CL);
18781: end;
18782:
18783: Proc SIRegister_JvIni(CL: TPSPascalCompiler);
18784: begin
18785:   TypeS('TReadObjectEvent', 'Func ( Sender : TObject; const Section, '
18786:    +'Item, Value :Str) : TObject;
18787:   TWriteObjectEvent,Procedure(Sender:TObject;const Section, Item:Str; Obj: TObject);
18788:  Func StringToFontStyles( const Styles :Str) : TFontStyles;
18789:  Func FontStylesToString( Styles : TFontStyles) :Str;
18790:  Func FontToString( Font : TFont) :Str;
18791:  Proc StringToFont( const Str :Str; Font : TFont);
18792:  Func RectToStr( Rect : TRect) :Str;
18793:  Func StrToRect( const Str :Str; const Def : TRect) : TRect;
18794:  Func JPointToStr( P : TPoint) :Str;
18795:  Func JStrToPoint( const Str :Str; const Def : TPoint) : TPoint;
18796:  Func DefProfileName :Str;
18797:  Func DefLocalProfileName :Str;
18798:  ConstantN('idnListItem','String').SetString( 'Item;
18799: end;
18800:
18801: Proc SIRegister_JvHtControls(CL: TPSPascalCompiler);
18802: begin
18803:  Proc ItemHtDrawEx(Canvas : TCanvas;Rect  TRect; const State:TOwnerDrawState; const Text :Str; const
       HideSelColor:Boolean; var PlainItem:Str; var Width: Int; CalcWidth:Boolean);
```

```
18804: Func ItemHtDraw(Canvas: TCanvas;Rect:TRect;const State TOwnerDrawState;const Text:str;const
       HideSelColor:Booln):str;
18805: Func ItemHtWidth(Canvas:TCanvas;Rect:TRect;const State:TOwnerDrawState;const Text:str;const
       HideSelColor:Bool):Int;
18806:  Func ItemHtPlain( const Text :Str) :Str;
18807:  Proc ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:str);
18808:  Func IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:str;MouseX,MouseY:Int;var HyperLink:str):Bool;
18809: end;
18810:
18811: Proc SIRegister_NeuralNetwork(CL: TPSPascalCompiler);
18812: begin
18813:   ClassN(CL.FindClass('TOBJECT'),'TNeuron;
18814:   TypeS(TSynapse', 'record W : Real; Connection : TNeuron; end;
18815:   TypeS(TAcson', 'record Alfa : Real; Beta : Real; Gama : Real; end;
18816:   SIRegister_TNeuron(CL);
18817:   SIRegister_TNeuronLayer(CL);
18818:   SIRegister_TNeuralNet(CL);
18819: end;
18820:
18821: Proc SIRegister_StExpr(CL: TPSPascalCompiler);
18822: begin
18823: //TypeS('PStFloat', '^TStFloat // will not work;
18824: TStMethod0Param', 'Func  : TStFloat;
18825: TStMethod1Param', 'Func ( Value1 : TStFloat) : TStFloat;
18826: TStMethod2Param', 'Func ( Value1, Value2 : TStFloat) : TStFloat;
18827: TStMethod3Param', 'Func ( Value1, Value2, Value3 : TStFloat) : TStFloat;
18828: TStGetIdentValueEvent', 'Proc ( Sender : TObject; const Ide'
18829:  +ntifier : Ansistr; var Value : TStFloat);
18830: TStToken', '( ssStart, ssInIdent, ssInNum, ssInSign, ssInExp, ss'
18831:  +Eol,ssNum,ssIdent,ssLPar,ssRPar,ssComma,ssPlus,ssMinus,ssTimes,ssDiv,ssEqual,ssPower);
18832: SIRegister_TStExpression(CL);
18833: TStExprErrorEvent', 'Proc ( Sender : TObject; ErrorNumber :'
18834:  + LongInt; const ErrorStr : Ansistr);
18835: SIRegister_TStExpressionEdit(CL);
18836:  Func AnalyzeExpr( const Expr : Ansistr) : Double;
18837:  Proc TpVal( const S : Ansistr; var V : Extended; var Code : Int);
18838: end;
18839:
18840: Proc SIRegister_GR32_Containers(CL: TPSPascalCompiler);
18841: begin
18842:  ConstantN('BUCKET_MASK','LongWord').SetUInt( $FF);
18843:  ConstantN('BUCKET_COUNT','LongInt').SetInt( BUCKET_MASK + 1);
18844:  Proc SmartAssign( Src, Dst : TPersistent; TypeKinds : TTypeKinds);
18845:  Proc Advance( var Node : TLinkedNode; Steps : Int);
18846: end;
18847:
18848: Proc SIRegister_StSaturn(CL: TPSPascalCompiler);
18849: begin
18850:  TStJupSatPos', 'record X: double; Y: Double; end;
18851:  TStJupSats record Io:TStJupSatPos;Europa:TStJupSatPos;Ganymede:TStJupSatPos;Callisto:TStJupSatPos;end;
18852:  Func ComputeSaturn( JD : Double) : TStEclipticalCord;
18853:  Func ComputePluto( JD : Double) : TStEclipticalCord;
18854:  Func ComputeVenus( JD : Double) : TStEclipticalCord;
18855:  Func ComputeMars( JD : Double) : TStEclipticalCord;
18856:  Func ComputeMercury( JD : Double) : TStEclipticalCord;
18857:  Func ComputeJupiter( JD : Double) : TStEclipticalCord;
18858:  Func ComputeUranus( JD : Double) : TStEclipticalCord;
18859:  Func ComputeNeptune( JD : Double) : TStEclipticalCord;
18860:  Func GetJupSats(JD : TDateTime; HighPrecision, Shadows :Bool) : TStJupSats;;
18861: end;
18862:
18863: Proc SIRegister_JclParseUses(CL: TPSPascalCompiler);
18864: begin
18865:   ClassN(CL.FindClass('TOBJECT'),'EUsesListError;
18866:  Func CreateGoal( Text : PChar) : TCustomGoal;
18867: end;
18868:
18869: Proc SIRegister_JvFinalize(CL: TPSPascalCompiler);
18870: begin
18871: //type
18872: //TFinalizeProc = procedure;
18873:   TypeS('TFinalizeProc', 'procedure;
18874:  Proc AddFinalizeProc( const UnitName :Str; FinalizeProc : TFinalizeProc);
18875:  Func AddFinalizeObject( const UnitName :Str; Instance : TObject) : TObject;
18876:  Func AddFinalizeObjectNil(const UnitName:str; var Reference: TObject):TObject;
18877:  Func AddFinalizeFreeAndNil(const UnitName:str; var Reference: TObject) : TObject;
18878:  Func AddFinalizeMemory( const UnitName :Str; Ptr : ___Pointer) : ___Pointer;
18879:  Func AddFinalizeMemoryNil(const UnitName:Str;var Ptr: ___Pointer) : ___Pointer;
18880:  Proc FinalizeUnit( const UnitName :Str);
18881: end;
18882:
18883: Proc SIRegister_BigIni(CL: TPSPascalCompiler);
18884: begin
18885:  ConstantN('IniTextBufferSize','LongWord').SetUInt( $7000);
18886:  cIniCount','String').SetString( 'Count;
18887:  TEraseSectionCallback', 'Func ( const sectionName :Str; '
18888:   const sl1, sl2 : TStringList) :Bool;
18889:   SIRegister_TCommaSeparatedInfo(CL);
18890:   SIRegister_TSectionList(CL);
```

```
18891:   SIRegister_TBigIniFile(CL);
18892:   SIRegister_TBiggerIniFile(CL);
18893:   SIRegister_TAppIniFile(CL);
18894:   SIRegister_TLibIniFile(CL);
18895:  Func ModuleName( getLibraryName :Bool) :Str;
18896: end;
18897:
18898: Proc SIRegister_ShellCtrls(CL: TPSPascalCompiler);
18899: begin
18900:   TypeS(TRoot', 'string;
18901:   TypeS(TRootFolder', '( rfDesktop, rfMyComputer, rfNetwork, rfRecycleBi'
18902:    +n, rfAppData, rfCommonDesktopDirectory, rfCommonPrograms, rfCommonStartMen'
18903:    +u, rfCommonStartup, rfControlPanel, rfDesktopDirectory, rfFavorites, rfFon'
18904:    +ts, rfInternet, rfPersonal, rfPrinters, rfPrintHood, rfPrograms, rfRecent,'
18905:    + rfSendTo, rfStartMenu, rfStartup, rfTemplates );
18906:   TShellFolderCapability', '( fcCanCopy, fcCanDelete, fcCanLink, f'
18907:    +'cCanMove, fcCanRename, fcDropTarget, fcHasPropSheet );
18908:   TShellFolderCapabilities', 'set of TShellFolderCapability;
18909:   TShellFolderProperty', '( fpCut, fpIsLink, fpReadOnly, fpShared,'
18910:    + fpFileSystem, fpFileSystemAncestor, fpRemovable, fpValidate );
18911:   TShellFolderProperties', 'set of TShellFolderProperty;
18912:   TShellObjectType', '( otFolders, otNonFolders, otHidden );
18913:   TShellObjectTypes', 'set of TShellObjectType;
18914:   ClassN(CL.FindClass('TOBJECT'),'EInvalidPath;
18915:   SIRegister_IShellCommandVerb(CL);
18916:   SIRegister_TShellFolder(CL);
18917:   TNotifyFilter', '( nfFileNameChange, nfDirNameChange, nfAttribut'
18918:    +eChange, nfSizeChange, nfWriteChange, nfSecurityChange );
18919:   TNotifyFilters', 'set of TNotifyFilter;
18920:   SIRegister_TShellChangeThread(CL);
18921:   SIRegister_TCustomShellChangeNotifier(CL);
18922:   SIRegister_TShellChangeNotifier(CL);
18923:   ClassN(CL.FindClass('TOBJECT'),'TCustomShellComboBox;
18924:   ClassN(CL.FindClass('TOBJECT'),'TCustomShellListView;
18925:   TAddFolderEvent', 'Proc ( Sender : TObject; AFolder: TShel'
18926:    +lFolder; var CanAdd :Bool);
18927:   TGetImageIndexEvent', 'Proc ( Sender : TObject; Index : Int'
18928:    +'eger; var ImageIndex : Int);
18929:   SIRegister_TCustomShellTreeView(CL);
18930:   SIRegister_TShellTreeView(CL);
18931:   SIRegister_TCustomShellComboBox(CL);
18932:   SIRegister_TShellComboBox(CL);
18933:   SIRegister_TCustomShellListView(CL);
18934:   SIRegister_TShellListView(CL);
18935:  Proc InvokeContextMenu(Owner : TWinControl; AFolder : TShellFolder; X, Y : Int);
18936: end;
18937:
18938: Proc SIRegister_fmath(CL: TPSPascalCompiler);
18939: begin
18940:  TypeS('Float', 'Double; !
18941:  'FN_OK','LongInt').SetInt( 0);
18942:  ConstantN('FN_DOMAIN','LongInt').SetInt( - 1);
18943:  'FN_SING','LongInt').SetInt( - 2);
18944:  'FN_OVERFLOW','LongInt').SetInt( - 3);
18945:  'FN_UNDERFLOW','LongInt').SetInt( - 4);
18946:  'FN_TLOSS','LongInt').SetInt( - 5);
18947:  'FN_PLOSS','LongInt').SetInt( - 6);
18948:  //ConstantN('NFACT','LongInt').SetInt( 33);
18949:  Func MathError : Int;
18950:  Func FMin( X, Y : Float) : Float;
18951:  Func FMax( X, Y : Float) : Float;
18952:  Func IMin( X, Y : Int) : Int;
18953:  Func IMax( X, Y : Int) : Int;
18954:  Func FSgn( X : Float) : Int;
18955:  Func Sgn0( X : Float) : Int;
18956:  Func DSgn( A, B : Float) : Float;
18957:  Proc FSwap( var X, Y : Float);
18958:  Proc ISwap( var X, Y : Int);
18959:  Func fExpo( X : Float) : Float;
18960:  Func fExp2( X : Float) : Float;
18961:  Func fExp10( X : Float) : Float;
18962:  Func fLog( X : Float) : Float;
18963:  Func fLog2( X : Float) : Float;
18964:  Func fLog10( X : Float) : Float;
18965:  Func fLogA( X, A : Float) : Float;
18966:  Func fIntPower( X : Float; N : Int) : Float;
18967:  Func fPower( X, Y : Float) : Float;
18968:  Func Pythag( X, Y : Float) : Float;
18969:  Func FixAngle( Theta : Float) : Float;
18970:  Func fTan( X : Float) : Float;
18971:  Func fArcSin( X : Float) : Float;
18972:  Func fArcCos( X : Float) : Float;
18973:  Func fArcTan2( Y, X : Float) : Float;
18974:  Proc fSinCos( X : Float; var SinX, CosX : Float);
18975:  Func fSinh( X : Float) : Float;
18976:  Func fCosh( X : Float) : Float;
18977:  Func fTanh( X : Float) : Float;
18978:  Func fArcSinh( X : Float) : Float;
18979:  Func fArcCosh( X : Float) : Float;
```

```
18980:  Func fArcTanh( X : Float) : Float;
18981:  Proc fSinhCosh( X : Float; var SinhX, CoshX : Float);
18982:  Func fFact( N : Int) : Float;
18983:  Func fBinomial( N, K : Int) : Float;
18984:  Func fGamma( X : Float) : Float;
18985:  Func fSgnGamma( X : Float) : Int;
18986:  Func LnGamma( X : Float) : Float;
18987:  Func fIGamma( A, X : Float) : Float;
18988:  Func fJGamma( A, X : Float) : Float;
18989:  Func fBeta( X, Y : Float) : Float;
18990:  Func fIBeta( A, B, X : Float) : Float;
18991:  Func fErf( X : Float) : Float;
18992:  Func fErfc( X : Float) : Float;
18993:  Func fPBinom(N: Int; P: Float; K : Int) : Float;
18994:  Func FBinom(N: Int; P: Float; K : Int) : Float;
18995:  Func PPoisson( Mu : Float; K : Int) : Float;
18996:  Func FPoisson( Mu : Float; K : Int) : Float;
18997:  Func fDNorm( X : Float) : Float;
18998:  Func FNorm( X : Float) : Float;
18999:  Func PNorm( X : Float) : Float;
19000:  Func InvNorm( P : Float) : Float;
19001:  Func fDStudent( Nu : Int; X : Float) : Float;
19002:  Func FStudent( Nu : Int; X : Float) : Float;
19003:  Func PStudent( Nu : Int; X : Float) : Float;
19004:  Func fDKhi2( Nu : Int; X : Float) : Float;
19005:  Func FKhi2( Nu : Int; X : Float) : Float;
19006:  Func PKhi2( Nu : Int; X : Float) : Float;
19007:  Func fDSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
19008:  Func FSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
19009:  Func PSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
19010:  Func fDExpo( A, X : Float) : Float;
19011:  Func FExpo( A, X : Float) : Float;
19012:  Func fDBeta( A, B, X : Float) : Float;
19013:  Func FBeta( A, B, X : Float) : Float;
19014:  Func fDGamma( A, B, X : Float) : Float;
19015:  Func FGamma( A, B, X : Float) : Float;
19016:  Proc RMarIn( Seed1, Seed2 : Int);
19017:  Func IRanMar : LongInt;
19018:  Func RanMar : Float;
19019:  Func RanGaussStd : Float;
19020:  Func RanGauss( Mu, Sigma : Float) : Float;
19021: end;
19022:
19023: Proc SIRegister_fcomp(CL: TPSPascalCompiler);
19024: begin
19025:  TypeS('ComplexForm', '( Rec, Pol );
19026:  TypeS(TComplex', 'record Form:ComplexForm; X Float;Y: Float; R:Float; Theta:Float; end;
19027:  Proc CSet( var Z : TComplex; A, B : Float; F : ComplexForm);
19028:  Proc CConvert( var Z : TComplex; F : ComplexForm);
19029:  Proc CSwap( var X, Y : TComplex);
19030:  Func CReal( Z : TComplex) : Float;
19031:  Func CImag( Z : TComplex) : Float;
19032:  Func CAbs( Z : TComplex) : Float;
19033:  Func CArg( Z : TComplex) : Float;
19034:  Func CSgn( Z : TComplex) : Int;
19035:  Proc CNeg( A : TComplex; var Z : TComplex);
19036:  Proc CConj( A : TComplex; var Z : TComplex);
19037:  Proc CAdd( A, B : TComplex; var Z : TComplex);
19038:  Proc CSub( A, B : TComplex; var Z : TComplex);
19039:  Proc CDiv( A, B : TComplex; var Z : TComplex);
19040:  Proc CMult( A, B : TComplex; var Z : TComplex);
19041:  Proc CLn( A : TComplex; var Z : TComplex);
19042:  Proc CExp( A : TComplex; var Z : TComplex);
19043:  Proc CPower( A, B : TComplex; var Z : TComplex);
19044:  Proc CIntPower( A : TComplex; N : Int; var Z : TComplex);
19045:  Proc CRealPower( A : TComplex; X : Float; var Z : TComplex);
19046:  Proc CSqrt( A : TComplex; var Z : TComplex);
19047:  Proc CRoot( A : TComplex; K, N : Int; var Z : TComplex);
19048:  Proc CCSin( A : TComplex; var Z : TComplex);
19049:  Proc CCos( A : TComplex; var Z : TComplex);
19050:  Proc CTan( A : TComplex; var Z : TComplex);
19051:  Proc CArcSin( A : TComplex; var Z : TComplex);
19052:  Proc CArcCos( A : TComplex; var Z : TComplex);
19053:  Proc CArcTan( A : TComplex; var Z : TComplex);
19054:  Proc CSinh( A : TComplex; var Z : TComplex);
19055:  Proc CCosh( A : TComplex; var Z : TComplex);
19056:  Proc CTanh( A : TComplex; var Z : TComplex);
19057:  Proc CArcSinh(A: TComplex; var Z : TComplex);
19058:  Proc CArcCosh(A: TComplex; var Z : TComplex);
19059:  Proc CArcTanh(A: TComplex; var Z : TComplex);
19060:  Proc CLnGamma(A: TComplex; var Z : TComplex);
19061: end;
19062:
19063:  XSDd:= DateTimeToXSDateTime(now,true)
19064:      xsdd.nativetoxs;
19065:      writeln(datetimetostr(xsdd.asUTCdatetime));
19066:      xsdd.free;
19067:
19068: Proc SIRegister_XSBuiltIns(CL: TPSPascalCompiler);
```

```
19069: begin
19070:  ConstantN('SHexMarker','String').SetString( '$;
19071:  Func DateTimeToXMLTime( Value : TDateTime; ApplyLocalBias :Bool) : WideString;
19072:  Func XMLTimeToDateTime(const XMLDateTime : WideString; AsUTCTime:Boolean): TDateTime;
19073:  Func DateTimeToXSDateTime(const Value:TDateTime;ApplyLocalBias:Bool):TXSDateTime;
19074:  Func GetDataFromFile( AFileName :Str) :Str;
19075:  Func SoapFloatToStr( Value : double) :Str;
19076:  Func SoapStrToFloat( Value :Str) : double;
19077:  Proc InitXSTypes;
19078: end;
19079:
19080: Proc SIRegister_CompFileIo(CL: TPSPascalCompiler);
19081: begin
19082:  SIRegister_TComponentStream(CL);
19083:  TypeS('TComponentStream', 'TMemoryStream;
19084:  TComponentFileFormat', '( cffText, cffBinary );
19085:  TComponentArray', 'array of TComponent;
19086:  TResourceNaming,rnClassNameTag,rnClassName,rnClassTag,rnNameTag,rnClass,rnName, rnTag );
19087:  Func VOID_COMP : ___Pointer;
19088:  Func VOID_FORM : ___Pointer;
19089:  Func CreateIoForm( AForm : TCustomForm; ClassType : TFormClass) : TCustomForm;
19090: Func GetComponentTree(Component:TComponent;pComponents:TComponentArray;bAddSelf:Bool):LongInt;
19091:  Func pPosEx( SearchStr: PChar; Str : PChar; var Pos : LongInt) : PChar;
19092:  Func pGetTextBetween(pBuff:PChar;bSearchCode:str;eSearchCode:str;Container:TStrings): LongInt;
19093:  Func pComponentToString( Component : TComponent) :Str;
19094:  //Func StringToComponent(Value:str; ComponentClass:TComponentClass): TComponent;
19095:  Func StringToObjectBinaryStream(Value:str;BinStream:TMemoryStream;ResName:str):Bool;
19096: Func ObjectBinaryStreamToString(BinStream:TMemoryStream;var sResult:str;bResource:Bool):Bool
19097:  Func ObjectTextToBinaryStream(StrStream,BinStream:TComponentStream;ObjCount:LongInt): LongInt;
19098:  Func
       ObjectBinaryStreamToObjectTextStream(BinStream:TMemoryStream;StrStream:TMemoryStream;bResource:Bool):Bool;
19099:  Func GetResHeaderInfo( Stream : TMemoryStream; sl : TStrings) :Bool;
19100:  Func pGetResourceName(Component:TComponent;NamingMethod: TResourceNaming) :Str;
19101: Func GetFormResourceStream(Form:TCustomForm;ResName:str;var ResourceStream:TMemoryStream):Bool;
19102:  Func WriteComponentsToFile(FormsAndComponents:array of
       TComponent;FileName:str;Format:TComponentFileFormat; StoreComponentNames:Boolean):Boolean;
19103:  Func ReadComponentsFromFile(FormsAndComponents:array of TComponent;FileName:str):Bool;
19104:  //Func ReadComponentsFromFile1(FileName:str;pComponents: array of TPComponent;ComponentClasses:array of
       TComponentClass):Bool;
19105:  Func ReadComponentTreeFromFile(FormOrComponent:TComponent;FileName:Str):Boolean;
19106:  Func
       WriteComponentToFile(Component:TComponent;FileName:str;Format:TComponentFileFormat;StoreComponentName:Bool):Bool;
19107:  Func
       WriteComponentTreeToFile(FormOrComponent:TComponent;FileName:str;Format:TComponentFileFormat;StoreComponentName:Bool):
19108:  Func ReadComponentFromFile4(FormOrComponent:TComponent; FileName :Str):Bool;;
19109:  Func ReadFormFromFile(pInstance:TComponent;FormClass:TFormClass;FileName:str): Bool;
19110:  Func ReadComponentResourceFile5( Instance : TObject; FileName :Str) :Bool;;
19111: Func WriteComponentResourceFile(nstance:TObj;FileName:str;StoreFormAsVisible:Bool):Bool;
19112:  Func ReadComponentsResourceFile(Components:array of TComponent;FileName:str;NamingMethod:TResourceNaming;
       bLoadTotalForm:Boolean):Boolean;
19113:  Func WriteComponentsResourceFile( Components : array of TComponent; FileName:Str; NamingMethod :
       TResourceNaming) :Bool;
19114:  Func ReadComponentResourceHeader(sHeaderInfo:TStrings; pSize : Int; FileName:Str; NamingMethod :
       TResourceNaming) :Bool;
19115:  Func ConvertComponentResourceToTextFile(SourceFileName:str;TargetFileName:str;bStoreResNames:Bool):Bool;
19116:  Func CheckComponentInResourceFile(Component:TComponent;FileName:str;NamingMethod:TResourceNaming):Boolean;
19117:  Func DeleteComponentFromResourceFile(ResourceName:str; FileName:str):Bool;;
19118:  Func DeleteComponentFromResourceFile8(Component:TComponent;FileName
       string;NamingMethod:TResourceNaming):Bool;
19119: //Func CreateFormFromResFile(AOwner:TComponent;NewClassType:TFormClass;FileName:str): Pointer;
19120:  //Func CreateComponentFromResFile( AOwner : TComponent; NewClassType : TComponentClass; FileName :Str) :
       Pointer;
19121: end;
19122:
19123: Proc SIRegister_SMScript(CL: TPSPascalCompiler);
19124: begin
19125:   CL.FindClass('TOBJECT'),'TSMScriptExecutor;
19126:   SIRegister_TSMSEError(CL);
19127:   ClassN(CL.FindClass('TOBJECT'),'TSMSEModule;
19128:   TypeS('TSMSEProcedureType', '( ptProcedure, ptFunc );
19129:   SIRegister_TSMSEProcedure(CL);
19130:   SIRegister_TSMSEProcedures(CL);
19131:   SIRegister_TSMSEModule(CL);
19132:   SIRegister_TSMSEModules(CL);
19133:   TypeS('TSMScriptLanguage', ( slCustom, slVBScript, slJavaScript );
19134:   SIRegister_TSMScriptExecutor(CL);
19135:  //DelphiFunction('Proc Register;
19136: end;
19137:
19138: Proc SIRegister_geometry2(CL: TPSPascalCompiler);
19139: begin
19140:   TPointF2', 'record X : Double; Y : Double; end;
19141:   TPoint3', 'record X : Int; Y : Int; Z : Int; end;
19142:   //TypeS('PPointF3', '^TPointF3 // will not work;
19143:   TPointF3', 'record X : double; Y : double; Z : Double; end;
19144:  Func AreLinesParallel( p1, p2, p3, p4 : TPoint) :Bool;;
19145:  Func AreLinesParallel1( p1, p2, p3, p4 : TPointF2; e : Double) :Bool;;
19146:  Func AreLinesParallel2( p1, p2, p3, p4, p5, p6 : TPoint3) :Bool;;
19147:  Func AreLinesParallel3( p1, p2, p3, p4, p5, p6 : TPointF3; e : Double) :Bool;;
```

```
19148:  Func IntersectLines( p1, p2, p3, p4 : TPoint) : TPoint;;
19149:  Func IntersectLines1( p1, p2, p3, p4 : TPointF2; e : Double) : TPointF2;;
19150:  Func AngleDifference( alpha, beta : Double) : Double;
19151: end;
19152:
19153:  source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
19154:  New Functions /Classes mX4:
19155:  Func DownloadJPGToBitmap(const URL :Str; ABitmap: TBitmap):Bool;);
19156:  Proc GetImageLinks(AURL:Str; AList: TStrings);;
19157:  Proc SaveByteCode;;
19158:  Proc ResetKeyPressed;;
19159:  Proc SetKeyPressed;;
19160:
19161:  RIRegister_HTTPProd_Routines(Exec);
19162:  Func ContentFromScriptStream(AStream:TStream; AWebModuleContext : TWebModuleContext; AStripParamQuotes :
       Bool; AHandleTag : THandleTagProc; AHandleScriptTag : THandledTagProc; const AScriptEngine :Str;
       ALocateFileService : ILocateFileService) :Str;
19163:  Func ContentFromScriptFile(const AFileName:TFileName; AWebModuleContext:TWebModuleContext;
       AStripParamQuotes :Bool; AHandleTag:THandleTagProc;AHandleScriptTag:THandledTagProc; '+
19164:                 const AScriptEngine:str; ALocateFileService:ILocateFileService) :Str;
19165:  Func FindComponentWebModuleContext( AComponent : TComponent) : TWebModuleContext;
19166:  Func GetTagID( const TagString :Str) : TTag;
19167:  Func ContentFromStream(AStream:TStream;
       AStripParamQuotes:Boolean;AHandleTag:THandleTagProc;AHandledTag:THandledTagProc):str;
19168:  Func ContentFromString(const AValue:str;
       AStripParamQuotes:Bool;AHandleTag:THandleTagProc;AHandledTag:THandledTagProc):str;
19169:  RIRegister_synacrypt_Routines(Exec);
19170:  Func TestDes:Bool;
19171:  {:Call internal test of all 3DES encryptions. Returns @true if all is OK.}
19172:  Func Test3Des:Bool;
19173:  {:Call internal test of all AES encryptions. Returns @true if all is OK.}
19174:  Func TestAes:Bool;
19175:  Proc LogMessage( const Fmt :Str; const Params : array of const);
19176:  Func UnixPathToDosPath2( const Path :Str) :Str;
19177:  Func DosPathToUnixPath2( const Path :Str) :Str;
19178:  Proc InitISAPIApplicationList;
19179:  Proc DoneISAPIAPplicationList;
19180:
19181:  RIRegister_xmldom_Routines(Exec);
19182:  Func IsPrefixed( const AName : DOMString) :Bool;
19183:  Func IsPrefixedW( const AName : DOMStringW) :Bool;
19184:  Func ExtractLocalName( const AName : DOMString) : DOMString;
19185:  Func ExtractLocalNameW( const AName : DOMStringW) : DOMStringW;
19186:  Func ExtractPrefixW( const AName : DOMStringW) : DOMStringW;
19187:  Func MakeNodeNameW( const Prefix, LocalName : DOMStringW) : DOMStringW;
19188:  Func ExtractPrefix( const AName : DOMString) : DOMString;
19189:  Func MakeNodeName( const Prefix, LocalName : DOMString) : DOMString;
19190:  Func SameNamespace(const Node:IDOMNode;const namespaceURI:WideString):Boolean;;
19191:  Func SameNamespace2( const URI1, URI2 : WideString) :Bool;;
19192:  Func NodeMatches(const Node:IDOMNode;const TagName,NamespaceURI:DOMString):Bool;;
19193:  Func GetDOMNodeEx( const Node : IDOMNode) : IDOMNodeEx;
19194:  Proc RegisterDOMVendor( const Vendor : TDOMVendor);
19195:  Proc UnRegisterDOMVendor( const Vendor : TDOMVendor);
19196:  Func GetDOMVendor( VendorDesc :Str) : TDOMVendor;
19197:  Func GetDOM( const VendorDesc :Str) : IDOMImplementation;
19198:  Proc DOMVendorNotSupported( const PropOrMethod, VendorName :Str);
19199:  Func IsValidLocale( Locale : LCID; dwFlags : DWORD) : BOOL;
19200:  Func ConvertDefaultLocale( Locale : LCID) : LCID;
19201:  Func GetThreadLocale : LCID;
19202:  Func SetThreadLocale( Locale : LCID) : BOOL;
19203:  Func GetSystemDefaultLangID : LANGID;
19204:  Func GetUserDefaultLangID : LANGID;
19205:  Func GetSystemDefaultLCID : LCID;
19206:  Func GetUserDefaultLCID : LCID;
19207:  Func AbsInt( const B : Int) : Int;
19208:  Func AbsFloat( const B : double) : extended;
19209:
19210:  Proc ReconcileDeltas(const cdsArray:array of TClientDataset;vDeltaArray:OleVariant);
19211:  Proc CDSApplyUpdates(ADatabase:TDatabase;var vDeltaArray:OleVariant;const
       vProviderArray:OleVariant;Local:Bool);
19212:  Function(@CheckMemory, 'Proc CheckMemory;;
19213:  Function(@GetMemoryInfo, 'Func getMemoryInfo;;
19214:  Function(@GetMemoryInfo, 'Func getMemInf;;
19215:  Proc RaiseLastWin32_2( const Text :Str);;
19216:
19217:  RIRegister_Gameboard_Routines(Exec);
19218:  Func Opponent( Player : TPlayer) : TPlayer;
19219:  Proc InitZobritsNumbers( var ZobristNumbers, Count : Int);
19220:  Proc SaveStringToStream( const Str :Str; Stream : TStream);
19221:  Func LoadStringFromStream( Stream : TStream) :Str;
19222:  Func WaitForSyncObject(SyncObject:THandle;Timeout:Card;BlockInput:Boolean):Card);
19223:  Func ProcessMessage :Bool;
19224:  Proc ProcessMessages( Timeout : DWORD);  //without application!
19225:
19226:  RIRegister_PersistSettings_Routines(Exec);
19227:  Proc SetStorageHandler( AFunc : TStorageHandlerFunction);;
19228:  Proc SetStorageHandler1( AMethod : TStorageHandlerMethod);;
19229:  Func GetStorage : TPersistStorage;
19230:  Proc SaveComponents( Root : TComponent; Storage : TPersistStorage);
```

```
19231:  Proc LoadComponents( Root : TComponent; Storage : TPersistStorage);
19232:  Proc AutoSave( Root : TComponent);
19233:  Proc AutoLoad( Root : TComponent);
19234:
19235:  Proc FloatToDecimalE(var Result:TFloatRec;const Value:extended;ValueType:TFloatValue;Precision,
        Decimals:Int);
19236:  Func FloatToTextE(BufferArg:PChar; const Value:extended; ValueType: FloatValue;Format:TFloatFormat;
        Precision, Digits: Int): Int;
19237:  Proc FloatToDecimalE(var Result:TFloatRec;const Value:extended;ValueType:TFloatValue;Precision,
        Decimals:Int);;
19238:  Func FloatToTextE(BufferArg:PChar;const
        Value:extended;ValueType:TFloatValue;Format:TFloatFormat;Precision,Digits:Int):Int;
19239:  Func GetSystemDefaultLCID : LCID;
19240:  Func GetUserDefaultLCID : LCID;
19241:  Func CreateMutex2(lpMutexAttributes:TObject;bInitialOwner:BOOL;lpName:PChar) : THandle;
19242:  Func CreateSemaphore2(lpSemaphoreAttributes:TObject;lInitialCount,
        lMaximumCount:Longint;lpName:PChar):THandle;
19243:  Func GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD) : BOOL;
19244:  Createmutex2 fhand:= OpenFileHandle(exepath+'maxbox4.exe')
19245:  Func getMatchString(arex, atext:Str):Str;
19246:  Func getLastInput: DWord;
19247:  Proc GetKLList(List: TStrings);;   //Keyboardlist2
19248:  Proc EnableCTRLALTDEL(YesNo :Bool);
19249:  Func LocalIP:Str;
19250:  Func IPAddrToName(IPAddr:Str):Str;
19251:  Func GetIPFromHost(const HostName:Str):Str;
19252:  function getIPFromHostName(hostName: string): string;
19253:  Func FindComputers(Computers : TStringList): DWORD;
19254:  Func GetWin32TypeLibList(var Lines: TStringList):Bool;
19255:  Func RecurseWin32(const R: TRegistry; const ThePath:Str;
19256:                          const TheKey:Str):Str;
19257:  Func RunAsAdmin(hWnd: HWND; filename:Str; Parameters:Str):Bool;
19258:  Ex.:     RunAsAdmin2(hinstance,'cmd.exe',+
19259:                          'Set OPENSSL_CONF='+OpenSSLPathF+'openssl.cfg');
19260:  writeln(getDosOutput('openssl.exe version',OpenSSLPath));
19261:  Proc SaveGraphicToStream(Graphic: TGraphic; Stream: TStream);
19262:  Func LoadGraphicFromStream(Stream: TStream): TGraphic;
19263:  Proc CopyStreamToFile(S: TStream; F: THandle);
19264:  Func BigPow(aone, atwo: Int):Str;
19265:  Proc GetPelsPerMeter(CONST  Bitmap: TBitmap;VAR xPelsPerMeter,yPelsPerMeter: Int);
19266:  Proc RainbowColor(CONST fraction:  Double; VAR R,G,B:  BYTE);
19267:  RIRegister_JclPCRE_Routines(S: TPSExec);
19268:  Proc InitializeLocaleSupport;
19269:  Proc TerminateLocaleSupport;
19270:  Func StrReplaceRegEx(const Subject, Pattern:Ansistr;Args:array of const):Ansistr;
19271:
19272: Proc SIRegister_VariantRtn(CL: TPSPascalCompiler);
19273: begin
19274: TProcReadElementValue','procedure(Value:Variant; IndexValue:Intarray; const HighBoundInd:Int; Var
        Continue:bool);
19275: TProcWriteElementValue','procedure(OldValue:Variant;IndexValue:Intarray;Var NewValue:Variant; Var
        Continue:bool);
19276:  //TProcReadElementValue=procedure(Value:Variant;IndexValue:array of Int; const HighBoundInd:Int;Var
        Continue:bool);
19277:  //TProcWriteElementValue=procedure(OldValue:Variant;IndexValue:array of Int;Var NewValue:Variant;Var
        Continue:bool); //
19278:  Func SafeVarArrayCreate(const Bounds:array of Int;VarType, DimCount: Int): Variant;
19279:  Func VarArrayGet2(constA:Variant;const Indice:array of Int;const HighBound:Int):Variant);
19280:  Proc VarArrayPut2(var A:Variant;const Value:Variant;const Indices:array of Int;const HighBound:Int);
19281:  Func CycleReadArray( vArray : Variant; CallBackProc : TProcReadElementValue) :Bool;
19282:  Func CycleWriteArray(var vArray:Variant;CallBackProc: TProcWriteElementValue):Bool;
19283:  Func CompareVarArray1( vArray1, vArray2 : Variant) :Bool;
19284:  Func EasyCompareVarArray1(vArray1,vArray2:Variant;HighBound:Int):Bool;
19285: end;
19286:
19287: Proc SIRegister_StdFuncs(CL: TPSPascalCompiler);
19288: begin
19289:   ClassN(CL.FindClass('TOBJECT'),'EParserError;
19290:   //TypeS('TCharSet', 'set of Char;
19291:  Func ConvertFromBase(sNum:Str; iBase: Int; cDigits:Str): Int;
19292:  Func ConvertToBase( iNum, iBase : Int; cDigits :Str) :Str;
19293:  Func EnsureSentenceTerminates( Sentence :Str; Terminator : Char) :Str;
19294:  Func FindTokenStartingAt(st:Str;var i:Int;TokenChars:TCharSet;TokenCharsInToken:Bool):Str;
19295:  Func GetDirectoryOfFile( FileName :Str) :Str;
19296:  Func GetDirOfFile( FileName :Str) :Str;
19297:  Func GetTempFile( FilePrefix :Str) :Str;
19298:  Func Icon2Bitmap( Icon : HICON) : HBITMAP;
19299:  Func Maxfib( n1, n2 : Int) : Int;
19300:  Func MaxD( n1, n2 : Double) : Double;
19301:  Func Minfib( n1, n2 : Int) : Int;
19302:  Func MinD( n1, n2 : Double) : Double;
19303:  Func RandomStringfib( iLength : Int) :Str;
19304:  Func RandomIntfib( iLow, iHigh : Int) : Int;
19305:  Func Soundexfib( st :Str) :Str;
19306:  Func StripStringfib( st :Str; CharsToStrip :Str) :Str;
19307:  Func ClosestWeekday( const d : TDateTime) : TDateTime;
19308:  Func Yearfib( d : TDateTime) : Int;
19309:  Func Monthfib( d : TDateTime) : Int;
19310:  Func DayOfYearfib( d : TDateTime) : Int;
```

```
19311:  Func DayOfMonth( d : TDateTime) : Int;
19312:  Func VarCoalesce( V1, V2 : Variant) : Variant;
19313:  Func VarEqual( V1, V2 : Variant) :Bool;
19314:  Proc WeekOfYearfib( d : TDateTime; var Year, Week : Int);
19315:  Func Degree10( Degree : Int) : double;
19316:  Func CompToStr( Value : comp) :Str;
19317:  Func StrToComp( const Value :Str) : comp;
19318:  Func CompDiv( A, B : comp) : comp;
19319:  Func CompMod( A, B : comp) : comp;
19320:  // TypeS('PComp', '^Comp // will not work;
19321: end;
19322:
19323: Proc SIRegister_RegUtils(CL: TPSPascalCompiler);
19324: begin
19325:  Proc DefWriteToRegistry(const OtherKeys,ParamNams:array of string;const Values:array of Variant);
19326:  Proc WriteToRegistry(aRootKey:HKEY;const OtherKeys,ParamNames:array of string;const Values:array of
       Variant);
19327:  Func ReadFromRegistry(aRootKey:HKEY;const OtherKeys,ParamNames: array of string): Variant;
19328:  Func DefReadFromRegistry( const OtherKeys, ParamNames : array of string) : Variant;
19329:  Func AllSubKey( aRootKey : HKEY; const ForPath : array of string) : Variant;
19330:  Func DefAllSubKey( const ForPath : array of string) : Variant;
19331:  Func SaveRegKey( const FileName :Str; const ForKey : array of string) :Bool;
19332:  Func LoadRegKey( const FileName :Str; const ForKey : array of string) :Bool;
19333:  Func AltSaveRegKey(const FileName:str;const ForKey: array of string) :Bool;
19334:  Func AltLoadRegKey(const FileName:str;const ForKey: array of string) :Bool;
19335:  Func GetKeyForParValue( const aRootKey, ParName, ParValue :Str) :Str;
19336: end;
19337:
19338: Func GetEulerPhi( n : int64) : int64;
19339:  Func isprime(f: int64):Bool;;
19340:  Func  DispositionFrom(const SQLText:str):TPoint;
19341:  Proc AllTables(const SQLText:str;FTables:Tstrings);
19342:  Func  TableByAlias(const SQLText,Alias:str):str;
19343:  Func  FullFieldName(const SQLText,FieldName:str):str;
19344:  Func  AddToWhereClause(const SQLText,NewClause:str):str;
19345:  Func  GetWhereClause(SQLText:str;N:Int;var StartPos,EndPos:Int ):str;
19346:  Func  WhereCount(SQLText:str):Int;
19347:  Func  GetOrderInfo(SQLText:str):variant;
19348:  Func  OrderStringTxt(SQLText:str; var StartPos,EndPos:Int ):str;
19349:  Func  PrepareConstraint(Src:Tstrings):str;
19350:  Proc DeleteEmptyStr(Src:Tstrings);
19351:  Func  NormalizeSQLText(const SQL:Str;MacroChar:Char):Str;
19352:  Func  CountSelect(const SrcSQL:str):str;
19353:  Func  GetModifyTable(const SQLText:str;AlreadyNormal:boolean):str;
19354: Func GetCharFromVKey(vkey: Word):Str;
19355: Func Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile:Str):Bool;
19356: Func IsObjectActive(ClassName:Str):Bool;
19357: Func GetActiveObject(ClassID:TGUID; anil:TObject; aUnknown:IUnknown):HRESULT;;
19358: Func RegisterOCX(FileName:Str):Bool;
19359: Func UnRegisterOCX(FileName:Str):Bool;
19360: Func RegisterServer2(const aDllFileName:Str; aRegister:Bool):Bool;
19361: Proc mIRCDDE(Service, Topic, Cmd:Str);
19362:             //mIRCDDE('mIRC', 'COMMAND', '/say Hallo von SwissDelphiCenter.ch;
19363: Func OpenIE(aURL:Str):Bool;
19364: Func XRTLIsInMainThread:Bool;
19365: Func IsInMainThread:Bool;
19366:
19367: Proc SIRegister_SpectraLibrary(CL: TPSPascalCompiler);
19368: begin
19369:  TypeS('Nanometers', 'Double;
19370:  ConstantN('WavelengthMinimum','LongInt').SetInt( 380);
19371:  ConstantN('WavelengthMaximum','LongInt').SetInt( 780);
19372:  Proc WavelengthToRGB( const Wavelength : Nanometers; var R, G, B : BYTE);
19373: end;
19374: Proc SIRegister_DrawFigures(CL: TPSPascalCompiler);
19375: begin
19376:  Proc DrawCube( const PantoGraph : TPantoGraph; const color : TColor);
19377:  Proc DrawSphere(const PantoGraph:TPantoGraph;const LatitudeColor,LongitudeColor:TColor; const
       LatitudeCircles,LongitudeSemicircles,PointsInCircle:WORD);
19378:  Proc DrawSurface( const PantoGraph : TPantoGraph);
19379:  Proc DrawFootballField(const PantoGraph:TPantoGraph;const ColorField,ColorLetters,ColorGoals:TColor);
19380: end;
19381: Proc SIRegister_synadbg(CL: TPSPascalCompiler);
19382: begin
19383:   SIRegister_TSynaDebug(CL);
19384:  Proc AppendToLog( const value : Ansistr);
19385: end;
19386:
19387: Proc SIRegister_XmlRpcCommon(CL: TPSPascalCompiler);
19388: begin
19389:   SIRegister_TRC4(CL);
19390:   TypeS(TRPCDataType', '( rpNone, rpString, rpInt, rpBoolean, rpDoub'
19391:    +le, rpDate, rpBase64, rpStruct, rpArray, rpName, rpError );
19392:  Func GetTempDirRPC :Str;
19393:  Func FileIsExpired( const FileName :Str; Elapsed : Int) :Bool;
19394:  Func EncodeEntities( const Data :Str) :Str;
19395:  Func DecodeEntities( const Data :Str) :Str;
19396:  Func ReplaceRPC(const Data:str;const Find:str;const Replace:str):Str;
19397:  Func InStr( Start : Int; const Data :Str; const Find :Str) : Int;
```

```
19398:  Func Mid( const Data :Str; Start : Int) :Str;
19399:  Func DateTimeToISO( ConvertDate : TDateTime) :Str;
19400:  Func IsoToDateTime( const ISOStringDate :Str) : TDateTime;
19401:  Func ParseStringRPC(const SearchString:str; Delimiter:Char; Substrings: TStrings; const
       AllowEmptyStrings:Bool; ClearBeforeParse:Boolean):Int;
19402:  Func ParseStream(
       SearchStream:TStream;Delimiter:Char;ubstrings:TStrings;AllowEmptyStrings:Bool;ClearBeforeParse:Bool): Int;
19403:  Func FixEmptyString( const Value :Str) :Str;
19404:  Func URLEncodeRPC( const Value :Str) :Str;
19405:  Func StreamToStringRPC( Stream : TStream) :Str;
19406:  Proc StringToStream( const Text :Str; Stream : TStream);
19407:  Func StreamToVariant( Stream : TStream) : OleVariant;
19408:  Proc VariantToStream( V : OleVariant; Stream : TStream);
19409:  Func Hash128AsHex( const Hash128Value : T4x4LongWordRecord) :Str;
19410: ConstantN(ValidURLChars','String').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$-_@.&+-!''*"(),;/#?:;
19411: end;
19412:
19413: Proc SIRegister_synafpc(CL: TPSPascalCompiler);
19414: begin
19415:  Func LoadLibraryfpc( ModuleName : PChar) : TLibHandle;
19416:  Func FreeLibraryfpc( Module : TLibHandle) : LongBool;
19417:  Func GetProcAddressfpc( Module : TLibHandle; Proc : PChar) : Pointer;
19418:  Func GetModuleFileNamefpc(Module:TLibHandle;Buffer:PChar; BufLen:Int):Int;
19419:   TypeS('TLibHandle', 'Int;
19420:   TypeS('TLibHandle2', 'HModule;
19421:   TypeS(LongWordfpc', DWord;
19422: Proc Sleepfpc( milliseconds :Card);
19423: end;
19424:
19425: Proc SIRegister_Spring_Utilsmx(CL: TPSPascalCompiler);
19426: begin
19427:   TypeS('TOSPlatformType', '( ptUnknown, ptWin3x, ptWin9x, ptWinNT );
19428:   SIRegister_TOperatingSystem(CL);
19429:   SIRegister_TEnvironmentClass(CL);
19430:   TypeS('Environment', 'TEnvironment;
19431:  Func ApplicationPath :Str;
19432:  Func ApplicationVersion : TVersion;
19433:  Func ApplicationVersionString :Str;
19434:  Func GetLastErrorMessage :Str;
19435:  function getLastSysErrorMessage: string;
19436:  Func CreateCallback( obj : TObject; methodAddress : Pointer) : TCallbackFunc;
19437:  Func ConvertFileTimeToDateTime(const fileTime:TFileTime;useLocalTimeZone:Bool):TDateTime;;
19438:  Func ConvertDateTimeToFileTime(const datetime:TDateTime;useLocalTimeZone:Bool):TFileTime;
19439:  {Proc Synchronize( threadProc : TThreadProcedure);
19440:  Proc Queue( threadProc : TThreadProcedure); }
19441:  Func TryGetPropInfo(instanc:TObject;const propertyName:str;out propInfo:PPropInfo):Bool;
19442:  Func IsCtrlPressed :Bool;
19443:  Func IsShiftPressed :Bool;
19444:  Func IsAltPressed :Bool;
19445:  Proc CheckFileExists( const fileName :Str);
19446:  Proc CheckDirectoryExists( const directory :Str);
19447:  ConstantN('COneKB','Int64').SetInt64( 1024);
19448:  COneMB',Int64').SetInt64( 1048576);
19449:  COneGB',Int64').SetInt64( 1073741824);
19450:  COneTB',Int64').SetInt64( 1099511627776);
19451:  Func TryConvertStrToDateTime(const s,format:str;out value:TDateTime):Boolean;;
19452:  Func ConvertStrToDateTime(const s, format:Str): TDateTime;;
19453: end;
19454:
19455: Proc SIRegister_rxOle2Auto(CL: TPSPascalCompiler);
19456: begin
19457:  ConstantN('MaxDispArgs','LongInt').SetInt( 64);
19458:  ConstantN('MaxDispArgs','LongInt').SetInt( 32);
19459:   ClassN(CL.FindClass('TOBJECT'),'EPropReadOnly;
19460:   ClassN(CL.FindClass('TOBJECT'),'EPropWriteOnly;
19461:   SIRegister_TOleController(CL);
19462:  Proc InitOLE2;
19463:  Proc DoneOLE;
19464:  Func OleInitialized :Bool;
19465:  Func MakeLangID( PrimaryLangID, SubLangID : Word) : Word;
19466:  Func MakeLCID( LangID : Word) : TLCID;
19467:  Func CreateLCID( PrimaryLangID, SubLangID : Word) : TLCID;
19468:  Func ExtractLangID( LCID : TLCID) : Word;
19469:  Func ExtractSubLangID( LCID : TLCID) : Word;
19470: Func CLSIDFromProgID(pszProgID: widestring; out clsid: TGUID): Longint'); //stdcall;
19471: Func ProgIDFromCLSID(const clsid: TGUID; out pszProgID: widestring): Longint'); //stdcall;
19472: Func StringFromCLSID(const clsid: TGUID; out psz: widestring): longint'); //stdcall;
19473: //{$EXTERNALSYM CLSIDFromString}
19474:  Func CLSIDFromString(psz: widestring; out clsid: TGUID): longint'); //stdcall;
19475:  Func StringFromGUID2(const guid: TGUID;psz: widestring;cbMax:Integer):Integer');//stdcall;
19476:  Func ProgIDExists(const ProgID:WideString):Boolean');
19477:  Proc CoUninitialize');
19478:  Proc OleUninitialize');
19479: end;
19480:
19481: Proc SIRegister_ulogifit(CL: TPSPascalCompiler);
19482: begin
19483: Proc LogiFit(X,Y:TVector;Lb,
       Ub:Int;ConsTerm:Boolean;General:Bool;MaxIter:Int;Tol:Float;B:TVector;V:TMatrix);
```

```
19484: Proc WLogiFit(X,Y,S:TVector;Lb,Ub:Int;ConsTerm Bool General:Bool;MaxIter
       Int;Tol:Float;B:TVector;V:TMatrix);
19485: Func LogiFit_Func( X : Float; B : TVector) : Float;
19486: end;
19487:
19488: Proc FormattedTextOut(TargetCanvas: TCanvas; const Rect : TRect; const Text :Str; Selected :Bool; Columns
       : TProposalColumns; Images : TImageList);
19489:  Func FormattedTextWidth(TargetCanvas:TCanvas;const
       Text:str;Columns:TProposalColumns;Images:TImageList):Int;
19490:  Func PrettyTextToFormattedString(const APrettyText:Str; AlternateBoldStyle:Boolean):str;
19491:
19492:  Proc SIRegister_ulinfit(CL: TPSPascalCompiler);
19493: begin
19494:  Proc LinFit( X, Y : TVector; Lb, Ub : Int; B : TVector; V : TMatrix);
19495:  Proc WLinFit( X, Y, S : TVector; Lb, Ub : Int; B : TVector; V : TMatrix);
19496:  Proc SVDLinFit(X,Y:TVector; Lb, Ub:Int; SVDTol : Float; B: TVector; V : TMatrix);
19497:  Proc WSVDLinFit(X,Y,S: TVector; Lb,Ub:Int;SVDTol: Float; B: TVector; V : TMatrix);
19498:   Proc SVDFit(X:TMatrix;Y:TVector;Lb,Ub Nvar:Int;ConsTerm:Bool; SVDTol: Float;B: TVector; V: TMatrix);
19499:  Proc WSVDFit(X: TMatrix;Y,S:TVector;Lb Ub,Nvar:Int;ConsTerm:Boolean;SVDTol:Float; B: TVector;V : TMatrix);
19500: end;
19501:
19502: Proc SIRegister_MaxUtils(CL: TPSPascalCompiler);
19503: begin
19504:   TypeS('MaxCharSet', 'set of Char;
19505:  Func GetMachineNamemax :Str;
19506:  Func GetModuleNamemax( HModule : THandle) :Str;
19507:  Func TrimChars( const S :Str; Chars : MaxCharSet) :Str;
19508:  Func TickCountToDateTime( Ticks :Card) : TDateTime;
19509:  Proc OutputDebugStringmax( const S :Str);
19510:  Proc OutputDebugFormat( const FmtStr :Str; Args : array of const);
19511:  Func IsAppRunningInDelphi :Bool;
19512: Proc ParseFields(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strigs:TStrings;Decode: Bool;
19513:  Func HTTPDecodemax( const AStr :Str) :Str;
19514:  Func HTTPEncodemax( const AStr :Str) :Str;
19515:  Func FormatDate( const DateString :Str) :Str;
19516:  Func FormatListMasterDate(const DateStr,FormatDefStr:str; Len:Int):str;
19517:  Func InvertCase( const S :Str) :Str;
19518:  Func CommentLinesWithSlashes( const S :Str) :Str;
19519:  Func UncommentLinesWithSlashes( const S :Str) :Str;
19520:  Func StripChars( const S :Str; Strip : CharSet) :Str;
19521:  Func TrimChars( const S :Str; Chars : CharSet) :Str;
19522:  Func TrimLeftChars( const S :Str; Chars : CharSet) :Str;
19523:  Func TrimRightChars( const S :Str; Chars : CharSet) :Str;
19524:  Func ContainsChars( const S :Str; Strip : CharSet) :Bool;
19525:  Func DequotedStrmax( const S :Str; AQuoteChar : Char) :Str;
19526:  Proc LeftPadStr( var S :Str; toLength : Int; withChar : Char);
19527:  Proc RightPadStr( var S :Str; toLength : Int; withChar : Char);
19528:  Func RemoveChars( S :Str; Chars : CharSet) :Str;
19529:  Func FilterChars( S :Str; Chars : CharSet) :Str;
19530:  Func RemoveNonNumericChars( S :Str) :Str;
19531:  Func RemoveNonAlphanumChars( S :Str) :Str;
19532:  Func RemoveNonAlphaChars( S :Str) :Str;
19533:  Func HasAlphaChars( S :Str) :Bool;
19534:  Func ReplaceChars( S :Str; Chars : CharSet; ReplaceWith : Char) :Str;
19535:  Func DomainOfEMail( const EMailAddress :Str) :Str;
19536:  Func IPToHexIP( const IP :Str) :Str;
19537:  Proc CmdLineToStrings( S : Ansistr; const List : TStrings);
19538:  BASE2','String').SetString( '01;
19539:  ConstantN('BASE10','String').SetString(0123456789;
19540:  ConstantN('BASE16','String').SetString(0123456789ABCDEF;
19541:  ConstantN('BASE36','String').SetString(0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ;
19542:  ConstantN('BASE62','String').SetString(0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;
19543:  Func BaseConvert( Number, FromDigits, ToDigits :Str) :Str;
19544: Func ValidXmlName( AName : PChar; ASize : Int) :Bool;;
19545: Func ValidXmlName1( const AName :Str) :Bool;;
19546:  Func EncodeXmlAttrValue( const AStr : Ansistr) : Ansistr;;
19547:  Proc EncodeXmlAttrValue3(ABuff:PChar;ABuffSize:Int;var AStr:AnsiStr;var ALen,AOffset:Int);
19548:  Proc EncodeXmlAttrValue4(const ASource:Str;var ADest:str;var ALen,AOffset:Int);
19549:  Func EncodeXmlString5( const AStr :Str) :Str;;
19550:  Proc EncodeXmlString6(ABuff:PChar;ABuffSize:Int;var AStr:Ansistr;var ALen,AOffset:Int;
19551:  Proc EncodeXmlComment(const ASource:Ansistr;var ADest:Ansistr;var ALen,AOffset:Int);
19552:  Func HasEncoding( const AStr : Ansistr) :Bool;
19553:  Func DecodeXmlAttrValue( const AStr :Str) :Str;
19554:  Proc ReallocateString(var AString:Ansistr; var ALen : Int; AReqLen : Int);
19555:  Proc AttrFillXMLString(AnAttr:IAttribute;var aString:Ansistr;var aOffset,aLen : Int);
19556:  Proc FillXMLString(ANode:INode;var AStr:Str;var AOffset,ALen:Int;ASiblig:INode;ALevel:Int)
19557:  Func NodeToXML( ANode : INode) :Str;
19558:  Proc XMLSaveToFile( ANode : INode; const AFileName :Str);
19559:  Func XMLLoadFromFile( const AFileName :Str) : INode;
19560:  Func Hashmax( const ASource : Ansistr) :Card;
19561:  ConstantN('GXMLIndentSpaces','Int').SetInt( 2);
19562:  //ConstantN('GXMLMultiLineAttributes','Boolean')BoolToStr( True);
19563:  GXMLMultiLineAttributeThreshold','Int').SetInt( 7);
19564: end;
19565:
19566: Proc SIRegister_MaxDOMDictionary(CL: TPSPascalCompiler);
19567: begin
19568:   SIRegister_IDictionary(CL);
19569:   SIRegister_TDictionary(CL);
```

```
19570:  Func HashFast( const AKey :Str) :Card;
19571:  Func HashCarlos( const AKey :Str) :Card;
19572:  Func BorlandHashOf( const AKey :Str) :Card;
19573:  Func HashSumOfChars( const AKey :Str) :Card;
19574: end;
19575:
19576: Proc SIRegister_MaxDOM(CL: TPSPascalCompiler);
19577: begin
19578:   TypeS('TNodeType', '( ntElement, ntText, ntCDATA, ntComment );
19579:   Interface(CL.FindInterface('IUNKNOWN'),INode, 'INode;
19580:   Interface(CL.FindInterface('IUNKNOWN'),IAttribute, 'IAttribute;
19581:   Interface(CL.FindInterface('IUNKNOWN'),IAttributeCollection, 'IAttributeCollection;
19582:   Interface(CL.FindInterface('IUNKNOWN'),INodeCollection, 'INodeCollection;
19583:   SIRegister_INode(CL);
19584:   SIRegister_IAttribute(CL);
19585:   SIRegister_IAttributeCollection(CL);
19586:   SIRegister_INodeCollection(CL);
19587:  Func NodeCreate( const ANodeName :Str; ANodeType : TNodeType) : INode;
19588:   SIRegister_TAttribute(CL);
19589:   TypeS(TNodes', 'array of INode;
19590:   TypeS(TAttributes2', 'array of IAttribute;
19591:   TypeS(THashedAttributes', 'record Attributes : TAttributes2; AttrCount '
19592:    +': Int; AttrCapacity : Int; end;
19593:   TypeS(TAttrHashTable', 'array of THashedAttributes;
19594:   SIRegister_TNode(CL);
19595:   //TypeS('TNodeClass', 'class of TNode;
19596:   //TypeS('TAttributeClass', 'class of TAttribute;
19597:  Func PointerToStr( P : ___Pointer) :Str;
19598:  Func StrToPointer( const S :Str) : ___Pointer;
19599:  Func INodeToStr( ANode : INode) :Str;
19600:  Func StrToINode( const S :Str) : INode;
19601:  Func CompareByNodeName( N1, N2 : INode) : Int;
19602:  Func CompareByNameAttr( N1, N2 : INode) : Int;
19603: end;
19604:
19605: Proc SIRegister_cASN1(CL: TPSPascalCompiler);
19606: begin
19607:   ClassN(CL.FindClass('TOBJECT'),'EASN1;
19608:  'ASN1_ID_END_OF_CONTENT','LongWord').SetUInt( $00);
19609:  ConstantN('ASN1_ID_BOOLEAN','LongWord').SetUInt( $01);
19610:  'ASN1_ID_Int','LongWord').SetUInt( $02);
19611:  'ASN1_ID_BIT_STRING','LongWord').SetUInt( $03);
19612:  'ASN1_ID_OCTET_STRING','LongWord').SetUInt( $04);
19613:  'ASN1_ID_NULL','LongWord').SetUInt( $05);
19614:  'ASN1_ID_OBJECT_IDENTIFIER','LongWord').SetUInt( $06);
19615:  'ASN1_ID_OBJECT_DESCRIPTOR','LongWord').SetUInt( $07);
19616:  'ASN1_ID_EXTERNAL','LongWord').SetUInt( $08);
19617:  'ASN1_ID_REAL','LongWord').SetUInt( $09);
19618:  'ASN1_ID_ENUMERATED','LongWord').SetUInt( $0A);
19619:  'ASN1_ID_EMBEDDED_PDV','LongWord').SetUInt( $0B);
19620:  'ASN1_ID_UTF8STRING','LongWord').SetUInt( $0C);
19621:  'ASN1_ID_RELATIVE_OID','LongWord').SetUInt( $0D);
19622:  'ASN1_ID_NUMERICSTRING','LongWord').SetUInt( $12);
19623:  'ASN1_ID_PRINTABLESTRING','LongWord').SetUInt( $13);
19624:  'ASN1_ID_T61STRING','LongWord').SetUInt( $14);
19625:  'ASN1_ID_VIDEOTEXSTRING','LongWord').SetUInt( $15);
19626:  'ASN1_ID_IA5STRING','LongWord').SetUInt( $16);
19627:  'ASN1_ID_UTCTIME','LongWord').SetUInt( $17);
19628:  'ASN1_ID_GENERALIZEDTIME','LongWord').SetUInt( $18);
19629:  'ASN1_ID_GRAPHICSTRING','LongWord').SetUInt( $19);
19630:  'ASN1_ID_VISIBLESTRING','LongWord').SetUInt( $1A);
19631:  'ASN1_ID_GENERALSTRING','LongWord').SetUInt( $1B);
19632:  'ASN1_ID_UNIVERSALSTRING','LongWord').SetUInt( $1C);
19633:  'ASN1_ID_CHARACTERSTRING','LongWord').SetUInt( $1D);
19634:  'ASN1_ID_BMPSTRING','LongWord').SetUInt( $1E);
19635:  'ASN1_ID_SEQUENCE','LongWord').SetUInt( $30);
19636:  'ASN1_ID_SET','LongWord').SetUInt( $31);
19637:  'ASN1_ID_CONSTRUCTED','LongWord').SetUInt( $20);
19638:  'ASN1_ID_APPLICATION','LongWord').SetUInt( $40);
19639:  'ASN1_ID_CONTEXT_SPECIFIC','LongWord').SetUInt( $80);
19640:  'ASN1_ID_PRIVATE','LongWord').SetUInt( $C0);
19641:   TASN1ObjectIdentifier', 'array of Int;
19642:   TASN1ParseProc', 'procedure(const TypeID:Byte; const DataBuf:Str; const DataSize: Int; const ObjectIdx:
       Int; const CallerData: Int);;
19643: {TASN1ParseProc =
19644:    Proc (const TypeID: Byte; const DataBuf; const DataSize: Int;
19645:                const ObjectIdx: Int; const CallerData: Int); }
19646:
19647:  Proc ASN1OIDInit( var A : TASN1ObjectIdentifier; const B : array of Int);
19648:  Func ASN1OIDToStr( const A : TASN1ObjectIdentifier) : Ansistr;
19649:  Func ASN1OIDEqual(const A:TASN1ObjectIdentifier; const B : array of Int) :Bool;
19650:  Func ASN1EncodeLength( const Len : Int) : Ansistr;
19651:  Func ASN1EncodeObj( const TypeID : Byte; const Data : Ansistr) : Ansistr;
19652:  Func ANS1EncodeEndOfContent : Ansistr;
19653:  Func ASN1EncodeNull : Ansistr;
19654:  Func ASN1EncodeBoolean( const A :Bool) : Ansistr;
19655:  Func ASN1EncodeDataInt8( const A : ShortInt) : Ansistr;
19656:  Func ASN1EncodeDataInt16( const A : SmallInt) : Ansistr;
19657:  Func ASN1EncodeDataInt24( const A : LongInt) : Ansistr;
```

```
19658:  Func ASN1EncodeDataInt32( const A : LongInt) : Ansistr;
19659:  Func ASN1EncodeDataInt64( const A : Int64) : Ansistr;
19660:  Func ASN1EncodeInt8( const A : ShortInt) : Ansistr;
19661:  Func ASN1EncodeInt16( const A : SmallInt) : Ansistr;
19662:  Func ASN1EncodeInt24( const A : LongInt) : Ansistr;
19663:  Func ASN1EncodeInt32( const A : LongInt) : Ansistr;
19664:  Func ASN1EncodeInt64( const A : Int64) : Ansistr;
19665:  Func ASN1EncodeIntBuf( const A :Str; const Size : Int) : Ansistr;
19666:  Func ASN1EncodeIntBufStr( const A : Ansistr) : Ansistr;
19667:  Func ASN1EncodeEnumerated( const A : Int64) : Ansistr;
19668:  Func ASN1EncodeBitString(const A: Ansistr; const UnusedBits : Byte): Ansistr;
19669:  Func ASN1EncodeOctetString( const A : Ansistr) : Ansistr;
19670:  Func ASN1EncodeInt32AsOctetString( const A : LongInt) : Ansistr;
19671:  Func ASN1EncodeUTF8String( const A : Ansistr) : Ansistr;
19672:  Func ASN1EncodeIA5String( const A : Ansistr) : Ansistr;
19673:  Func ASN1EncodeVisibleString( const A : Ansistr) : Ansistr;
19674:  Func ASN1EncodeNumericString( const A : Ansistr) : Ansistr;
19675:  Func ASN1EncodePrintableString( const A : Ansistr) : Ansistr;
19676:  Func ASN1EncodeTeletexString( const A : Ansistr) : Ansistr;
19677:  Func ASN1EncodeUniversalString( const A : WideString) : Ansistr;
19678:  Func ASN1EncodeBMPString( const A : WideString) : Ansistr;
19679:  Func ASN1EncodeUTCTime( const A : TDateTime) : Ansistr;
19680:  Func ASN1EncodeGeneralizedTime( const A : TDateTime) : Ansistr;
19681:  Func ASN1EncodeOID( const OID : array of Int) : Ansistr;
19682:  Func ASN1EncodeSequence( const A : Ansistr) : Ansistr;
19683:  Func ASN1EncodeSet( const A : Ansistr) : Ansistr;
19684:  Func ASN1EncodeContextSpecific( const I : Int; const A : Ansistr) : Ansistr;
19685:  Func ASN1DecodeLength(const Buf:Str;const Size:Int;var Len:Int) : Int;
19686:  Func ASN1DecodeObjHeader(const Buf:str;const Size:Int;var TypeID:Byte;var Len:Int;var Data:str):Int;
19687:  Func ASN1TypeIsConstructedType( const TypeID : Byte) :Bool;
19688:  Func ASN1TypeIsContextSpecific( const TypeID : Byte; var Idx : Int) :Bool;
19689:  Func ASN1DecodeDataBoolean(const Buf:str;const Size:Int; var A:Boolean) : Int;
19690:  Func ASN1DecodeDataInt32(const Buf:str;const Size:Int;var A:LongInt): Int;
19691:  Func ASN1DecodeDataInt64(const Buf:str; const Size:Int; var A:Int64): Int;
19692:  Func ASN1DecodeDataIntBuf(const Buf:str; const Size:Int;var A: Ansistr): Int;
19693:  Func ASN1DecodeDataBitString(const Buf:str; const Size:Int;var A:Ansistr;var UnusedBits:Byte): Int;
19694:  Func ASN1DecodeDataRawAnsistr(const Buf:str;const Size:Int;var A:Ansistr):Int;
19695:  Func ASN1DecodeDataOctetString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19696:  Func ASN1DecodeDataIA5String(const Buf:str;const Size:Int;var A:Ansistr): Int;
19697:  Func ASN1DecodeDataVisibleString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19698:  Func ASN1DecodeDataNumericString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19699:  Func ASN1DecodeDataPrintableString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19700:  Func ASN1DecodeDataTeletexString(const Buf:str;const Size:Int; var A:Ansistr):Int;
19701:  Func ASN1DecodeDataUTF8String(const Buf:str; const Size:Int;var A:Ansistr):Int;
19702:  Func ASN1DecodeDataUniversalString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19703:  Func ASN1DecodeDataBMPString(const Buf:Str; const Size:Int; var A: Ansistr):Int;
19704:  Func ASN1DecodeDataOID(const Buf:str;const Size:Int;var A:TASN1ObjectIdentifier):Int;
19705:  Func ASN1DecodeDataUTCTime(const Buf:str; const Size:Int;var A TDateTime): Int;
19706:  Func ASN1DecodeDataGeneralizedTime(const Buf:str;const Size:Int;var A:TDateTime):Int;
19707:  Func ASN1Parse(const Buf:str;const Size:Int;const ParseProc:TASN1ParseProc;const CallerData:Int):Int;
19708: Func ASN1DecodeBoolean(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var A:Bool):Int;
19709: Func ASN1DecodeInt32( const TypeID:Byte;const DataBuf:str;const DataSize: Int; var A : LongInt): Int;
19710: Func ASN1DecodeInt64(const TypeID:Byte;const DatBuf:str;const DataSize:Int;var A:Int64):Int;
19711: Func ASN1DecodeIntBuf(const TypeID:Byte;const DataBuf:Str; const DataSize: Int;var A : Ansistr):Int;
19712:  Func ASN1DecodeBitString(const TypeID:Byte;const DataBuf:str; const DataSize:Int;var A:Ansistr;var
       UnusedBits:Byte):Int;
19713:  Func ASN1DecodeString(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var A:Ansistr):Int;
19714:  Func ASN1DecodeOID(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var
       A:TASN1ObjectIdentifier):Int;
19715:  Func ASN1DecodeTime const TypeID:Byte;const DataBuf:str; const DataSize:Int;var A:TDateTime): Int;
19716:  Proc SelfTestASN1;
19717: end;
19718:
19719: Proc SIRegister_cX509Certificate(CL: TPSPascalCompiler);
19720: begin
19721:   ClassN(CL.FindClass('TOBJECT'),'EX509;
19722:   TypeS(TX509AttributeType', 'TASN1ObjectIdentifier;
19723:   TypeS(TX509AttributeValue', 'Ansistr;
19724:   TypeS(TX509AttributeTypeAndValue', 'record AType : TX509AttributeType;'
19725:    +' Value : TX509AttributeValue; _Decoded :Bool; end;
19726:   //TypeS('PX509AttributeTypeAndValue', '^TX509AttributeTypeAndValue // wil not work;
19727:  Proc InitX509AttributeTypeAndValue(var A : TX509AttributeTypeAndValue; const AType : TX509AttributeType;
       const Value : TX509AttributeValue);
19728:  Proc InitX509AtName( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19729:  Proc InitX509AtSurname( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19730:  Proc InitX509AtGivenName( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19731:  Proc InitX509AtInitials( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19732:  Proc InitX509AtGenerationQuailifier(var A:TX509AttributeTypeAndValue; const B:Ansistr);
19733:  Proc InitX509AtCommonName(var A: TX509AttributeTypeAndValue; const B : Ansistr);
19734:  Proc InitX509AtLocailityName(var A:TX509AttributeTypeAndValue;const B: Ansistr);
19735:  Proc InitX509AtStateOrProvince(var A:TX509AttributeTypeAndValue;const B:Ansistr);
19736:  Proc InitX509AtOriganizationName(var A: TX509AttributeTypeAndValue; const B:Ansistr);
19737:  Proc InitX509AtOriganizationUnitName(var A:TX509AttributeTypeAndValue;const B:Ansistr);
19738:  Proc InitX509AtTitle( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19739:  Proc InitX509AtDnQualifier(var A: TX509AttributeTypeAndValue; const B: Ansistr);
19740:  Proc InitX509AtCountryName(var A :TX509AttributeTypeAndValue; const B : Ansistr);
19741:  Proc InitX509AtEmailAddress(var A: TX509AttributeTypeAndValue; const B: Ansistr);
19742:  Func EncodeX509AttributeTypeAndValue(const A:TX509AttributeTypeAndValue):Ansistr;
19743:  Func DecodeX509AttributeTypeAndValue(const TypeID  Byte;const Buf:str; const Size:Int;var
       A:TX509AttributeTypeAndValue): Int;
```

```
19744:   TypeS('TX509RelativeDistinguishedName', 'array of TX509AttributeTypeAndValue;
19745:  // TypeS('PX509RelativeDistinguishedName','^TX509RelativeDistinguishedName //will not work;
19746:  Proc AppendX509RelativeDistinguishedName( var A : TX509RelativeDistinguishedName; const V :
       TX509AttributeTypeAndValue);
19747:  Func EncodeX509RelativeDistinguishedName(const A:TX509RelativeDistinguishedName):AnsiStr;
19748:  Func DecodeX509RelativeDistinguishedName(const TypeID:Byte; const Buf:Str; const Size:Int;var
       A:TX509RelativeDistinguishedName): Int;
19749:  TypeS('TX509RDNSequence', 'array of TX509RelativeDistinguishedName;
19750:   //TypeS('PX509RDNSequence', '^TX509RDNSequence // will not work;
19751:  TypeS('TX509Name', 'TX509RDNSequence;
19752:  Proc AppendX509RDNSequence(var A:TX509RDNSequence;const B:TX509RelativeDistinguishedName);
19753:  Func EncodeX509RDNSequence( const A : TX509RDNSequence) : Ansistr;
19754: Func DecodeX509RDNSequence(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RDNSequence):Int;
19755:  Func EncodeX509Name( const A : TX509Name) : Ansistr;
19756: Func DecodeX509Name(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Name):Int;
19757:  'CurrentX509Version','string').SetString(' X509v3;
19758: type
19759:  TX509Version = (
19760:      X509v1 = 0,
19761:      X509v2 = 1,
19762:      X509v3 = 2,
19763:      X509vUndefined = $FF // implementation defined value   );
19764:  TX509Version', '(X509v1, X509v2,X509v3, X509vUndefined);
19765:
19766: type
19767:  TX509GeneralNameType = (
19768:  gnOtherName                 = 0,
19769:  gnRFC822Name                = 1,
19770:  gnDNSName                   = 2,
19771:  gnX400Address               = 3,
19772:  gnDirectoryName             = 4,
19773:  gnEDIPartyName              = 5,
19774:  gnUniformResourceIdentifier = 6,
19775:  gnIPAddress                 = 7,
19776:  gnRegisteredID              = 8);
19777: TypeS(TX509GeneralNameType',(gnOtherName,gnRFC822Name,gnDNSName,gnX400Address,gnDirectoryName,
       gnEDIPartyName,gnUniformResourceIdentifier,gnIPAddress,gnRegisteredID);
19778:
19779:  Proc InitX509Version( var A : TX509Version);
19780:  Func EncodeX509Version( const A : TX509Version) : Ansistr;
19781:  Func DecodeX509Version(const TypeID:Byte;const Buf:str;const Size:Int;var A: TX509Version):Int;
19782:   TypeS(TX509Time', TDateTime;
19783:  Func EncodeX509Time( const A : TX509Time) : Ansistr;
19784:  Func DecodeX509Time(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Time):Int;
19785:   TX509Validity','record NotBefore:TX509Time; NotAfter:TX509Time; _Decoded:Bool; end;
19786:  // TypeS('PX509Validity', '^TX509Validity // will not work;
19787:  Func EncodeX509Validity( const A : TX509Validity) : Ansistr;
19788:  Func DecodeX509Validity(const TypeID:Byte;const Buf:Str; const Size:Int; var A: TX509Validity): Int;
19789:   TypeS('TX509DHValidationParms','record Seed: Ansistr; PgenCounter : Int; end;
19790:   //TypeS('PX509DHValidationParms', '^TX509DHValidationParms // will not work;
19791:   TypeS('TX509DHDomainParameters', record P : Ansistr; G : Ansistr'
19792:    +; Q : Ansistr; J : Ansistr; ValidationParms : TX509DHValidationParms; end;
19793:  // TypeS('PX509DHDomainParameters', '^TX509DHDomainParameters // will not work;
19794:  Func EncodeX509DHValidationParms( const A : TX509DHValidationParms) : Ansistr;
19795:  Func DecodeX509DHValidationParms(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509DHValidationParms): Int;
19796:  Func EncodeX509DHDomainParameters( const A: TX509DHDomainParameters) : Ansistr;
19797:  Func DecodeX509DHDomainParameters(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509DHDomainParameters):Int;
19798:   TypeS('TX509DSSParms', 'record P : Ansistr; Q : Ansistr; G : Ansistr; end;
19799:   //TypeS('PX509DSSParms', '^TX509DSSParms // will not work;
19800:  Func EncodeX509DSSParms( const A : TX509DSSParms) : Ansistr;
19801:  Func DecodeX509DSSParms(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509DSSParms):Int;
19802: TypeS('TX509DSSSigValue', 'record R : Ansistr; S : Ansistr; end;
19803:   //TypeS('PX509DSSSigValue', '^TX509DSSSigValue // will not work;
19804:  Func EncodeX509DSSSigValue( const A : TX509DSSSigValue) : Ansistr;
19805:  Func DecodeX509DSSSigValue(const TypeID:Byte; const Buf:Str; const Size:Int; var A:TX509DSSSigValue):Int;
19806:   TypeS(TX509AlgorithmIdentifier','record Algorithm : TASN1ObjectIdenti'
19807:    +fier; Parameters : Ansistr; _Decoded :Bool; end;
19808: //TypeS('PX509AlgorithmIdentifier', '^TX509AlgorithmIdentifier // will not work;
19809:  Proc InitX509AlgorithmIdentifier(var A:TX509AlgorithmIdentifier; const Algorithm : array of Int; const
       Parameters : Ansistr);
19810:  Proc InitX509AlgorithmIdentifierDSA_SHA1(var A:TX509AlgorithmIdentifier;const Params:AnsiStr;
19811:  Func EncodeX509AlgorithmIdentifier( const A : TX509AlgorithmIdentifier) : Ansistr;
19812:  Func DecodeX509AlgorithmIdentifier(const TypeID:Byte; const Buf:str; const Size:Int; var
       A:TX509AlgorithmIdentifier): Int;
19813:  TypeS(TX509RSAPublicKey','record Modulus:Ansistr;PublicExponent:Ansistr; end;
19814:  //TypeS('PX509RSAPublicKey', '^TX509RSAPublicKey // will not work;
19815:  Func EncodeX509RSAPublicKey( const A : TX509RSAPublicKey) : Ansistr;
19816:  Func DecodeX509RSAPublicKey(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RSAPublicKey): Int;
19817:  Func ParseX509RSAPublicKey(const Buf:str;const Size:Int;var A:TX509RSAPublicKey):Int);
19818:   TypeS('TX509DHPublicKey', 'Ansistr;
19819:  Func EncodeX509DHPublicKey( const A : TX509DHPublicKey) : Ansistr;
19820:   TypeS(TX509DSAPublicKey', 'Ansistr;
19821:  Func EncodeX509DSAPublicKey( const A : TX509DSAPublicKey) : Ansistr;
19822:   TypeS(TX509SubjectPublicKeyInfo', 'record Algorithm : TX509AlgorithmId'
19823:    +entifier; SubjectPublicKey : Ansistr; _Decoded :Bool; end;
19824:  //TypeS('PX509SubjectPublicKeyInfo', '^TX509SubjectPublicKeyInfo // will not work;
19825:  Proc InitX509SubjectPublicKeyInfoDSA(var A:TX509SubjectPublicKeyInfo;const B:TX509DSSParms;const
       PublicKey: Ansistr);
```

```
19826:  Func EncodeX509SubjectPublicKeyInfo( const A: TX509SubjectPublicKeyInfo) : Ansistr;
19827:  Func DecodeX509SubjectPublicKeyInfo(const TypeID:Byte;const Buf:str;const Size:Int;var A
       TX509SubjectPublicKeyInfo): Int;
19828:  Func ParseX509SubjectPublicKeyInfo(const Buf:str;const Size:Int;var A:TX509SubjectPublicKeyInfo):Int;
19829:  Func EncodeX509GeneralName(const A:TX509GeneralNameType;const EncodedName:Ansistr):Ansistr;
19830:  Func DecodeX509GeneralName(const TypeID:Byte; const Buf:str; const Size:Int;var
       A:TX509GeneralNameType;var B  Ansistr): Int;
19831:  TX509GeneralNames','array of record NameType:TX509GeneralNameType; Name : Ansistr; end;
19832:   //TypeS('PX509GeneralNames', '^TX509GeneralNames // will not work;
19833:  Func EncodeX509GeneralNames( const A : TX509GeneralNames) : Ansistr;
19834:  Func DecodeX509GeneralNames(const TypeID:Byte;const Buf:str; const Size:Int; var A:TX509GeneralNames):Int;
19835:   TX509BasicConstraints,record CA:Bool;PathLenConstraint:Ansistr;_DecodedCA:Boolean;end
19836:   //TypeS('PX509BasicConstraints', '^TX509BasicConstraints // will not work;
19837:  Func EncodeX509BasicConstraints( const A : TX509BasicConstraints) : Ansistr;
19838:  Func DecodeX509BasicConstraints(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509BasicConstraints): Int;
19839:  TypeS(TX509AuthorityKeyIdentifier', 'record KeyIdentifier : Ansistr'
19840:  +; AuthorityCertIssuer : TX509GeneralNames; AuthorityCertSerialNumber : Int64; end;
19841:  //TypeS('PX509AuthorityKeyIdentifier','^TX509AuthorityKeyIdentifier //will not work;
19842: Func EncodeX509AuthorityKeyIdentifier(const A:TX509AuthorityKeyIdentifier): Ansistr;
19843:  TypeS(TX509SubjectKeyIdentifier', 'Ansistr;
19844:  TypeS('TX509KeyUsage', 'Ansistr;
19845:  TypeS(TX509Extension', 'record ExtnID : TASN1ObjectIdentifier; Critica'
19846:   +l :Bool; ExtnValue: Ansistr;_DecodedCritical:Bool; _Decoded :Bool; end;
19847:  //TypeS('PX509Extension', '^TX509Extension // will not work;
19848:  Proc InitX509ExtAuthorityKeyIdentifier(var A: TX509Extension; const B TX509AuthorityKeyIdentifier);
19849:  Proc InitX509ExtSubjectKeyIdentifier(var A:TX509Extension; const B:TX509SubjectKeyIdentifier);
19850:  Proc InitX509ExtBascConstraints(var A:TX509Extension;const B:TX509BasicConstraints);
19851:  Func EncodeX509Extension( const A : TX509Extension) : Ansistr;
19852:  Func DecodeX509Extension(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Extension):Int;
19853:  TypeS('TX509Extensions', 'array of TX509Extension;
19854:   //TypeS('PX509Extensions', '^TX509Extensions // will not work;
19855:  Proc AppendX509Extensions( var A : TX509Extensions; const B : TX509Extension);
19856:  Func EncodeX509Extensions( const A : TX509Extensions) : Ansistr;
19857:  Func DecodeX509Extensions( const TypeID: Byte; const Buf:Str;const Size: Int;var A: TX509Extensions):Int;
19858:  Func NormaliseX509IntKeyBuf( var KeyBuf : Ansistr) : Int;
19859:   TypeS(TX509TBSCertificate', 'record Version : TX509Version; SerialNumb'
19860:    +er : Ansistr; Signature : TX509AlgorithmIdentifier; Issuer : TX509Name;'
19861:    + Validity : TX509Validity; Subject : TX509Name; SubjectPublicKeyInfo : TX5'
19862:    +09SubjectPublicKeyInfo; IssuerUniqueID : Ansistr; SubjectUniqueID : Ans'
19863:    +iString; Extensions: TX509Extensions;_DecodedVersion :Bool; _Decoded:Bool; end;
19864:  // TypeS('PX509TBSCertificate', '^TX509TBSCertificate // will not work;
19865:  Func EncodeX509TBSCertificate( const A : TX509TBSCertificate) : Ansistr;
19866:  Func DecodeX509TBSCertificate(const TypeID:Byte;const Buf:str;const Size:Int;var
       A:TX509TBSCertificate):Int;
19867:  TypeS('TX509Certificate', record TBSCertificate : TX509TBSCertificate;'
19868:    + SignatureAlgorithm: TX509AlgorithmIdentifier;SignatureValue:Ansistr; _Decoded:Bool;end;
19869:  // TypeS('PX509Certificate', '^TX509Certificate // will not work;
19870:  TypeS('TX509CertificateArray', 'array of TX509Certificate;
19871:  Proc InitX509Certificate( var A : TX509Certificate);
19872:  Func EncodeX509Certificate( const A : TX509Certificate) : Ansistr;
19873:  Func DecodeX509Certificate(const TypeID:Byte; const Buf:Str;const Size: Int; var A:TX509Certificate):Int;
19874:  Func ParseX509Certificate(const Buf:str;const Size:Int;var A:TX509Certificate):Int;
19875:  Proc ParseX509CertificateStr( const BufStr : Ansistr; var A : TX509Certificate);
19876:  Proc ParseX509CertificatePEM( const BufStr : Ansistr; var A : TX509Certificate);
19877:   TypeS('TX509RSAPrivateKey', record Version : Int; Modulus : AnsiSt'
19878:    +ring; PublicExponent : Ansistr; PrivateExponent : Ansistr; Prime1 : '
19879:    +Ansistr; Prime2 Ansistr;Exponent1:Ansistr;Exponent2:Ansistr;CRTCoefficient:Ansistr; _Decoded:Bool; end;
19880:  // TypeS('PX509RSAPrivateKey', '^TX509RSAPrivateKey // will not work;
19881:  Func EncodeX509RSAPrivateKey( const A : TX509RSAPrivateKey) : Ansistr;
19882:  Func DecodeX509RSAPrivateKey(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RSAPrivateKey):Int;
19883:  Func ParseX509RSAPrivateKey(const Buf:str;const Size:Int;var A:TX509RSAPrivateKey):Int;
19884:  Proc ParseX509RSAPrivateKeyStr(const BufStr: Ansistr;var A: TX509RSAPrivateKey);
19885:  Proc ParseX509RSAPrivateKeyPEM(const BufStr: Ansistr;var A: TX509RSAPrivateKey);
19886:  Proc SelfTestX509;
19887: end;
19888:
19889: Proc SIRegister_FileStreamW(CL: TPSPascalCompiler);
19890: begin
19891:   SIRegister_TFileStreamW(CL);
19892:   {TFormFadeSettings = record
19893:     Form: TForm;
19894:     Step: ShortInt;
19895:     DisableBlendOnFinish:Bool;
19896:     Callback: TNotifyEvent;
19897:     MinAlpha, MaxAlpha: Byte;
19898:   end;}
19899:  TFormFadeSettings','record Form:TForm;Step:ShortInt;DisableBlendOnFinish:Bool;Callback:TNotifyEvent;
       MinAlpha,MaxAlpha:Byte;end;
19900:  ConstantN('fmForcePath','LongWord').SetUInt( $80000000);
19901:  Func LoadUnicodeFromStream( S : TStream; AsIsAnsi :Bool) : WideString;
19902:  Func GetClipboardText: WideString;;
19903:  Proc CopyToClipboard(Str: WideString);;
19904:  Func CurrentWinUser: WideString;;
19905:  Func GetTempPathW: WideString;;
19906:  Func GetTempFileNameW: WideString;;
19907:  Func GetDesktopFolderW: WideString;;
19908:  Func IsWritable(const FileName: WideString):Bool;;
19909:  Func SysErrorMessageW(ErrorCode: Int): WideString;;
```

```
19910:  Func FormatExceptionInfo: WideString;;
19911:  Proc ShowExceptionW(Message: WideString);;
19912:  Proc ChangeWindowStyle(const Form: HWND; Style: DWord; AddIt:Bool);;
19913:  Func SetNtfsCompressionW(const FileName: WideString; Level: Word):Bool;;
19914:  Func WriteWS(const Stream: TStream; const Str: WideString): Word;;
19915:  Proc FormFadeIn(Form: TForm; Step: ShortInt);;
19916:  Proc FormFadeOut(Form: TForm; Step: ShortInt);;
19917:  Proc FormFadeOutAndWait(Form: TForm; Step: ShortInt);;
19918:  Func BrowseForFolderw(const Caption,DefaultPath:WideStr;const OwnerWindow:HWND):WideStr;
19919:  Proc FormFade(const Settings: TFormFadeSettings);;
19920:  Func HashOfString(const Str: WideString): DWord;
19921:  Func ComparePoints(const First, Second: TPoint): ShortInt;
19922: end;
19923:
19924: Proc SIRegister_InetUtils(CL: TPSPascalCompiler);
19925: begin
19926:  TypeS(TNetUtilsSettings', 'record UserAgent :Str; ProxyURL : Strin'
19927:   +g; OpenURLFlags: DWord; TrafficCounter:Dword;UploadedCounter:DWord;ReadBufferSize:DWord;end;
19928:   TRawCharset', 'set of Char;
19929:   TInetHeaders', 'array of String;
19930:   SIRegister_EInet(CL);
19931:  TInetDownloadCallback', 'Func ( Downloaded, TotalSize : DWord) :Bool;
19932: Func InetDownloadTo(const DestFile:WideString;const URL:str;Callback:TInetDownloadCallback):Bool;
19933:  Func InetDownloadTo1(const DestFile:WideString;const URL:Str;const Settings:
       TNetUtilsSettings;Callback:TInetDownloadCallback):Bool;;
19934:  Func InetDownload(const URL:str;Dest:TStream;Callback:TInetDownloadCallback):Bool;
19935:  Func InetDownload3(const URL:Str; Dest:TStream; const Settings:TNetUtilsSettings;Callback :
       TInetDownloadCallback) :Bool;;
19936:  Func InetBufferedReadFrom(Handle:HInternet;Dest:TStream;const
       Settings:TNetUtilsSettings;Callback:TInetDownloadCallback):DWord;;
19937:  Func InetBufferedReadFrom5( Handle : HInternet; const Settings : TNetUtilsSettings; Callback :
       TInetDownloadCallback) :Str;;
19938:  Func IsResponseStatusOK( Handle : HInternet) :Bool;
19939:  TMultipartItem', 'record Headers : TInetHeaders; Data : TStream; end;
19940:  { TMaskMatchInfo = record
19941:    Matched:Bool;
19942:    StrPos: Word;
19943:    MatchLength: Word;
19944:  end;}
19945:  TMaskMatchInfo', 'record Matched:Bool; StrPos: Word; MatchLength: Word; end;
19946:  TMultipartItems', 'array of TMultipartItem;
19947:   TUploadFile','record Name:str; SourceFileName : WideString; Data : TStream; end;
19948:   TUploadFiles', 'array of TUploadFile;
19949:  Func FindBoundaryFor( const Items : TMultipartItems) :Str;
19950:  Func RandomBoundary :Str;
19951: Func GenerateMultipartFormFrom(const Items:TMultipartItems;out ExtraHeadrs:TInetHeaders):Str
19952:  Func InetUploadTo( const ToURL:Str;const Headers:TInetHeaders;const Items: TMultipartItems; const
       Settings:TNetUtilsSettings) :Str;;
19953:  Func InetUploadTo7(const ToURL:Str;const Items:TMultipartItems;const Settings:TNetUtilsSettings):str;
19954:  Func InetUploadTo8(const ToURL:Str; const Items: TMultipartItems) :Str;;
19955:  Func InetUploadStreamsTo(const ToURL:str;const Settings:TNetUtilsSettings;Streams:TUploadFiles):str;
19956:  Func InetUploadStreamsTo10(const ToURL:str;const Streams:TUploadFiles):str;
19957:  Func InetUploadFileTo(const ToURL:str;const Settings:TNetUtilsSettings;const ItemName:str;const
       FilePath:WideString):str;
19958:  Func InetUploadFileTo12(const ToURL:Str;const ItemName:Str;const FilePath:WideStr):str;
19959:  Func InetUploadFilesTo(const ToURL:str;const Settings:TNetUtilsSettings;const Files:array of const):str;
19960:  Func InetUploadFilesTo14(const ToURL:Str;const Files: array of const):str;
19961:  Func AppendQueryTo(const URL:Str; const Arguments: array of const):str;
19962:  Func HasQueryPart( const URL :Str) :Bool;
19963:  Func BuildQueryFrom( const Arguments : array of const) :Str;
19964:  Func BuildURLW(Protocol,Host:Str;Port:Word;Path,Script:Str;const Arguments:array of const):Str
19965:  Func CustomEncode(const Str: WideString; const RawChars:TRawCharset):Str;
19966:  Func EncodeURI( const Str : WideString) :Str;
19967:  Func EncodeURIComponent( const Str : WideString) :Str;
19968:  Proc InetGetLastError( out ErrorCode : DWord; out ErrorMessage :Str);
19969:  Func InetGetLastErrorCode : DWord;
19970:  Func InetGetLastErrorMsg :Str;
19971:  Func AbsoluteURLFrom( URL, BaseURL, BasePath :Str) :Str;
19972:  Proc SplitURL( const URL :Str; out Domain, Path :Str);
19973:  Func DomainOf( const URL :Str) :Str;
19974:  Func PathFromURL( const URL :Str) :Str;
19975:  Func InetHeaders( const NameValues : array of const) : TInetHeaders;
19976:  Func NoInetHeaders : TInetHeaders;
19977:  Func JoinHeaders( const Headers : TInetHeaders) :Str;
19978:  ConstantN('InetHeaderEOLN','String').SetString( #13#10);
19979:  Proc SetDefaultNetUtilsSettings;
19980:  Func TotalDownTrafficThroughNetUtils : DWord;
19981:   TDBDraw', 'record DisplayDC : HDC; MemDC : HDC; MemBitmap : HBIT'
19982:    +'MAP; OldBitmap : HBITMAP; OldFont : HFONT; OldPen : HPEN; end;
19983:   TPieceFormatData', 'record Position : TMaskMatchInfo; Color : TColor; end;
19984:   TFormatData', 'array of TPieceFormatData;
19985:   TDrawFormattedTextSettings', 'record Text : WideString; FormatDa'
19986:    +'ta : TFormatData; Canvas : TCanvas; WrapText :Bool; DestPos : TPoint; '
19987:    +'MaxWidth : Word; CharSpacing : Word; end;
19988:   TWrapTextSettings', 'record DC : HDC; Str : WideString; Delimite'
19989:    +r :WideString; MaxWidth:Word; LeftMargin:Word; CharSpacing:Word;LastChar: TSize; end;
19990:  Func TextSize( const DC : HDC; const Str : WideString) : TSize;
19991:  Func TextWidthW2( const DC : HDC; const Str : WideString) : Int;
19992:  Func TextHeightW2( const DC : HDC; const Str : WideString) : Int;
```

```
19993:  Func GetLineHeightOf( const Font : HFONT) : Word;
19994:  Func TextWidthEx(const DC:HDC;const Str:WideString; const CharSpacing:Word): Int;
19995:  Func TextHeightEx(const DC:HDC;const Str:WideString; const CharSpacing:Word): Int;
19996:  Func TextSizeEx(const DC:HDC;const Str:WideString;const CharSpacing:Word):TSize;
19997:  Func TextWithBreaksSize( Settings : TWrapTextSettings) : TSize;;
19998:  Func DoubleBufferedDraw(const DisplaySurface:HDC;const BufferSize:TPoint):TDBDraw;
19999:  Func DoubleBufferedDraw17(const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
20000:  Func DoubleBufferedDraw2(const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
20001:  Proc DrawFormattedText( const Settings : TDrawFormattedTextSettings);
20002: Func GetLastCharPos(const DC:HDC;const Str:WideString;const MaxWidth:Word;const ChrSpacing:Word):TSize;
20003:  Func WrapNonMonospacedText(const DC:HDC;const Str:WideString;const Delimiter:WideString; const
       MaxWidth:Word;const CharSpacing:Word):WideString;;
20004:  Func WrapNonMonospacedText2( var Settings : TWrapTextSettings) : WideString;;
20005: end;
20006:
20007: Proc SIRegister_StrConv(CL: TPSPascalCompiler);
20008: begin
20009:  TypeS('TCodepage', 'DWord;
20010:  ConstantN('CP_INVALID','LongInt').SetInt( TCodepage ( - 1 ));
20011:  'CP_ASIS','LongInt').SetInt( TCodepage ( - 2 ));
20012:  'CP_ANSI','longint').SetInt(0);
20013:  'CP_OEM','longint').SetInt(1);
20014:  'CP_SHIFTJIS','LongInt').SetInt( 932);
20015:  'CP_LATIN1','LongInt').SetInt( 1250);
20016:  'CP_UNICODE','LongInt').SetInt( 1200);
20017:  'CP_UTF8','LongInt').SetInt( 65001);
20018:  Func MinStrConvBufSize( SrcCodepage : TCodepage; Str :Str) : Int;;
20019:  Func MinStrConvBufSize1( DestCodepage : TCodepage; Wide : WideString) : Int;;
20020:  Func ToWideString(SrcCodepage: TCodepage; Str :Str; BufSize: Int) : WideString;
20021:  Func FromWideString(DestCodepage:TCodepage;Str:WideString;BufSize:Int;Fail:Boolean):str;
20022:  Func CharsetToID( Str :Str) : TCodepage;
20023:  Func IdToCharset( ID : TCodepage; GetDescription :Bool) :Str;
20024:  Func CompareStrW(const S1, S2: WideString; Flags: DWord = 0): Int;
20025:  Func CompareTextW(const S1, S2: WideString): Int;
20026:  Func MaskMatch(const Str, Mask: WideString):Bool;
20027:  { Info can have special values in some cases:
20028:   * Matched = True but MatchLength = 0(and StrPos having random value) - means that Mask consisted of only
       "*" and
20029:     thus no particular substring could specified (since it could match anypart ofthe string). }
20030:  Func MaskMatchInfo(const Str,Mask:WideString;StartingPos:Word = 1):TMaskMatchInfo;
20031:  //Strutils ------------------------
20032:   Func TryStrToIntStrict(const S:Str; out Value:Int; Min:Int):Boolean;
20033:  Func TryStrToFloatStrict(const S:str; out Value:Single;const FormatSettings:TFormatSettings):Bool;
20034:  Func TryStrToFloatStrict1(const S:str;out Value:Double;const FormatSettings: TFormatSettings):Bool;
20035:  Func DetectEolnStyleIn( const Str : WideString) : WideString;
20036:  Func DetectEolnStyleInANSI( Stream : TStream) : WideString;
20037:  Func PascalQuote( const Str : WideString) : WideString;
20038:  Func StrRepeatW( const Str : WideString; Times : Int) : WideString;
20039:  Func EscapeString(const Str: WideString; CharsToEscape:WideString): WideString;
20040:  Func UnescapeString(const Str: WideString; CharsToEscape: WideString):WideString;
20041:  Func BinToHexW( const Buf :Str; Delim :Str) :Str;
20042:  Func HexToBinW( Text :Str) :Str;
20043:  Func SoftHexToBin( Text :Str) :Str;
20044:  Func FormatVersion( Version : Word) : WideString;
20045:  Func FormatDateW( Date : DWord) : WideString;
20046:  Func FormatNumber( Number : DWord) : WideString;
20047:  Func GenericFormat(Number:Single;const Language:TGenericFormatLanguage):WideString;
20048:  Func FormatInterval( Millisecs : DWord) : WideString;
20049:  Func FormatSize( Bytes : DWord) : WideString;
20050:  Func PosLast( const Substr, Str :Str; Start : Word) : Int;
20051:  Func PosLastW( const Substr, Str : WideString; Start : Word) : Int;
20052:  Func IsDelimiterW(const Delimiters,S: WideString; Index: Int):Bool;
20053:  Func RemoveNonWordChars(const Str: WideString;DoNotRemove:WideString):WideString;
20054:  Func IsQuoteChar( const aChr : Char) :Bool;
20055:  Func WrapTextW(const Str:WideStr;const Delimiter:WideStr;const MaxWidth:Word):WideStr;
20056:  Func PadText(const Str:WideStr;const NewLine,PadStr:WideStr;const MaxWidth:Word):WideStr;
20057:  Func PadTextWithVariableLineLength(const Str:WideString;const NewLine,PadStr:WideString;const
       LineLengths:array of Int):WideString;
20058:  Func StrPadW(const Str:WideString; ToLength:Int; PadChar: WideChar) : WideString;
20059:  Func StrPadLeftW(const Str:WideString;ToLength:Int;PadChar: WideChar): WideString;
20060:  //Func StrRepeat( const Str : WideString; Times : Int) : WideString;
20061:  Func StrReverseW( const Str : WideString) : WideString;
20062:  Func CountSubstr( const Substr, Str : WideString) : Int;
20063:  Proc DeleteArrayItem( var A : TWideStringArray; Index : Int);
20064:  Func TrimStringArray( WSArray : TWideStringArray) : TWideStringArray;
20065:  Func TrimWS( Str : WideString; const Chars : WideString) : WideString;
20066:  Func TrimLeftWS( Str : WideString; const Chars : WideString) : WideString;
20067:  //Func TrimRightWS( Str : WideString; const Chars : WideString) : WideString;
20068:  Func ConsistsOfChars( const Str, Chars : WideString) :Bool;
20069:  Func UpperCaseW( const Str : WideString) : WideString;
20070:  Func LowerCaseW( const Str : WideString) : WideString;
20071:  Func UpperCaseFirst( const Str : WideString) : WideString;
20072:  Func LowerCaseFirst( const Str : WideString) : WideString;
20073:  Func StripAccelChars( const Str : WideString) : WideString;
20074: end;
20075:
20076: Proc SIRegister_REXX(CL: TPSPascalCompiler);
20077: begin
20078:   TypeS(TState', '( TrimLeader, StartToken, EndToken );
```

```
20079:   TStrIndex', 'LongInt;
20080:   TTokIndex', 'WORD;
20081:   TStrIndexB', 'BYTE;
20082:   TTokIndexB', 'BYTE;
20083:   ClassN(CL.FindClass('TOBJECT'),'EConversionError;
20084:  Func Abbrev(const information, info:Str; const nMatch: TStrIndex):BOOLEAN;
20085:  Func AllSame( const s :Str; const c : CHAR) :Bool;
20086:  Func Capitalize( const s :Str) :Str;
20087:  Func Center( const s :Str; const sLength : TStrIndex) :Str;
20088:  Func Left( const s :Str; const sLength : TStrIndex) :Str;
20089:  Func Right( const s :Str; const sLength : TStrIndex) :Str;
20090:  Func Copies( const s :Str; const n : TStrIndex) :Str;
20091:  Func CountChar( const s :Str; const c : CHAR) : TStrIndex;
20092:  Func DeleteStringrexx( const substring :Str; const s :Str) :Str;
20093:  Func Overlay( const ovly, target :Str; const n : TStrIndex) :Str;
20094:  Func Plural(const n: LongInt; const singularform, pluralform :Str) :Str;
20095:  Func Reverse( const s :Str) :Str;
20096:  Func Spacerexx( const s :Str; const n : TStrIndex) :Str;
20097:  Func Striprexx( const s :Str; const option :Str) :Str;
20098:  Func TestString( const sLength : TStrIndex) :Str;
20099:  Func Translate( const s, OutTable, InTable :Str) :Str;
20100:  Func XRange( const start, stop : BYTE) :Str;
20101:  Func B2X(const b: BYTE) :Str;
20102:  Func C2D(const s:Str) : DOUBLE;
20103:  Func C2I(const s:Str) : Int;
20104:  Func C2L(const s:Str) : LONGINT;
20105:  Func C2W(const s:Str) : WORD;
20106:  Func C2X(const s:Str) :Str;
20107:  Func I2C(const i: Int) :Str;
20108:  Func I2X(const i: Int) :Str;
20109:  Func L2C(const i: LONGINT) :Str;
20110:  Func L2X(const i: LONGINT) :Str;
20111:  Func D2C(const x: DOUBLE; const d : BYTE):Str;
20112:  Func W2C(const w: WORD) :Str;
20113:  Func W2X(const w: WORD) :Str;
20114:  Func X2W(const s:Str) : WORD;
20115:  Func JulianDate( const DateTime : TDateTime) : LongInt;
20116:  Func TimeDifference( const StartTime, StopTime : TDateTime) : DOUBLE;
20117:  Func Pwr( const x, y : DOUBLE) : DOUBLE;
20118: end;
20119:
20120: Proc SIRegister_StringGridLibrary(CL: TPSPascalCompiler);
20121: begin
20122:  Proc ReadGridFile( var StringGrid : TStringGrid; GridFile :Str);
20123:  Proc WriteGridFile( var StringGrid : TStringGrid; GridFile :Str);
20124:  Proc AddBlankRowToTop( var StringGrid : TStringGrid);
20125:  Proc DeleteSelectedRow( var StringGrid : TStringGrid);
20126:  Func StringGridSearch(const StringGrid:TStringGrid;const column:Int;const target:STR):Int;
20127:  Func XLeft( rect : TRect; canvas : TCanvas; s :Str) : Int;
20128:  Func XCenter( rect : TRect; canvas : TCanvas; s :Str) : Int;
20129:  Func XRight( rect : TRect; canvas : TCanvas; s :Str) : Int;
20130:  Func YCenter( rect : TRect; canvas : TCanvas; s :Str) : Int;
20131: end;
20132:
20133: Proc SIRegister_InetUtils2(CL: TPSPascalCompiler);
20134: begin
20135:  Proc AdjustArray(var DWArray:array of DWord;const Delta:Int;MinValueToAdjust:DWord);
20136:  Func GetDroppedFileNames( const DropID : Int) : TWideStringArray;
20137:  Func AreBytesEqual( const First, Second : array of Byte) :Bool;;
20138:  Func AreBytesEqual1( const First, Second, Length : DWord) :Bool;;
20139:  Func MaskForBytes( const NumberOfBytes : Byte) : DWord;
20140:  Func IntToBinByte( Int : Byte) :Str;;
20141:  Func IntToBinWord( Int : Word) :Str;;
20142:  Func IntToBinDWord( Int : DWord; Digits : Byte; SpaceEach : Byte) :Str;;
20143:  Func WriteWS( const Stream : TStream; const Str : WideString) : Word;
20144:  Proc WriteArray( const Stream : TStream; const WSArray : array of WideString);;
20145:  Proc WriteArray6( const Stream : TStream; const DWArray : array of DWord);;
20146:  Func ReadWS( const Stream : TStream) : WideString;;
20147:  Func ReadWS8( const Stream : TStream; out Len : Word) : WideString;;
20148:  Proc ReadArray( const Stream : TStream; var WSArray : array of WideString);;
20149:  Proc ReadArray10( const Stream : TStream; var DWArray : array of DWord);;
20150:  Func ParamStrW( Index : Int) : WideString;
20151:  Func ParamStrFrom( CmdLine : WideString; Index : Int) : WideString;
20152:  Func ParamStrEx( CmdLine : WideString; Index : Int; out Pos : Int) : WideString;
20153:   Proc FindMask( Mask : WideString; Result : TStringsW);
20154:  Proc FindAll( BasePath, Mask : WideString; Result : TStringsW);
20155:  Proc FindAllRelative( BasePath, Mask : WideString; Result : TStringsW);
20156:  Func IsInvalidPathChar( const Char : WideChar) :Bool;
20157:  Func MakeValidFileNameW(const Str:WideString;const SubstitutionChar:WideChar):WideString;
20158:   Func ExtractFilePathW( FileName : WideString) : WideString;
20159:  Func ExtractFileNameW( Path : WideString) : WideString;
20160:  Func ExpandFileNameW( FileName : WideString) : WideString;;
20161:  Func ExpandFileName12( FileName, BasePath : WideString) : WideString;;
20162:  Func CurrentDirectory : WideString;
20163:  Func ChDirW( const ToPath : WideString) :Bool;
20164:  Func ExtractFileExtW( FileName : WideString) : WideString;
20165:  Func ChangeFileExtW2( FileName, Extension : WideString) : WideString;
20166:  Func IncludeTrailingBackslashW( Path : WideString) : WideString;
20167:  Func ExcludeTrailingBackslashW( Path : WideString) : WideString;
```

```
20168:  Func IncludeTrailingPathDelimiterW( Path : WideString) : WideString;
20169:  Func ExcludeTrailingPathDelimiterW( Path : WideString) : WideString;
20170:  Func IncludeLeadingPathDelimiter( Path : WideString) : WideString;
20171:  Func ExcludeLeadingPathDelimiter( Path : WideString) : WideString;
20172:  Func FileInfo( Path : WideString) : TWin32FindData;
20173:  Func IsDirectoryW( Path : WideString) :Bool;
20174:  Func FileAgeW( const FileName : WideString) : Int;
20175:  Func FileExistsW( Path : WideString) :Bool;
20176:  Func FileSizeW( Path : WideString) : DWord;
20177:  Func FileSize64( Path : WideString) : Int64;
20178:  Func DeleteFileW( Path : WideString) :Bool;
20179:  Func CopyDirectoryW( Source, Destination : WideString) :Bool;
20180:  Func RemoveDirectoryW( Path : WideString) :Bool;
20181:  Func ForceDirectoriesW( Path : WideString) :Bool;
20182:  Func MkDirW( Path : WideString) :Bool;
20183:  Func GetEnvironmentVariableW( Name : WideString) : WideString;
20184:  Func ResolveEnvVars(Path:WideString; Callback:TEnvVarResolver;Unescape:Boolean):WideString;
20185:  Func ResolveEnvVars14( Path : WideString; Unescape :Bool) : WideString;;
20186:  Func ReadRegValue( Root : DWord; const Path, Key : WideString) : WideString;
20187:  Func FadeSettings(Form:TForm;Step:ShortInt;Callback:TNotifyEvent):TFormFadeSettings;
20188:  Func FadeSettings17(Form:TForm;MinAlpha,MaxAlpha:Byte;Step:ShortInt):TFormFadeSettings
20189:  Func GetSpecialFolderID( const Path :Str) : Int;
20190:  Func GetSpecialFolderPath2( FolderID : Int) :Str;
20191: end;
20192:
20193: Proc SIRegister_KFunctions(CL: TPSPascalCompiler);
20194: begin
20195:  SHFolderDll','String').SetString( 'SHFolder.dll;
20196:  ConstantN('KM_BASE','LongInt').SetInt( LM_USER + 1024);
20197:  ConstantN('KM_LATEUPDATE','LongInt').SetInt( KM_BASE + 1);
20198:  crHResize','LongInt).SetInt( TCursor ( 101 ));
20199:  ConstantN('crVResize','LongInt').SetInt( TCursor ( 102 ));
20200:  crDragHandFree','LongInt).SetInt( TCursor ( 103 ));
20201:  crDragHandGrip','LongInt).SetInt( TCursor ( 104 ));
20202:  cCheckBoxFrameSize','LongInt).SetInt( 13);
20203:  cCR','Char).SetString( #13);
20204:  cLF','Char).SetString( #10);
20205:  cTAB','Char).SetString( #9);
20206:  cSPACE','Char).SetString( #32);
20207:  cNULL','Char).SetString( #0);
20208:  {cWordBreaks','LongInt').Value.ts32 := ord(cNULL) or ord(cTAB) or ord(cSPACE);
20209:  cLineBreaks','LongInt').Value.ts32 := ord(cCR) or ord(cLF);
20210:  cEllipsis','String').SetString( '...;
20211:  cEOL','').SetString( cLF);
20212:  cFirstEOL','').SetString( cLF);
20213:  cFirstEOL','').SetString( cCR); }
20214:   TypeS('TLMessage', 'TMessage;
20215:   TypeS('TKkString', 'Widestring;
20216:   TypeS('TKkChar', 'char;
20217:   //TKChar = TUTF8Char;
20218:   TypeS(LONG_PTR2', 'Longint;
20219:   TypeS(TKSysCharSet', 'set of AnsiChar;
20220:   TypeS(TKCurrencyFormat', 'record CurrencyFormat : Byte; CurrencyDecima'
20221:    +ls : Byte; CurrencyString : TKkString; DecimalSep : Char; ThousandSep : Cha'
20222:    +r; UseThousandSep :Bool; end;
20223:  {TypeS('TKAppContext', 'record Application : TApplication; Screen : TScr'
20224:    +'een; GlobalNameSpace : IReadWriteSync; MainThreadID : LongWord; IntConstLi'
20225:    +'st : TThreadList; WidgetSet : TWidgetSet; DragManager : TDragManager; end;  }
20226:  // TypeS('PKAppContext', '^TKAppContext // will not work;
20227:  //TypeS('TKClipboardFormat', 'TClipboardFormat;
20228:   TypeS('TKClipboardFormat', 'Word;
20229:   TypeS('TKCellSpan', 'record ColSpan : Int; RowSpan : Int; end;
20230:  DelphiFunction('Func AdjustDecimalSeparator( const S :Str) :Str;
20231:  Func AnsistrToString( const Text : Ansistr; CodePage :Card) : TKkString;
20232:  //Func BinarySearch(AData:str; ACount:Int;KeyPtr:str;ACompareProc:TBSCompareProc;ASortedDown: Bool):Int;
20233:  Proc CallTrackMouseEvent( Control : TWinControl; var Status :Bool);
20234:  Proc CenterWindowInWindow( CenteredWnd, BoundWnd : HWnd);
20235:  Proc CenterWindowOnScreen( CenteredWnd : HWnd);
20236:  Func CharInSetEx( AChar : AnsiChar; const ASet : TKSysCharSet) :Bool;;
20237:  Func CharInSetEx1( AChar : WideChar; const ASet : TKSysCharSet) :Bool;;
20238:  Func ClipboardLoadStreamAs(const AFormat:str;AStream:TStream;var AText:TKkString):Bool;
20239:  Func ClipboardSaveStreamAs(const AFormat:str;AStream:TStream;const AText:TKkString):Bool;
20240:  Func CompareInts(I1,I2: Int):Int;//Func CompareWideChars(W1,W2:PWideChar;Locale:Card):Int;
20241: // Func CompareChars( S1, S2 : PChar; Locale :Card) : Int;
20242:  Func CompareWideStrings( W1, W2 : WideString; Locale :Card) : Int;
20243: // Func CompareStrings( S1, S2 :Str; Locale :Card) : Int;
20244:  Proc ConvertTabsToSpaces( var AText : TKkString; ASpacesForTab : Int);
20245:  Func CreateMultipleDir( const Dir :Str) :Bool;
20246:  Func DigitToNibble( Digit : AnsiChar; var Nibble : Byte) :Bool;
20247:  Func DivUp( Dividend, Divisor : Int) : Int;
20248:  Func DivDown( Dividend, Divisor : Int) : Int;
20249:  Func EditIsFocused( AMustAllowWrite :Bool) :Bool;
20250:  Func EditFocusedTextCanCopy :Bool;
20251:  Func EditFocusedTextCanCut :Bool;
20252:  Func EditFocusedTextCanDelete :Bool;
20253:  Func EditFocusedTextCanPaste :Bool;
20254:  Func EditFocusedTextCanUndo :Bool;
20255:  Proc EditUndoFocused;
20256:  Proc EditDeleteFocused;
```

```
20257:  Proc EditCutFocused;
20258:  Proc EditCopyFocused;
20259:  Proc EditPasteFocused;
20260:  Proc EditSelectAllFocused;
20261:  Proc EnableControls2( AParent : TWinControl; AEnabled, ARecursive :Bool);
20262:  Proc EnsureLastPathSlash( var APath :Str);
20263:  Proc Error2( const Msg :Str);
20264:  Func FillMessage( Msg :Card; WParam : WPARAM; LParam : LPARAM) : TLMessage;
20265:  Func FormatCurrency( Value : Currency; const AFormat : TKCurrencyFormat) : TKkString;
20266:  //Func GetAppContext( var Ctx : TKAppContext) :Bool;
20267:  Func GetAppVersion(const ALibName:str;var MajorVersion,MinorVersion,BuildNumber,RevisionNumber:Word):Bool;
20268:  Func GetCharCount( const AText : TKkString; AChar : TKkChar) : Int;
20269:  Func GetControlText( Value : TWinControl) : TKkString;
20270:  Func GetFormatSettings2 : TFormatSettings;
20271:  Func GetShiftState : TShiftState;
20272:  Func IntToAscii( Value : Int64; Digits : Int) :Str;
20273:  Func IntToBinStr( Value : Int64; Digits : Byte; const Suffix :Str) :Str;
20274:  Func IntToBCD( Value :Card) :Card;
20275:  Func IntToDecStr( Value : Int64) :Str;
20276:  Func IntToHexStr(Value:Int64;Digits:Byte;const Prefix,Suffix:str;UseLowerCase:Bool):str;
20277:  Func IntToOctStr( Value : Int64) :Str;
20278:  Func IntToRoman2( Value : Int; AUpperCase :Bool) :Str;
20279:  Func IntToLatin( Value : Int; AUpperCase :Bool) :Str;
20280:  Func IntPowerInt( Value : Int64; Exponent : Int) : Int64;
20281:  Func AsciiToInt( S :Str; Digits : Int) : Int64;
20282:  Func BCDToInt( Value :Card) :Card;
20283:  Func BinStrToInt2(S:str; Digits:Byte; Signed :Bool; var Code : Int) : Int64;
20284:  Func DecStrToInt( S :Str; var Code : Int) : Int64;
20285:  Func HexStrToInt(S:Str; Digits: Byte; Signed:Boolean; var Code : Int) : Int64;
20286:  Func OctStrToInt( S :Str; var Code : Int) : Int64;
20287:  Func KFormat14(const Format:str;const Args:array of const;const AFormatSettings:TFormatSettings):str;
20288:  Func KFormat15(const Format:WideString;const Args:array of const;const AFormatSettings:TFormatSettings):
        WideString;
20289:  Func MakeCellSpan( AColumns, ARows : Int) : TKCellSpan;
20290:  Func MinMax16(Value,Min,Max: ShortInt) : ShortInt;;
20291:  Func MinMax17(Value,Min,Max: SmallInt) : SmallInt;;
20292:  Func MinMax18(Value,Min,Max: Int) : Int;;
20293:  Func MinMax19(Value,Min,Max: Int64) : Int64;;
20294:  Func MinMax20(Value,Min,Max: Single) : Single;;
20295:  Func MinMax21(Value,Min,Max: Double) : Double;;
20296:  Func MinMax22(Value,Min,Max: Extended) : Extended;;
20297:  Func NibbleToDigit( Nibble : Byte; UpperCase :Bool) : AnsiChar;
20298:  Proc OpenURLWithShell( const AText : TKkString);
20299:  //Proc QuickSortNR(AData: Pointer;ACount:Int; ACompareProc:TQsCompareProc;AExchangeProc:TQsExchangeProc;
        ASortedDown:Bool);
20300:  //Proc QuickSort(AData:Pointer;ACount:Int;ACompareProc:TQsCompareProc;AExchangeProc:TQsExchangeProc;
        ASortedDown:Bool);
20301:  Proc OffsetPoint23( var APoint : TPoint; AX, AY : Int);;
20302:  Proc OffsetPoint24( var APoint : TPoint; const AOffset : TPoint);;
20303:  Func RectInRect( Bounds, Rect : TRect) :Bool;
20304:  Proc OffsetRect25( var ARect : TRect; AX, AY : Int);;
20305:  Proc OffsetRect26( var ARect : TRect; const AOffset : TPoint);;
20306:  //Func SetAppContext( const Ctx : TKAppContext) :Bool;
20307:  Proc SetControlClipRect( AControl : TWinControl; const ARect : TRect);
20308:  Proc SetControlText( Value : TWinControl; const Text : TKkString);
20309:  Proc StripLastPathSlash( var APath :Str);
20310:  Func StrNextCharIndex( const AText : TKkString; Index : Int) : Int;
20311:  Func StrPreviousCharIndex( const AText : TKkString; Index : Int) : Int;
20312:  Func StringCharBegin( const AText : TKkString; Index : Int) : Int;
20313:  Func StringLength( const AText : TKkString) : Int;
20314:  Func StringCopy( const ASource : TKkString; At, Count : Int) : TKkString;
20315:  Proc StringDelete( var ASource : TKkString; At, Count : Int);
20316:  Proc TrimWhiteSpaces27(const AText:TKkString;var AStart,ALen:Int;const ASet:TKSysCharSet);
20317:  Proc TrimWhiteSpaces28( var AText : TKkString; const ASet : TKSysCharSet);;
20318:  Proc TrimWhiteSpaces29( var AText : Ansistr; const ASet : TKSysCharSet);;
20319:  Func StringToAnsistr( const AText : TKkString; CodePage :Card) : Ansistr;
20320:  Func StringToChar( const AText : TKkString; AIndex : Int) : TKkChar;
20321:  Func GetWindowsFolder2( CSIDL :Card; var APath :Str) :Bool;
20322:  Func RunExecutable( const AFileName :Str; AWaitForIt :Bool) : DWORD;
20323:  Func SystemCodePage : Int;
20324:  Func NativeUTFToUnicode( const AText : TKkString) : WideChar;
20325:  Func UnicodeUpperCase2( const AText : TKkString) : TKkString;
20326:  Func UnicodeLowerCase2( const AText : TKkString) : TKkString;
20327:  Func UnicodeToNativeUTF( const AParam : WideChar) : TKkString;
20328: Func UnicodeStringReplace(const AText,AOldPattern,ANewPattern:TKkString;AFlags:TReplaceFlags):TKkString;
20329:  end;
20330:
20331: Proc SIRegister_KMessageBox(CL: TPSPascalCompiler);
20332: begin
20333: TypeS('TKMsgBoxButton',(mbYes,mbNo,mbOK,mbCancel,mbClose,mbAbort,mbRetry,mbIgnorembAllmbNoToAll,mbYesToAll,
       mbHelp);
20334:   TypeS('TKMsgBoxIcon', ( miNone, miInformation, miQuestion, miWarning, miStop );
20335:  Func CreateMsgBox(const Caption,Text:str;const Buttons:array of TKMsgBoxButton;Icon:TKMsgBoxIcon;Def
       Int):TCustomForm;
20336:  Func CreateMsgBoxEx(const Caption,Text:str;const Btns:array of
       string;Icon:TKMsgBoxIcon;Def:Int):TCustomForm;
20337:  Proc FreeMsgBox( AMsgBox : TCustomForm);
20338:  Func KMsgBox(const Caption,Text:str;const Buttons:array of TKMsgBoxButton;Icon:TKMsgBoxIcon;Def:Int):Int;
20339:  Func KMsgBoxEx(const Caption,Text:str;const Buttons:array of string;Icon:TKMsgBoxIcon;Def:Int):Int;
```

```
20340:  Func KInputBox( const Caption, Prompt :Str; var Text :Str) : TModalResult;
20341:  //Func KNumberInputBox(const ACaption,APrompt:str;var AValue:double;AMin,AMax:double;
      AFormats:TKNumberEditAcceptedFormats):TModalResult;
20342:  TKMsgBoxButtons','(mbAbortRetryIgnore,mbOkOnly,mbOkCancel,mbRetryCancel,mbYesNo,mbYesNoCancel);
20343:  Func MsgBox2(const Caption,Txt:str;const Buttons:TKMsgBoxButtons;Icon:TKMsgBoxIcon):Int;
20344:  Func AppMsgBox(const Caption, Text :Str; Flags: Int) : Int;
20345: end;
20346:
20347: Proc SIRegister_Kronos(CL: TPSPascalCompiler);
20348: begin
20349:  'ChurchDayCount','LongInt').SetInt( 21);
20350:  ConstantN('CommonDayCount','LongInt').SetInt( 4);
20351:  'chAdvent1','LongInt').SetInt( 1);
20352:  'chAdvent2','LongInt').SetInt( 2);
20353:  'chAdvent3','LongInt').SetInt( 3);
20354:  'chAdvent4','LongInt').SetInt( 4);
20355:  'chChristmasEve','LongInt').SetInt( 5);
20356:  'chChristmasDay','LongInt').SetInt( 6);
20357:  'chBoxingDay','LongInt').SetInt( 7);
20358:  'chNewYearEve','LongInt').SetInt( 8);
20359:  'chNewYearDay','LongInt').SetInt( 9);
20360:  'chShroveTuesday','LongInt').SetInt( 10);
20361:  'chAshWednesday','LongInt').SetInt( 11);
20362:  'chPalmSunday','LongInt').SetInt( 12);
20363:  'chMaundyThursday','LongInt').SetInt( 13);
20364:  'chGoodFriday','LongInt').SetInt( 14);
20365:  'chEasterEve','LongInt').SetInt( 15);
20366:  'chEasterSunday','LongInt').SetInt( 16);
20367:  'chEasterMonday','LongInt').SetInt( 17);
20368:  'chWhitEve','LongInt').SetInt( 18);
20369:  'chWhitSunday','LongInt').SetInt( 19);
20370:  'chWhitMonday','LongInt').SetInt( 20);
20371:  'chAscensionDay','LongInt').SetInt( 21);
20372:  'coUNDay','LongInt').SetInt( 22);
20373:  'coWomensDay','LongInt').SetInt( 23);
20374:  'coMayDay','LongInt').SetInt( 24);
20375:  'coLiteracyDay','LongInt').SetInt( 25);
20376:  'UserDayType','LongInt').SetInt( 26);
20377:  //UserDayType = ChurchDayCount + CommonDayCount + 1;
20378:    TFirstLastNumber', 'array[1..2] of word;;
20379:    TDaytypeID', 'array[1..255] of word;;
20380:    TMonthImage2', 'array[0..7] of smallint;;
20381:    TMonthImage', 'array[1..6] of tmonthimage2;;
20382:     //TMonthImage = array[1..6, 0..7] of smallint;
20383:   // TFirstLastNumber = array[1..2] of word;
20384:    // TDaytypeID = array[1..255] of word;
20385:   TDay','record Daynum : Word; MonthDate : word; DOWNum : word; M'
20386:    +'onth : word; Week : word; DayCode : Word; end;
20387:   TypeS('TWeek', 'record WeekNum : word; WhichDays : TFirstLastNumber; end;
20388:   TMonth', 'record Month : word; Daycount : Word; WeekCount : Word'
20389:    +'; WhichWeeks : TFirstLastNumber; WhichDays : TFirstLastNumber; end;
20390:   TYear', 'record WeekCount : word; DayCount : Word; end;
20391:   TKron', 'record ActiveYear : Word; IsInitialized :Bool; end;
20392:   TYearExt', 'record Year : word; NumDays : word; NumWeeks : word;'
20393:    + LeapYear :Bool; YeartypeCount : word; end;
20394:   TDateExt', 'record Year : word; DayOfWeekNumber : word; DayName '
20395:    +:Str; MonthDay : Word; DayNumber : word; DaytypeCount: word; DaytypeI'
20396:    +D : TDaytypeID; MonthNumber : word; WeekNumber : word; Holiday : bool; '
20397:    +ChurchDay : Bool; Flagday :Bool; end;
20398:   TMonthExt', 'record Year : word; MonthNumber : word; MonthName :'
20399:    +' string; FirstDay : word; LastDay : word; NumDays : word; NumWeeks : word;'
20400:    +' FirstWeek : word; LastWeek : word; MonthImage : TMonthImage; end;
20401:   TWeekExt', 'record Year: word; WeekNumber: word; FirstDay : word; LastDay : word; end;
20402:   TForeignKey','record KeyName :Str; KeyValue : Variant; end;
20403:   TDaytypeDef','record AName :Str; ADate : word; ARelDayTyp'
20404:    +e : word; AnOffset : Int; AFirstShowUp : word; ALastShowUp : word; ASh'
20405:    +owUpFrequency : word; AChurchDay :Bool; AHoliday :Bool; AFlagday :'
20406:    + bool; AUserCalc :Bool; ATag : Int; end;
20407:   SIRegister_TDaytype(CL);
20408:   TWeekDay', '( Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday );
20409:   TWeekHolidays', 'set of TWeekDay;
20410:   TOcEVent', '( ocYear, ocMonth, ocMonthnumber, ocWeek, ocWeeknumb'
20411:    +'er, ocMonthDay, ocWeekday, ocDate, ocToday, ocCalcDaytype );
20412:   TCalcDaytypeEvent', 'Proc ( Sender : TObject; Daytype : TDa'
20413:    +'ytype; ADateExt : TDateExt; IsCurrentDate :Bool; var Accept :Bool);
20414:   TLoadDaytypeEvent', 'Proc ( Sender : TObject; const Daytype'
20415:    +'Def : TDaytypeDef; const DescKeys :Str; ClassId : Int; var LoadIt: bool);
20416:   TSaveDaytypeEvent', 'Proc ( Sender : TObject; Daytype : TDa'
20417:    +'ytype; var DescKeys :Str; var ClassID : Int; var SaveIt :Bool);
20418:   SIRegister_TKronos(CL);
20419:   ClassN(CL.FindClass('TOBJECT'),'EKronosError;
20420:  //DelphiFunction('Proc Register;
20421: end;
20422:
20423: Proc SIRegister_MapFiles(CL: TPSPascalCompiler);
20424: begin
20425:   ClassN(CL.FindClass('TOBJECT'),'EMemoryMappedFile;
20426:   ClassN(CL.FindClass('TOBJECT'),'EMMEndOfFile;
20427: TypeS('TFileOptions,(foCreateNew,foCreateAlways,foOpenExisting,foOpenAlways,foTruncateExisting);
```

```
20428:   TypeS('TmapFileType', ( ftUnspecified, ftRandomAccess, ftSequentialAccess);
20429:   SIRegister_TMapStream(CL);
20430:   SIRegister_TTextMap(CL);
20431:   SIRegister_TRecordMap(CL);
20432: end;
20433:
20434: Proc SIRegister_KGraphics(CL: TPSPascalCompiler);
20435: begin
20436:   PNGHeader','String').SetString(' #137 ''PNG'' #13#10#26#10;
20437:   MNGHeader','String').SetString( '#138 ''MNG'' #13#10#26#10;
20438:   TKBrightMode', '( bsAbsolute, bsOfBottom, bsOfTop );
20439:   TKColorRec', 'record R : byte; G : byte; B : byte; A : Byte; Value :Card; end;
20440:   //PKColorRec', '^TKColorRec // will not work;
20441:   TKDynColorRecs', 'array of TKColorRec;
20442:   TKImageHeaderString', 'string;
20443:   //TKPngImage', 'TPortableNetworkGraphic;
20444:  with ClassN(CL.FindClass('TLinarGraphic'),'TPNGGraphic') do
20445:       TKPngImage', 'TPNGGraphic;
20446:       TKKString', 'WideString;
20447:    // TKString = WideString;
20448:     //TKPngImage', 'TPngImage;
20449:   //TKPngImage', 'TPngObject;
20450:   TKTextAttribute', '( taCalcRect, taClip, taEndEllipsis, taFillRe'
20451:    +'ct, taFillText, taIncludePadding, taLineBreak, taPathEllipsis, taWordBreak'
20452:    +', taWrapText, taTrimWhiteSpaces, taStartEllipsis );
20453:   TKTextAttributes', 'set of TKTextAttribute;
20454:   TKHAlign', '( halLeft, halCenter, halRight, halJustify );
20455:   TKStretchMode', '( stmNone, stmZoomOutOnly, stmZoomInOnly, stmZoom );
20456:   TKTextHAlign', 'TKHAlign;
20457:   TKVAlign', '( valTop, valCenter, valBottom );
20458:   TKTextVAlign', 'TKVAlign;
20459:   TKButtonDrawState', '( bsUseThemes, bsDisabled, bsPressed, bsFocused, bsHot );
20460:   TKButtonDrawStates', 'set of TKButtonDrawState;
20461:   SIRegister_TKGraphic(CL);
20462:   SIRegister_TKAlphaBitmap(CL);
20463:   SIRegister_TKMetafile(CL);
20464:   TKTextBoxFunction', '( tbfMeasure, tbfGetIndex, tbfGetRect, tbfDraw );
20465:   SIRegister_TKTextBox(CL);
20466:   SIRegister_TKDragWindow(CL);
20467:   SIRegister_TKHintWindow(CL);
20468:   SIRegister_TKTextHint(CL);
20469:   SIRegister_TKGraphicHint(CL);
20470:  //DelphiFunction('Proc BlendLine( Src, Dest : PKColorRecs; Count : Int);
20471:  Func TKBrightColor( Color : TColor; Percent : Single; Mode : TKBrightMode): TColor;
20472:  Proc CanvasGetScale( ACanvas : TCanvas; out MulX, MulY, DivX, DivY : Int);
20473:  Proc CanvasResetScale( ACanvas : TCanvas);
20474:  Func CanvasScaled( ACanvas : TCanvas) :Bool;
20475:  Proc CanvasSetScale( ACanvas : TCanvas; MulX, MulY, DivX, DivY : Int);
20476:  Proc CanvasSetOffset( ACanvas : TCanvas; OfsX, OfsY : Int);
20477:  Func ColorRecToColor( Color : TKColorRec) : TColor;
20478:  Func ColorToColorRec( Color : TColor) : TKColorRec;
20479:  Func TKColorToGrayScale( Color : TColor) : TColor;
20480:  Func CompareBrushes( ABrush1, ABrush2 : TBrush) :Bool;
20481:  Func CompareFonts( AFont1, AFont2 : TFont) :Bool;
20482:  Proc CopyBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcX, SrcY : Int);
20483:  Func CreateEmptyPoint : TPoint;
20484:  Func CreateEmptyRect : TRect;
20485:  Func CreateEmptyRgn : HRGN;
20486:  Proc DrawAlignedText(Canvas: TCanvas; var ARect : TRect; HAlign : TKHAlign; VAlign : TKVAlign; HPadding,
        VPadding:Int;const AText:TKKString;BackColor:TColor;Attributes:TKTextAttributes);
20487:  Proc TKDrawButtonFrame(ACanvs:TCanvas;const ARect:TRect;AStates:TKButtonDrawStates);
20488:  Proc DrawEdges(Canvas:TCanvas;const R:TRect;HighlightColor,ShadowColor:TColor;Flags:Cardi;
20489:  Proc DrawFilledRectangle(Canvas: TCanvas; const ARect : TRect; BackColor : TColor);
20490:  Proc DrawGradientRect(Canvas:TCanvas; const ARect:TRect;AStartColor,AEndColor TColor; AColorStep:
        Int;AHorizontal:Bool);
20491:  Proc ExcludeShapeFromBaseRect(var BaseRect:TRect; ShapeWidth, ShapeHeight : Int; HAlign : TKHAlign;
        VAlign : TKVAlign; HPadding, VPadding : Int; StretchMode : TKStretchMode; out Bounds, Interior : TRect);
20492:  Func ExtSelectClipRect(DC:HDC; ARect:TRect; Mode:Int; var PrevRgn: HRGN):Bool;
20493:  Func ExtSelectClipRectEx(DC:HDC;ARect:TRect;Mode:Int;CurRgn,PrevRgn : HRGN):Bool;
20494:  Proc FillAroundRect(ACanvas:TCanvas;const Boundary,Interior:TRect;BackColor:TColor);
20495:  Func GetFontHeight( DC : HDC) : Int;
20496:  Func GetFontAscent( DC : HDC) : Int;
20497:  Func GetFontDescent( DC : HDC) : Int;
20498:  Func GDICheck( Value : Int) : Int;
20499:  Func HorizontalShapePosition(AAlignment:TKHAlign;const ABoundary:TRect;const AShapeSize:TPoint):Int;
20500:  Func ImageByType( const Header : TKImageHeaderString) : TGraphic;
20501:  Func IntersectClipRectIndirect( DC : HDC; ARect : TRect) :Bool;
20502:  Func IsBrightColor( Color : TColor) :Bool;
20503:  Proc LoadCustomCursor( Cursor : TCursor; const ResName :Str);
20504:  Proc TKLoadGraphicFromResource(Graphic:TGraphic;const ResName:str;ResType:PChar);
20505:  Func MakeColorRec( R, G, B, A : Byte) : TKColorRec;;
20506:  Func MakeColorRec5( Value : LongWord) : TKColorRec;;
20507:  Func PixelFormatFromBpp( Bpp :Card) : TPixelFormat;
20508:  Func TKRectInRegion( Rgn : HRGN; ARect : TRect) :Bool;
20509:  Func RgnCreateAndGet( DC : HDC) : HRGN;
20510:  Proc RgnSelectAndDelete( DC : HDC; Rgn : HRGN);
20511:  Proc RoundRectangle(ACanvas: TCanvas; const ARect: TRect;AXRadius,AYRadius: Int);
20512:  Proc SafeStretchDraw(ACanvas:TCanvas; ARect:TRect;AGraphic:TGraphic;ABackColor:TColor);
20513:  Proc SelectClipRect( DC : HDC; const ARect : TRect);
```

```
20514:  Proc TKStretchBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcRect : TRect);
20515:  Func SwitchRGBToBGR( Value : TColor) : TColor;
20516:  Proc TranslateRectToDevice( DC : HDC; var ARect : TRect);
20517:  Func VerticalShapePosition(AAlignment:TKVAlign;const ABoundary:TRect;const AShapeSize: TPoint):Int;
20518: end;
20519:
20520: Proc SIRegister_umaxPipes(CL: TPSPascalCompiler);
20521: begin
20522:  ConstantN('cShutDownMsg','String').SetString( 'shutdown pipe ;
20523:  ConstantN(cPipeFormat',String).SetString( '\\%s\pipe\%s;
20524:   SIRegister_TPipeServermax(CL);
20525:   SIRegister_TPipeClientmax(CL);
20526: end;
20527:
20528: Proc SIRegister_KControls(CL: TPSPascalCompiler);
20529: begin
20530: TypeS('TKColorArray', 'array of TColor;
20531: TKPreviewColorIndex', 'Int;
20532: TKPrintOption', '( poCollate, poFitToPage, poMirrorMargins, poPa'
20533:  +'geNumbers, poPaintSelection, poTitle, poUseColor );
20534: TKPrintOptions', 'set of TKPrintOption;
20535: TKPrintRange', '( kprAll, kprSelectedOnly, kprRange );
20536: TKPrintUnits', '( kpuMM, kpuCM, kpuInch, kpuHundredthInch );
20537:  //ConstantN('cBorderStyleDef','').SetString( bsSingle);
20538:  'cContentPaddingBottomDef','LongInt').SetInt( 0);
20539:  'cContentPaddingLeftDef','LongInt').SetInt( 0);
20540:  ConstantN(cOptionsDef','LongInt).Value.ts32:= ord(poFitToPage) or ord(poPageNumbers) or ord(poUseColor);
20541:  //ConstantN('cRangeDef','').SetString( prAll);
20542:  'cScaleDef','LongInt').SetInt( 100);
20543:  'cScaleMin','LongInt').SetInt( 10);
20544:  'cScaleMax','LongInt').SetInt( 500);
20545:  //ConstantN('cUnitsDef','').SetString( puCM);
20546:  //ConstantN('cPaperDef','').SetString( clWhite);
20547:  //ConstantN('cBkGndDef','').SetString( clAppWorkSpace);
20548:  //ConstantN('cBorderDef','').SetString( clBlack);
20549:  //ConstantN('cSelectedBorderDef','').SetString( clNavy);
20550:  ConstantN('ciPaper','LongInt').SetInt( TKPreviewColorIndex ( 0 ));
20551:  'ciBkGnd','LongInt').SetInt( TKPreviewColorIndex ( 1 ));
20552:  'ciBorder','LongInt').SetInt( TKPreviewColorIndex ( 2 ));
20553:  'ciSelectedBorder','LongInt').SetInt( TKPreviewColorIndex ( 3 ));
20554:  // 'ciPreviewColorsMax','').SetString( ciSelectedBorder);
20555:  'cScrollNoAction','LongInt').SetInt( - 1);
20556:  'cScrollDelta','LongInt').SetInt( - 2);
20557:  'cPF_Dragging','LongWord').SetUInt( $00000001);
20558:  'cPF_UpdateRange','LongWord').SetUInt( $00000002);
20559:   TKPreviewScaleMode', '( smScale, smPageWidth, smWholePage );
20560:   TKPreviewChangedEvent', 'Proc ( Sender : TObject);
20561:   TKPrintMeasureInfo', 'record OutlineWidth : Int; OutlineHeig'
20562:    ht : Int; ControlHorzPageCount : Int; ControlVertPageCount : Integ'
20563:    er; ExtraLeftHorzPageCount : Int; ExtraLeftVertPageCount : Int; Ex'
20564:    traRightHorzPageCount : Int; ExtraRightVertPageCount : Int; end;
20565:   TKPrintStatus', '( kepsBegin, kepsNewPage, kepsEnd );
20566:   TKPrintNotifyEvent,Procedure(Sender:TObject;Status: TKPrintStatus;var Abort:Bool);
20567:   TKPrintPaintEvent', 'Proc ( Sender : TObject);
20568:   ClassN(CL.FindClass('TOBJECT'),TKPrintPageSetup;
20569:   ClassN(CL.FindClass('TOBJECT'),TKPrintPreview;
20570:   SIRegister_TKRect(CL);
20571:   ClassN(CL.FindClass('TOBJECT'),'TKObjectList;
20572:   SIRegister_TKObject(CL);
20573:   //TKObjectClass', 'class of TKObject;
20574:   SIRegister_TKObjectList(CL);
20575:   SIRegister_TKCustomControl(CL);
20576:   TKColorScheme', '( kcsNormal, kcsGrayed, kcsBright, kcsGrayScale );
20577:   TKColorIndex', 'Int;
20578:   TKColorData', 'record Index:TKColorIndex;Color:TColor; Default:TColor; Name:str; end;
20579:   TKColorSpec', 'record Def : TColor; Name :Str; end;
20580:   SIRegister_TKCustomColors(CL);
20581:   TKPrintMeasureEvent', 'Proc ( Sender : TObject; var Info : TKPrintMeasureInfo);
20582:   SIRegister_TKPrintPageSetup(CL);
20583:   SIRegister_TKPreviewColors(CL);
20584:   SIRegister_TKPrintPreview(CL);
20585:  Func InchesToValue(Units : TKPrintUnits; Value : Double) : Double;
20586:  Func ValueToInches(Units : TKPrintUnits; Value : Double) : Double;
20587: end;
20588:
20589: Proc SIRegister_IdAntiFreezeBase(CL: TPSPascalCompiler);
20590: begin
20591:  ConstantN(ID_Default_TIdAntiFreezeBase_Active,Boolean).SetInt(1);
20592:  ConstantN('ID_Default_TIdAntiFreezeBase_ApplicationHasPriority','Boolean').SetInt(1);
20593:  ConstantN('ID_Default_TIdAntiFreezeBase_IdleTimeOut','LongInt').SetInt( 250);
20594:  ConstantN('ID_Default_TIdAntiFreezeBase_OnlyWhenIdle','Boolean').SetInt( 250);
20595:  SIRegister_TIdAntiFreezeBase(CL);
20596: end;
20597:
20598: Proc SIRegister_OverbyteIcsConApp(CL: TPSPascalCompiler);
20599: begin
20600:  ConstantN('WM_STARTUP','LongInt').SetInt( WM_USER + 789);
20601:  SIRegister_TKeyboardThread(CL);
20602:  //TypeS('TConApplicationClass', 'class of TConApplication;
```

```
20603:  SIRegister_TConApplication(CL);
20604: end;
20605:
20606: Proc SIRegister_KMemo(CL: TPSPascalCompiler);
20607: begin
20608:  'cUndoLimitMin','LongInt').SetInt( 100);
20609:  cUndoLimitMax','LongInt).SetInt( 10000);
20610:  cUndoLimitDef','LongInt).SetInt( 1000);
20611:  cScrollPaddingMin','LongInt).SetInt( 0);
20612:  cScrollPaddingMax','LongInt).SetInt( 1000);
20613:  cScrollPaddingDef','LongInt).SetInt( 30);
20614:  cScrollSpeedMin','LongInt).SetInt( 50);
20615:  cScrollSpeedMax','LongInt).SetInt( 1000);
20616:  cScrollSpeedDef','LongInt).SetInt( 100);
20617:  //cBkGndDef','').SetString( clWindow);
20618:  //cInactiveCaretBkGndDef','').SetString( clBlack);
20619:  //cInactiveCaretSelBkGndDef','').SetString( clBlack);
20620:  //cInactiveCaretSelTextDef','').SetString( clYellow);
20621:  //cInactiveCaretTextDef','').SetString( clYellow);
20622:  //cSelBkGndDef','').SetString( clGrayText);
20623:  //cSelBkGndFocusedDef','').SetString( clHighlight);
20624:  //cSelTextDef','').SetString( clHighlightText);
20625:  //cSelTextFocusedDef','').SetString( clHighlightText);
20626:  ciBkGnd','LongInt).SetInt( TKColorIndex ( 0 ));
20627:  ciInactiveCaretBkGnd','LongInt).SetInt( TKColorIndex ( 1 ));
20628:  ciInactiveCaretSelBkGnd','LongInt).SetInt( TKColorIndex ( 2 ));
20629:  ciInactiveCaretSelText','LongInt).SetInt( TKColorIndex ( 3 ));
20630:  ciInactiveCaretText','LongInt).SetInt( TKColorIndex ( 4 ));
20631:  ciSelBkGnd','LongInt).SetInt( TKColorIndex ( 5 ));
20632:  ciSelBkGndFocused','LongInt).SetInt( TKColorIndex ( 6 ));
20633:  ciSelText','LongInt).SetInt( TKColorIndex ( 7 ));
20634:  ciSelTextFocused','LongInt).SetInt( TKColorIndex ( 8 ));
20635:  //ciMemoColorsMax','').SetString( ciSelTextFocused);
20636:  cInvalidListID','LongInt).SetInt( - 1);
20637:  cHorzScrollStepDef',LongInt').SetInt( 4);
20638:  cVertScrollStepDef',LongInt').SetInt( 10);
20639:  //cHeight','LongInt').SetInt( 200);
20640:  //cWidth','LongInt').SetInt( 300);
20641:  cNewLineChar','Char).SetString( #$B6);
20642:  cSpaceChar','Char).SetString( #$B7);
20643:  cTabChar','Char).SetString( #$2192);
20644:  cBullet','Char).SetString( #$2022);
20645:  cSquareBullet,Char').SetString( #$25AB);
20646:  cArrowBullet',Char').SetString( #$25BA);
20647:  //cDefaultWordBreaks,String').SetString(' ' or  '/' or '\' or ';' or  ':' or  '?' or  '!;
20648:  cRichText','String').SetString( 'Rich Text Format;
20649:  ClassN(CL.FindClass(TOBJECT'),'TKCustomMemo;
20650:  TypeS('TKMemoLinePosition', '( eolInside, eolEnd );
20651:  TKMemoBlockPosition', '( mbpText, mbpRelative, mbpAbsolute );
20652:  TKMemoState', '( elCaretCreated, elCaretVisible, elCaretUpdate, '
20653:    +elIgnoreNextChar, elModified, elMouseCapture, elOverwrite, elReadOnly );
20654:   TKMemoStates', 'set of TKMemoState;
20655:   TKMemoUpdateReason', '( muContent, muExtent, muSelection, muSelectionScroll );
20656:   TKMemoUpdateReasons', 'set of TKMemoUpdateReason;
20657:   SIRegister_TKMemoSparseItem(CL);
20658:   SIRegister_TKMemoSparseList(CL);
20659:   SIRegister_TKMemoSparseStack(CL);
20660:   SIRegister_TKMemoDictionaryItem(CL);
20661:   SIRegister_TKMemoDictionary(CL);
20662:   TKMemoParaNumbering', '( pnuNone, pnuBullets, pnuArabic, pnuLett'
20663:    +erLo, pnuLetterHi, pnuRomanLo, pnuRomanHi );
20664:   SIRegister_TKMemoNumberingFormatItem(CL);
20665:   SIRegister_TKMemoNumberingFormat(CL);
20666:   ClassN(CL.FindClass('TOBJECT'),'TKMemoListLevels;
20667:   SIRegister_TKMemoListLevel(CL);
20668:   ClassN(CL.FindClass('TOBJECT'),'TKMemoList;
20669:   SIRegister_TKMemoListLevels(CL);
20670:   ClassN(CL.FindClass('TOBJECT'),'TKMemoListTable;
20671:   SIRegister_TKMemoList(CL);
20672:   TKMemoListChangedEvent', 'Proc (AList: TKMemoList; ALevel: TKMemoListLevel);
20673:   SIRegister_TKMemoListTable(CL);
20674:   TKMemoScriptCapitals', '( tcaNone, tcaNormal, tcaSmall );
20675:   TKMemoScriptPosition', '( tpoNormal, tpoSuperscript, tpoSubscript );
20676:   SIRegister_TKMemoTextStyle(CL);
20677:   TKMemoBlockWrapMode', '( wrAround, wrAroundLeft, wrAroundRight, '
20678:    +wrTight, wrTightLeft, wrTightRight, wrTopBottom, wrNone, wrUnknown );
20679:   TKMemoBlockStyleChangedEvent', 'Proc ( Sender : TObject; AR'
20680:    +'easons : TKMemoUpdateReasons);
20681:   SIRegister_TKMemoBlockStyle(CL);
20682:   TKMemoLineSpacingMode', '( lsmFactor, lsmValue );
20683:   SIRegister_TKMemoParaStyle(CL);
20684:   SIRegister_TKMemoLine(CL);
20685:   SIRegister_TKMemoLines(CL);
20686:   SIRegister_TKMemoWord(CL);
20687:   SIRegister_TKMemoWordList(CL);
20688:   ClassN(CL.FindClass('TOBJECT'),'TKMemoBlocks;
20689:   TKMemoMouseAction', '( maMove, maLeftDown, maLeftUp, maRightDown'
20690:    +', maRightUp, maMidDown, maMidUp );
20691:   //TKMemoBlockClass', 'class of TKMemoBlock;
```

```
20692:   SIRegister_TKMemoBlock(CL);
20693:   SIRegister_TKMemoSingleton(CL);
20694:   SIRegister_TKMemoTextBlock(CL);
20695:   SIRegister_TKMemoHyperlink(CL);
20696:   SIRegister_TKMemoParagraph(CL);
20697:   SIRegister_TKMemoImageBlock(CL);
20698:   SIRegister_TKMemoContainer(CL);
20699:   ClassN(CL.FindClass('TOBJECT'),'TKMemoTable;
20700:   ClassN(CL.FindClass('TOBJECT'),'TKMemoTableRow;
20701:   SIRegister_TKMemoTableCell(CL);
20702:   SIRegister_TKMemoTableRow(CL);
20703:   SIRegister_TKMemoTable(CL);
20704:   TKMemoUpdateEvent', 'Proc ( Reasons : TKMemoUpdateReasons);
20705:   SIRegister_TKMemoBlocks(CL);
20706:   SIRegister_TKMemoColors(CL);
20707:   TKMemoChangeKind', '( ckCaretPos, ckDelete, ckInsert );
20708:   TKMemoUndoChangeEvent','Procedure(Sender : TObject; ItemReason: TKMemoChangeKind);
20709:   TKMemoChangeItem', 'record Blocks : TKMemoBlocks; Group : Cardin'
20710:    al; GroupKind : TKMemoChangeKind; Inserted :Bool; ItemKind : TKMemoCha'
20711:    ngeKind; Position : Int; end;
20712:   //PKMemoChangeItem', '^TKMemoChangeItem // will not work;
20713:   SIRegister_TKMemoChangeList(CL);
20714:   TKMemoRTFString', 'string;
20715:   SIRegister_TKCustomMemo(CL);
20716:   SIRegister_TKMemo(CL);
20717:   SIRegister_TKMemoEditAction(CL);
20718:   SIRegister_TKMemoEditCopyAction(CL);
20719:   SIRegister_TKMemoEditCutAction(CL);
20720:   SIRegister_TKMemoEditPasteAction(CL);
20721:   SIRegister_TKMemoEditSelectAllAction(CL);
20722:  Func NewLineChar : TKkString;
20723:  Func SpaceChar : TKkString;
20724:  Func TabChar : TKkString;
20725: end;
20726:
20727: Proc SIRegister_OverbyteIcsTicks64(CL: TPSPascalCompiler);
20728: begin
20729:  ConstantN('ISODateMask',String).SetString( 'yyyy-mm-dd;
20730:  'ISODateTimeMask','String').SetString( 'yyyy-mm-dd"T"hh:nn:ss;
20731:  'ISODateLongTimeMask','String').SetString( 'yyyy-mm-dd"T"hh:nn:ss.zzz;
20732:  'ISOTimeMask','String').SetString( 'hh:nn:ss;
20733:  'LongTimeMask','String').SetString( 'hh:nn:ss:zzz;
20734:  'Ticks64PerDay','int64').SetInt64( 24 * 60 * 60 * 1000);
20735:  'Ticks64PerHour','int64').SetInt64( 60 * 60 * 1000);
20736:  'Ticks64PerMinute','int64').SetInt64( 60 * 1000);
20737:  'Ticks64PerSecond','int64').SetInt64( 1000);
20738:   TypeS('TTicks64Mode', ( TicksNone, TicksAPI64, TicksPerf, TicksAPI32 );
20739:  DelphiFunction(Func IcsGetTickCount64 : int64;
20740:  Proc IcsInitTick64( NewMode : TTicks64Mode);
20741:  Func IcsNowPC : TDateTime;
20742:  Proc IcsAlignNowPC;
20743:  Func IcsLastBootDT : TDateTime;
20744:  Func IcsGetPerfCountsPerSec : int64;
20745:  Func IcsPerfCountCurrent : int64;
20746:  Func IcsPerfCountCurrMilli : int64;
20747:  Func IcsPerfCountToMilli( LI : int64) : int64;
20748:  Func IcsPerfCountGetMilli( startLI : int64) : int64;
20749:  Func IcsPerfCountGetMillStr( startLI : int64) :Str;
20750:  Func IcsPerfCountToSecs( LI : int64) : Int;
20751:  Func IcsPerfCountGetSecs( startLI : int64) : Int;
20752:  Func IcsDiffTicks64( const StartTick, EndTick : int64) : int64;
20753:  Func IcsElapsedTicks64( const StartTick : int64) : int64;
20754:  Func IcsElapsedMsecs64( const StartTick : int64) : int64;
20755:  Func IcsElapsedSecs64( const StartTick : int64) : Int;
20756:  Func IcsElapsedMins64( const StartTick : int64) : Int;
20757:  Func IcsWaitingSecs64( const EndTick : int64) : Int;
20758:  Func IcsGetTrgMSecs64( const MilliSecs : int64) : int64;
20759:  Func IcsGetTrgSecs64( const DurSecs : Int) : int64;
20760:  Func IcsGetTrgMins64( const DurMins : Int) : int64;
20761:  Func IcsTestTrgTick64( const TrgTick : int64) :Bool;
20762:  Func IcsAddTrgMsecs64( const TickCount, MilliSecs : int64) : int64;
20763:  Func IcsAddTrgSecs64( const TickCount : int64; DurSecs : Int) : int64;
20764: end;
20765:
20766: Proc SIRegister_OverbyteIcsSha1(CL: TPSPascalCompiler);
20767: begin
20768:  ConstantN(IcsSHA1Version','LongInt.SetInt( 800);
20769:  CopyRight','String.SetString(  IcsSHA1 (c) 2004-2012 F. Piette V8.00 ;
20770:  shaSuccess','LongInt.SetInt( 0);
20771:  shaNull','LongInt.SetInt( 1);
20772:  shaInputTooLong','LongInt.SetInt( 2);
20773:  shaStateError','LongInt.SetInt( 3);
20774:  SHA1HashSize','LongInt.SetInt( 20);
20775:   TypeS('uint32_t', 'LongWord;
20776:   TypeS('uint8_t', 'Byte;
20777:   TypeS('int_least16_t', 'LongInt;
20778:   TypeS('SHA1DigestString', 'Ansistr;
20779:  Func SHA1Reset( var context : SHA1Context) : Int;
20780: Func SHA1Input(var context:SHA1Context;const message_array:str;message_array:PAnsiChar;length
       Cardinal):Int;
```

```
20781:  Func SHA1Result( var context : SHA1Context; var Message_Digest : SHA1Digest) : Int;
20782:  Func SHA1ofStr( const s : Ansistr) : SHA1DigestString;
20783:  Func SHA1ofBuf( const buf, buflen : Int) : SHA1DigestString;
20784:  Func SHA1ofStream( const strm : TStream) : SHA1DigestString;
20785:  Func SHA1toHex( const digest : SHA1DigestString) :Str;
20786:  Func SHA1DigestToLowerHex( const Digest : SHA1Digest) :Str;
20787:  Func SHA1DigestToLowerHexA( const Digest : SHA1Digest) : RawByteString;
20788:  Func SHA1DigestToLowerHexW( const Digest : SHA1Digest) : UnicodeString;
20789:  Proc HMAC_SHA1(const Data,DataLen:Int;const Key,KeyLen:Int;out Digest:SHA1Digest);
20790:  Func HMAC_SHA1_EX( const Data : Ansistr; const Key : Ansistr) : Ansistr;
20791: end;
20792:
20793: Proc SIRegister_KEditCommon(CL: TPSPascalCompiler);
20794: begin
20795:  cCharMappingSize','LongInt.SetInt( 256);
20796:  TKEditCommandk', '( ecNonek, ecLeftk, ecRightk, ecUpk, ecDownk, ecLinek'
20797:   Start, ecLineEndk, ecPageUpk, ecPageDownk, ecPageLeftk, ecPageRightk, ecPageTopk,
20798:    htk, ecSelUpk, ecSelDownk, ecSelLineStartk, ecSelLineEndk, ecSelPageUpk, ecSelPa'
20799:    geDownk, ecSelPageLeftk, ecSelPageRightk, ecSelPageTopk, ecSelPageBottomk, ecSe'
20800:    lEditorTopk, ecSelEditorBottomk, ecSelGotoXYk, ecScrollUpk, ecScrollDownk, ecSc'
20801:    rollLeftk, ecScrollRightk, ecScrollCenterk, ecUndok, ecRedok, ecCopyk, ecCutk, ec'
20802:    +Pastek, ecInsertChark, ecInsertDigitsk, ecInsertStringk, ecInsertNewLinek, ecDe'
20803:    +leteLastChark, ecDeleteChark, ecDeleteBOLk, ecDeleteEOLk, ecDeleteLinek, ecSele'
20804:    +ctAllk, ecClearAllk, ecClearIndexSelectionk, ecClearSelectionk, ecSearchk, ecRe'
20805:    +'placek, ecInsertModek, ecOverwriteModek, ecToggleModek, ecGotFocusk, ecLostFocusk ); *)
20806:   TKEditDisabledDrawStyle', '( eddBright, eddGrayed, eddNormal );
20807:   TKEditKey', 'record Key : Word; Shift : TShiftState; end;
20808:   TKEditCommandAssignment', 'record Key : TKEditKey; Command : TKEditCommand; end;
20809:   TKEditCommandMap', 'array of TKEditCommandAssignment;
20810:   TKEditDropFilesEvent','Procedure(Sender:TObject;X,Y:Int;Files:TStrings);
20811:   SIRegister_TKEditKeyMapping(CL);
20812:   TKEditCharMapping', 'array of AnsiChar;
20813:   //TypeS('PKEditCharMapping', '^TKEditCharMapping // will not work;
20814:   //TypeS('TKEditOption', '( eoDropFiles, eoGroupUndo, eoUndoAfterSave, eoS'
20815:    //+'howFormatting, eoWantTab );
20816:   //TypeS('TKEditOptions', 'set of TKEditOption;
20817:   TKEditReplaceAction', '( eraCancel, eraYes, eraNo, eraAll );
20818:   TKEditReplaceTextEvent', 'Proc ( Sender : TObject; const Te'
20819:    xtToFind, TextToReplace :Str; var Action : TKEditReplaceAction);
20820:   TKEditSearchError', '( eseOk, eseNoDigitsFind, eseNoDigitsReplace, eseNoMatch );
20821:   TKEditSearchOption', '( esoAll, esoBackwards, esoEntireScope, es'
20822:    oFirstSearch,esoMatchCase,esoPrompt,esoSelectedOnly,esoTreatAsDigits,esoWereDigits );
20823:   TKEditSearchOptions', 'set of TKEditSearchOption;
20824:   TKEditSearchData', 'record ErrorReason : TKEditSearchError; Opti'
20825:    ons:TKEditSearchOptions;SelStart:Int;SelEnd:Int;TextToFind:str;TextToReplace:str; end;
20826:  //TypeS('PKEditSearchData', '^TKEditSearchData // will not work;
20827:  //cEditDisabledDrawStyleDef','.SetString( eddBright);
20828:  Func CreateDefaultKeyMapping : TKEditKeyMapping;
20829:  Func DefaultCharMapping : TKEditCharMapping;
20830:  Func DefaultSearchData : TKEditSearchData;
20831: end;
20832:
20833: unit UtilsMax4; unit uPSI_UtilsMax4;
20834:  Func AllDigitsDifferent(N: Int64):Bool;
20835:  Proc DecimalToFraction(Decimal: Extended; out FractionNumerator: Extended;
20836:            out FractionDenominator: Extended; const AccuracyFactor: Extended);
20837:  Func ColorToHTML(const Color: TColor):Str;
20838:  Func DOSCommand(const CommandLine:Str; const CmdShow:Int;
20839:             const WaitUntilComplete:Bool; const WorkingDir:Str = ':Bool;
20840:  Func GetDosOutput(CommandLine:Str; Work:Str = 'C:\:Str;
20841:  Ex.: writeln(GetDosOutput('java -version','C:\'));
20842:       >>>java version "1.8.0_211"
20843:          Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
20844:          Java HotSpot(TM) Client VM (build 25.211-b12, mixed mode)
20845:  Proc CaptureConsoleOutput(DosApp :Str;AMemo : TMemo);
20846:  Ex.: CaptureConsoleOutput('cmd /C dir *.* ',memo2);
20847:       CaptureConsoleOutput('powershell /C dir *.* ',memo2);
20848:  Func ExecuteCommandDOS(CommandLine:str):str;
20849:  Func DOSCommandRedirect(const CommandLine:Str;
20850:            const OutStream: Classes.TStream):Bool; overload;      //8
20851: Proc SendKeysToWindow(const HWnd: Windows.HWND; const Text:Str);
20852: Func IsRunningOnBattery:Bool;
20853: Func IsHexStr(const S:Str):Bool;
20854: Func IsCharInSet(const Ch: Char; const Chars: TCharSet):Bool;
20855: Func StreamHasWatermark(const Stm:Classes.TStream;const Watermark:array of Byte):Bool;
20856: Func ReadBigEndianWord(Stm: Classes.TStream): Word;
20857: Func DownloadURLToFile(const URL, FileName:Str):Bool;
20858: Func ExtractURIQueryString(const URI:Str):Str;
20859: Func GetBiosVendor:Str;
20860: Func GetIEVersionStr:Str;          //18
20861: Func FloatToFixed(const Value: Extended; const DecimalPlaces: Byte;
20862:                              const SeparateThousands:Bool):Str;
20863: Func IntToFixed(const Value: Int;const SeparateThousands:Bool):Str;
20864: Func Int64ToFixed(const Value: Int64;const SeparateThousands:Bool):Str;
20865: Func IntToNumberText2(const Value: Int):Str;      //22
20866:
20867: Func IsLibraryInstalled2(const LibFileName:Str):Bool;
20868: Func RemainingBatteryPercent: Int;
20869: Proc SetLockKeyState(KeyCode: Int; IsOn:Bool);
```

```
20870: Func IsLockKeyOn(const KeyCode: Int):Bool;
20871: Proc PostKeyEx322(const Key: Word; const Shift: Classes.TShiftState;
20872:                           const SpecialKey:Bool = False);
20873: Func TerminateProcessByID(ProcessID:Card):Bool;
20874: Func GetWindowProcessName(const Wnd: Windows.HWND):Str;
20875: Func GetProcessName(const PID: Windows.DWORD):Str;
20876: Func GetWindowProcessID(const Wnd: Windows.HWND): Windows.DWORD;
20877: Func IsAppResponding(const Wnd: Windows.HWND):Bool;
20878: Func IsTabletOS:Bool;
20879: Func ProgIDInstalled(const PID:Str):Bool;
20880: Func GetProcessorName:Str;
20881: Func GetProcessorIdentifier:Str;                    //36
20882: Proc EmptyKeyQueue;
20883: Proc TrimAppMemorySize;
20884: Func GetEnvironmentBlockSize:Card;
20885: Func GetDefaultPrinterName:Str;                     //40
20886: Proc ListDrives(const List: Classes.TStrings);
20887: Proc MultiSzToStrings(const MultiSz: PChar; const Strings: Classes.TStrings);
20888: Func BrowseURL(const URL:Str):Bool;
20889: Func IsValidURLProtocol(const URL:Str):Bool;
20890: Func ExecAssociatedApp(const FileName:Str):Bool;   //45
20891: Func CheckInternetConnection(AHost: PAnsiChar):Bool;
20892: Func MakeSafeHTMLText(TheText:Str):Str;
20893: Func RemoveURIQueryString(const URI:Str):Str;
20894: Func GetRegistryString(const RootKey: Windows.HKEY;const SubKey,Name:str):str;
20895: Proc RefreshEnvironment2(const Timeout:Card = 5000);  //50
20896: Func IsKeyPressed2(const VirtKeyCode: Int):Bool;
20897: Func SizeOfFile64(const FileName:Str): Int64;
20898: Func IsHugeFile(const FileName:Str):Bool;
20899:  Func SetTransparencyLevel(const HWnd: Windows.HWND;const Level: Byte):Bool;
20900: Func IsEqualResID(const R1, R2: PChar):Bool;         //55
20901: Func GetGenericFileType(const FileNameOrExt:Str):Str;
20902: Func GetFileType2(const FilePath:Str):Str;
20903: Proc ShowShellPropertiesDlg(const APath:Str);
20904: Func EllipsifyText(const AsPath:Bool; const Text:Str;
20905:          const Canvas: Graphics.TCanvas; const MaxWidth: Int ):Str;
20906: Func CloneByteArray(const B: array of Byte): TBytes;       //60
20907: Proc AppendByteArray(var B1: TBytes; const B2: array of Byte);
20908: Func IsUnicodeStream(const Stm: Classes.TStream):Bool;
20909: Func FileHasWatermark(const FileName:Str;
20910:             const Watermark: array of Byte; const Offset: Int = 0):Bool;
20911: Func FileHasWatermarkAnsi(const FileName:Str;
20912:            const Watermark: Ansistr; const Offset: Int = 0):Bool;
20913: Func IsASCIIStream(const Stm: Classes.TStream; Count:Int64 = 0;
20914:                              BufSize: Int = 8*1024):Bool;
20915: Func IsASCIIFile(const FileName:str; BytesToCheck:Int64 = 0;
20916:                            BufSize: Int = 8*1024):Bool;      //66
20917: Func BytesToAnsistr(const Bytes: SysUtils.TBytes; const CodePage: Word):str;
20918: Func UnicodeStreamToWideString(const Stm: Classes.TStream): WideString;
20919: Proc WideStringToUnicodeStream(const Str: WideString;const Stm: Classes.TStream);
20920: Proc GraphicToBitmap(const Src: Graphics.TGraphic;
20921:           const Dest: Graphics.TBitmap; const TransparentColor: Graphics.TColor);  //70
20922: Func URIDecode(S:Str; const IsQueryString:Bool):Str;
20923: Func URIEncode(const S:Str; const InQueryString:Bool):Str;
20924: Func URLDecode(const S:Str):Str;
20925: Func URLEncode(const S:Str; const InQueryString:Bool):Str;
20926: Func AllDigitsSame(N: Int64):Bool;
20927: Func GetDosOutput(CommandLine:Str; Work:Str = 'C:\:Str;
20928: Proc CaptureConsoleOutput(DosApp :Str;AMemo : TMemo);
20929:  Ex.: CaptureConsoleOutput('cmd /C dir *.* ',memo2);
20930:     CaptureConsoleOutput('powershell /C dir *.* ',memo2);
20931: Func ExecuteCommandDOS(CommandLine:str):str;
20932: Func DOSCommandRedirect(const CommandLine:Str;
20933:          const OutStream: Classes.TStream):Bool; overload;     //8
20934: Proc SendKeysToWindow(const HWnd: Windows.HWND; const Text:Str);  //75
20935:  Func FoldWrapText(const Line,BreakStr:str;BreakChars:TSysCharSet;MaxCol:Int):str;;
20936:  Func TextWrap(const Text:Str; const Width, Margin: Int):Str;
20937:  Func ShellExecuteX(Operation,FileName,Parameters,Directory:str;ShowCmd:Int):Card;;
20938:  Func KeyboardStateToShiftState1(const KeyboardState: TKeyboardState): TShiftState;;
20939:  Proc SetCustomFormGlassFrame(const CustomForm:TCustomForm;const GlassFrame:TGlassFrame);
20940:  Func GetCustomFormGlassFrame( const CustomForm : TCustomForm) : TGlassFrame;
20941:  Proc SetApplicationMainFormOnTaskBar(const Application:TApplication;Value:Bool);
20942:  Func GetApplicationMainFormOnTaskBar( const Application : TApplication) :Bool;
20943:  Func CompressWhiteSpace(const S:Str):Str;
20944:  Func IsASCIIDigit(const Ch: Char):Bool;
20945:   Func CompareNumberStr(const S1, S2:Str): Int;
20946:  Proc HexToBuf(HexStr:Str; var Buf:Str);
20947:   Func BufToHex(const Buf:Str; const Size:Card):Str;
20948:  Func DOSExec( CommandLine :Str; Work :Str) :Str;
20949:  Proc HexToStrBuf(HexStr:Str; var Buf:Str);;
20950:  Func StrBufToHex(const Buf:Str; const Size:Card):Str;;
20951:  Func GetCharFromVirtualKey(AKey: Word):Str;
20952:  Func GetParentProcessID(const PID: Windows.DWORD): Windows.DWORD;
20953:  Func FormInstanceCount2(AFormClass: Forms.TFormClass): Int; //overload;
20954:  Func FormInstanceCount(const AFormClassName:Str): Int; //overload;
20955:  Func FindAssociatedApp(const Doc:Str):Str;
20956:  Func CreateShellLink2(const LinkFileName,AssocFileName,Desc,WorkDir,Args,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 FormInstanceCount2(AFormClass: TFormClass): Int;;
20963: Func FormInstanceCount(const AFormClassName:Str): Int;;
20964: Func FindAssociatedApp(const Doc:Str):Str;;
20965: Func CreateShellLink2(const LinkFileName,AssocFileName,Desc,WorkDir,Args
20966:    ,IconFileName:Str; const IconIdx: Int):Bool;
20967: Func FileFromShellLink(const LinkFileName:Str):Str;
20968: Func IsShellLink(const LinkFileName:Str):Bool;
20969: Func ResourceIDToStr(const ResID: PChar):Str;
20970: Func IsEqualResID(const R1, R2: PChar):Bool;
20971: Func RecycleBinInfo(const Drive: Char; out BinSize, FileCount: Int64):Bool;
20972: Func SysImageListHandle(const Path:str; const WantLargeIcons:Boolean): THandle;
20973: Func EmptyRecycleBin:Bool;
20974: Func ExploreFile(const Filename:Str ):Bool;
20975: Func ExploreFolder(const Folder:Str):Bool;
20976: Proc ClearRecentDocs2;
20977: Proc AddToRecentDocs2(const FileName:Str);                    //100
20978: Func StringsToMultiSz(const Strings:TStrings;const MultiSz:PChar;const BufSize:Int):Int;
20979: Proc DrawTextOutline(const Canvas: TCanvas;const X,Y:Int;const Text:str);
20980: Func CloneGraphicAsBitmap(const Src:Graphics.TGraphic; const PixelFmt:Graphics.TPixelFormat;
20981: const TransparentColor: Graphics.TColor): Graphics.TBitmap;
20982: Proc InvertBitmap(const ABitmap: Graphics.TBitmap); //overload;
20983: Proc InvertBitmap2(const SrcBmp, DestBmp: Graphics.TBitmap); //overload;
20984: Func MakeFilenameInExePath( aFilename : TFilename) : TFilename;
20985: Func YearOfDate2( DateTime : TDateTime) : Int;
20986: Func MonthOfDate2( DateTime : TDateTime) : Int;
20987: Func DayOfDate2( DateTime : TDateTime) : Int;
20988: Func HourOfTime2( DateTime : TDateTime) : Int;
20989: Func MinuteOfTime2( DateTime : TDateTime) : Int;
20990: Func SecondOfTime2( DateTime : TDateTime) : Int;
20991: Func IsLeapYear2( DateTime : TDateTime) :Bool;
20992: Func DaysInMonth2( DateTime : TDateTime) : Int;
20993: Func MakeUTCTime( DateTime : TDateTime) : TDateTime;
20994: Func MakeLocalTimeFromUTC( DateTime : TDateTime) : TDateTime;
20995: Func IsStandardTime :Bool;
20996: Func UnixNow : Int64;
20997: Func NowString :Str;
20998: Func MakeClosedTag( aTagName, aTagValue :Str) :Str;
20999: Func MakeOpenTag( aTagName, aTagAttributes :Str) :Str;
21000: Func MakeBold( Str :Str) :Str;
21001: Func MakeItalic( Str :Str) :Str;
21002: Func MakeUnderline( Str :Str) :Str;
21003: Func MakeStrikeout( Str :Str) :Str;
21004: Func MakeCenter( Str :Str) :Str;
21005: Func MakeParagraph( Str :Str) :Str;
21006: Func MakeCode( Str :Str) :Str;
21007: Func MakeOption( aValue, aText :Str) :Str;
21008: Func MakeHTMLFontSize( Str :Str; SizeParam :Str) :Str;
21009: Func AddQuotes2( Str :Str) :Str;
21010: Func AddSingleQuotes( Str :Str) :Str;
21011: Func MakeHTMLParam( Str :Str) :Str;
21012: Func MakeLink( URL, name :Str) :Str;
21013: Func MakeLinkTarget( URL, name, Target :Str) :Str;
21014: Func MakeMailTo( Address, name :Str) :Str;
21015: Func HTMLToDelphiColor( S :Str) : TColor;
21016: Func ColorToHTMLHex( Color : TColor) :Str;
21017: Func GetStringFromRes( ResName :Str) :Str;
21018: Func EscapeText( sText :Str) :Str;
21019: Func EncodeForXML( const aString :Str) :Str;
21020: Func IsStringAlpha( Str :Str) :Bool;
21021: Func IsStringNumber( Str :Str) :Bool;
21022: Func EnsurePrefix( aPrefix, aText :Str) :Str;
21023: Func StringToAcceptableChars( S :Str; AcceptableChars : TCharSet) :Str;
21024: Func StringIsAcceptable( S :Str; AcceptableChars : TCharSet) :Bool;
21025: Func ValidateEMailAddress( aEmail :Str) :Bool;
21026: Func FirstChar( Str :Str) : Char;
21027: Func LastChar( Str :Str) : Char;
21028: Func StringIsEmpty( Str :Str) :Bool;
21029: Func StringIsNotEmpty( Str :Str) :Bool;
21030: Func StringHasSpacesInMiddle( Str :Str) :Bool;
21031: Func StringContains( aString :Str; aSubStr :Str) :Bool;
21032: Func SpacesToUnderscore( S :Str) :Str;
21033: Func SpacesToPluses( Str :Str) :Str;
21034: Func SwapString( Str :Str) :Str;
21035: Func MakeCopyrightNotice( aCopyrightHolder :Str) :Str;
21036: Proc WriteStringToFile( aStr :Str; aFilename : TFilename);
21037: //Func WinExecandWait32( Path : PChar; Visibility : Word) : Int;
21038: Func WinExecAndWait32V2( Filename :Str; Visibility : Int) : DWORD;
21039: Func WindowsExit( RebootParam : Longword) :Bool;
21040: Func GetVersionInfo2 :Str;
21041: Func VersionString( aPrefix :Str; aUseColon :Bool) :Str;
21042: Func CreateTempFileName( aPrefix :Str) :Str;
21043: Func GetWindowsTempDir :Str;
21044: Func GetWindowsDir2 :Str;
21045: Func GetSystemDir2 :Str;
21046: Func GetSpecialFolderLocation( aFolderType : Int) :Str;
```

```
21047: Func GetTextFileContents( aFilename : TFilename) :Str;
21048: Func GetBDSDir( aVersion : Int) :Str;
21049: Func CaptionMessageDlg(const aCaption:str;const Msg:str;DlgType:TMsgDlgType;Buttons
       TMsgDlgButtons;HelpCtx:Longint):Int;
21050: Func StreamFileCopy(const SourceFilename,TargetFilename:str;KeepDate:Boolean):Int;
21051: Func MakePercentString( f : Double):Str;
21052: Proc DumpKey( var aKey : Char);
21053: Func ValidateKey2(var aKey:Char; AcceptableKeys:TCharSet;KillTheKey:Bool) :Bool;
21054: Func MakeInterbaseString( aHostName, aFilename :Str) :Str;
21055: Func ParseToken( const S :Str; var FromPos : Int; Delimiter : Char):Str;
21056: Proc MatchBounds( MovedControl, TemplateControl : TControl);
21057: Proc LockWidth( aControl : TControl);
21058: Proc LockHeight( aControl : TControl);
21059: Proc LockBounds( aControl : TControl);
21060: Func TruncateFilename(aCanvas:TCanvas;aRect:TRect;aFilename:str;aMargin Int):Str;
21061: Func PointInRect2( const R : TRect; const P : TPoint) :Bool;;
21062: Func PointInRect3( const R : TRect; const X, Y : Int) :Bool;;
21063: Proc VariantToStream2( const V : OLEVariant; Stream : TStream);
21064: Proc StreamToVariant2( Stream : TStream; var V : OLEVariant);
21065: Proc AssignDocument( Browser : TWebBrowser; Text :Str);
21066: Proc LoadStreamToWebBrowser( WebBrowser : TWebBrowser; Stream : TStream);
21067: Proc SaveWebBrowserSourceToStream( Document : IDispatch; Stream : TStream);
21068: Proc GetStylesFromBrowser( aBrowser : TWebBrowser; aStyles : TStrings);
21069: TypeS('TnxBits', 'Int; TypeS('TDayHours', 'Int;
21070:  ClassN(CL.FindClass('TOBJECT,'NixUtilsException;
21071: Func REG_CURRENT_VERSION :Str;
21072: Func RegisteredOwner :Str;
21073: Func RegisteredCompany :Str;
21074: Func GetLocalComputerName2 :Str;
21075: Func GetLocalUserName2 :Str;
21076: Func DeleteToRecycleBin(WindowHandle: HWND; Filename:str; Confirm:Bool): Bool;
21077: Func RemoveBackSlash2( const Dir :Str) :Str;
21078: Func EnsureBackSlash( aPath :Str) :Str;
21079: Func EnsureForwardSlash( aPath :Str) :Str;
21080: Func RemoveLeadingSlash( aPath :Str) :Str;
21081: Func SameDirectories( aDir1, aDir2 : TFilename) :Bool;
21082: Func EnsureExtensionHasLeadingPeriod( aExtension :Str) :Str;
21083: Func RemoveExtension( aFilename :Str) :Str;
21084: Func GetModuleFileNameStr :Str;
21085: Func ModulePath :Str;
21086: Func IniFileName :Str;
21087: Func MakeFilenameInExePath( aFilename : TFilename) : TFilename;
21088: Func IsBitSet( Bits : Int; BitToSet : TnxBits) :Bool;
21089: Func SetBit6( Bits : Int; BitToSet : TnxBits) : Int;
21090: Func UnSetBit( Bits : Int; BitToSet : TnxBits) : Int;
21091: Func FlipBit( Bits : Int; BitToSet : TnxBits) : Int;
21092: Func / FontToOleFont
21093: Func / OleFontToFont
21094: Func GetCachedFileFromURL(strUL:Str; var strLocalFile:Str):Bool;
21095: Func IAddrToHostName(const IP:Str):Str;
21096: Func GetIEHandle(WebBrowser: TWebbrowser; ClassName:Str): HWND;
21097: Func GetTextFromHandle(WinHandle: THandle):Str;
21098: Proc Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
21099: Func FillWebForm(WebBrowser:TWebBrowser;FieldName:Str; Value:Str):Bool;
21100: Proc WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode:Str);
21101: Func NetSend(dest, Source, Msg:Str): Longint; overload;
21102: end;
21103: Proc SIRegister_UWANTUtils(CL: TPSPascalCompiler);
21104: begin
21105: TErrorSeverity,(errwantNone,errwantInfo,errwantHint,errwantWarning,errwantError,errwantFatal );
21106:  Func ErrTag( Sev : TErrorSeverity) :Str;
21107:  Func MakeInt64( LowInt32, HiInt32 :Card) : Int64;
21108:  Func FolderContent( const APath :Str) : INode;
21109:  Func FindFiles2(const AFolder,AMask:Str;Attrs:Int;AList:TStrings;ASearchSubdirs:Bool):Bool;
21110:  Func ErrorSeverityStr( ASeverity : TErrorSeverity) :Str;
21111:  Func IsParameterOptional( AParamDef : INode) :Bool;
21112:  Func ExtractMethodName( const S :Str) :Str;
21113:  Func LineNoOf( ANode : INode) : Int;
21114:  Func ColNoOf( ANode : INode) : Int;
21115:  Func WANTBoolToStr( AValue :Bool) :Str;
21116:  Func WANTStrToBool( const AValue :Str; dEFAULT :Bool) :Bool;
21117:  Func StrDefault( const S, Default :Str) :Str;
21118:  Func PluralNoun( Count : Int; const S :Str) :Str;
21119:  Func IsUnqualifiedName( const AName :Str) :Bool;
21120:  Func IsModuleTag( const ATag :Str) :Bool;
21121:  Func IsCallableTag( const ATag :Str) :Bool;
21122:  Func IsConditionalTag( const ATag :Str) :Bool;
21123:  Func IsMethodTag( const ATag :Str) :Bool;
21124:  Func StripDriveFromPath( const AFileName :Str) :Str;
21125:  Func TotalTime( NumTicks : Int64) :Str;
21126: end;
21127:
21128: Proc SIRegister_IdNNTPServer(CL: TPSPascalCompiler);
21129: begin
21130:   TIdNNTPAuthType', '( atUserPass, atSimple, atGeneric );
21131:   TIdNNTPAuthTypes', 'set of TIdNNTPAuthType;
21132:   SIRegister_TIdNNTPThread(CL);
21133:   TypeS('TIdNNTPOnAuth', Proc ( AThread : TIdNNTPThread; var VAccept :Bool);
21134:   TIdNNTPOnNewGroupsList','Procedure(AThread:TIdNNTPThread;const ADateStamp:TDateTime;const
       ADistributions:str);
```

```
21135:   TIdNNTPOnNewNews', 'Proc ( AThread : TIdNNTPThread; const Nributions :Str);
21136:   TIdNNTPOnIHaveCheck', 'Procedure(AThread:TIdNNTPThread;const AMsgID String; VAccept : Bool);
21137:   TIdNNTPOnArticleByNo', 'Proc ( AThread : TIdNNTPThread; const AMsgNo : Int);
21138:   TIdNNTPOnArticleByID', 'Proc ( AThread : TIdNNTPThread; const AMsgID :Str);
21139:   TIdNNTPOnCheckMsgNo', 'Procedure(AThread:TIdNNTPThread;const AMsgNo:Int; var VMsgID :Str);
21140:   TIdNNTPOnCheckMsgID','Proc (AThread:TIdNNTPThread; const AMsgId:str;var VMsgNo : Int);
21141:   TIdNNTPOnMovePointer', 'Proc (AThread TIdNNTPThread;var AMsgNo:Int;var VMsgID:str);
21142:   TIdNNTPOnPost','Procedure(AThread:TIdNNTPThread;var VPostOk:Boolean;var VErrorText :Str);
21143: TIdNNTPOnSelectGroup','Procedure(AThread:TIdNNTPThread;const AGroup:str;var VMsgCount:Int;var
       VMsgFirst:Int;var
21144:    +VMsgLast : Int; var VGroupExists :Bool);
21145:   TIdNNTPOnCheckListGroup', 'Proc ( AThread : TIdNNTPThread; '
21146:    +'const AGroup :Str; var VCanJoin :Bool; var VFirstArticle : Int);
21147:   TIdNNTPOnXHdr', 'Proc ( AThread : TIdNNTPThread; const AHea'
21148:    +derName:Str; const AMsgFirst: Int; const AMsgLast:Int;const AMsgID:str);
21149:   TIdNNTPOnXOver', 'Proc (AThread:TIdNNTPThread;const AMsgFirst:Int;const AMsgLast : Int);
21150:   TIdNNTPOnXPat','Procedure(AThread:TIdNNTPThread;const AHeaderName:str;const AMsgFirst:Int; const
       AMsgLast:Int;const AMsgID:str;const AHeaderPattern:str);
21151: TIdNNTPOnAuthRequired','Procedure(AThread:TIdNNTPThread;const ACommand,AParams:str;var VRequired:Bool);
21152: TIdNNTPOnListPattern', 'Proc (AThread:TIdNNTPThread;const AGroupPattern:str);
21153:   SIRegister_TIdNNTPServer(CL);
21154: end;
21155:
21156: Proc SIRegister_OverbyteIcsAsn1Utils(CL: TPSPascalCompiler);
21157: begin
21158:  'ASN1UtilsVersion','LongInt.SetInt( 101);
21159:  ConstantN('ASN1_BOOL','LongWord.SetUInt( $01);
21160:  'ASN1_INT','LongWord.SetUInt( $02);
21161:  'ASN1_OCTSTR','LongWord').SetUInt( $04);
21162:  'ASN1_NULL','LongWord').SetUInt( $05);
21163:  'ASN1_OBJID','LongWord').SetUInt( $06);
21164:  'ASN1_ENUM','LongWord').SetUInt( $0a);
21165:  'ASN1_SEQ','LongWord').SetUInt( $30);
21166:  'ASN1_SETOF','LongWord').SetUInt( $31);
21167:  'ASN1_IPADDR','LongWord').SetUInt( $40);
21168:  'ASN1_COUNTER','LongWord').SetUInt( $41);
21169:  'ASN1_GAUGE','LongWord').SetUInt( $42);
21170:  'ASN1_TIMETICKS','LongWord').SetUInt( $43);
21171:  'ASN1_OPAQUE','LongWord').SetUInt( $44);
21172:  Func ASNEncOIDItem2( Value : Int) : Ansistr;
21173:  Func ASNDecOIDItem2( var Start : Int; const Buffer : Ansistr) : Int;
21174:  Func ASNEncLen2( Len : Int) : Ansistr;
21175:  Func ASNDecLen2( var Start : Int; const Buffer : Ansistr) : Int;
21176:  Func ASNEncInt2( Value : Int) : Ansistr;
21177:  Func ASNEncUInt2(Value : Int) : Ansistr;
21178:  Func ASNObject2( const Data : Ansistr; ASNType : Int) : Ansistr;
21179:  Func ASNItem2( var Start:Int; const Buffer:Ansistr; var ValueType:Int):Ansistr;
21180:  Func MibToId2( Mib :Str) : Ansistr;;
21181:  //Func MibToId1( Mib : Ansistr) : Ansistr;;
21182:  Func IdToMib2( const Id : Ansistr) :Str;
21183:  Func IntMibToStr2( const Value : Ansistr) : Ansistr;
21184:  Func ASNdump2( const Value : Ansistr) :Str;
21185: end;
21186: const  SM_MEDIACENTER = 87; //metrics flag not defined in Windows unit
21187:        SM_TABLETPC = 86;
21188:   http://snippets.delphidabbler.com/#
21189:
21190: Proc SIRegister_wmiserv(CL: TPSPascalCompiler);
21191: begin
21192:  ConstantN('EOAC_NONE','LongInt').SetInt( 0);
21193:  'RPC_C_AUTHN_WINNT','LongInt').SetInt( 10);
21194:  'RPC_C_AUTHZ_NONE','LongInt').SetInt( 0);
21195:  'RPC_E_CHANGED_MODE','LongInt').SetInt( - 2147417850);
21196:  Func WMIStart : ISWBemLocator;
21197:  Func WMIConnect(WBemLocator:ISWBemLocator;Server,account,password:str):ISWBemServices;
21198:  Func WMIExecQuery( WBemServices : ISWBemServices; query :Str) : ISWbemObjectSet;
21199: Func WMIRowFindFirst(ObjectSet:ISWbemObjectSet;var ENum:IEnumVariant;var tempobj:OleVariant):bool;
21200:  Func WMIRowFindNext( ENum : IENumVariant; var tempobj : OleVariant) :Bool;
21201:  Func WMIColFindFirst(var propENum:IENumVariant; var tempObj : OleVariant):Bool;
21202:  Func WMIColFindNext( propENum : IENumVariant; var tempobj : OleVariant) :Bool;
21203:  Func WMIGetValue(wbemservices:ISWBemServices; tablename,fieldname:str):Str;
21204:  Func WMIConvValue( tempobj : OleVariant; var keyname :Str):Str;
21205:  --- wmiserv2:
21206:  Func WMIRegConnect(WBemLocator:ISWBemLocator;Server,account,password:str): ISWBemServices;
21207:  Proc WMIGetMethodInfo(srv:ISWbemServices;objname,method:str; var regobject,inparms: ISWBemObject);
21208:  Proc WMISetValue(InParam: ISWBemObject; keyvalue:Str; invalue: OleVariant);
21209: end;
21210:
21211: Proc SIRegister_RegSvrUtils(CL: TPSPascalCompiler);
21212: begin
21213:  SIRegister_ERegistryException(CL);
21214:  Func RegOpenKey2( Key : HKey; const SubKey :Str) : HKey;
21215:  Func RegGetKey( Key : HKey; const SubKey :Str) :Str;
21216:  Func RegCanOpenKey( Key : HKey; const SubKey :Str; var OutKey : HKey) :Bool;
21217:  Func RegKeyExists2( Key : HKey; const SubKey :Str) :Bool;
21218:  Func RegCloseAndNilKey( var Key : HKey) :Bool;
21219:  Func RegQuerySubKeyCount( Key : HKey) : Int;
21220:  Func RegEnumKey2( Key : HKey; Index : Int; var Value :Str) :Bool;
21221:  Func RegQueryKey( Key : HKey; const SubKey :Str; var Value :Str) :Bool;
```

```
21222:  Func RegGetDefaultValue( Key : HKey) :Str;
21223:  Func RegGetValueEx( Key : HKey; const ValName :Str) :Str;
21224:  Proc ErrorFmt( const Ident :Str; const Args : array of const);
21225:  Proc FmtError( const Ident :Str; const Args : array of const);
21226: end;
21227:
21228: Proc SIRegister_osFileUtil(CL: TPSPascalCompiler);  //Linux
21229: begin
21230:  Func OsDOS2UnixFileAttributes( Attr : LongWord) : LongWord;
21231:  Func OsUnix2DosFileAttributes( Attr : LongWord) : LongWord;
21232:  Func OsUnixFileTimeToDateTime( UnixTime : LongInt) : TDateTime;
21233:  Func OsDateTimeToUnixFileTime( DateTime : TDateTime) : LongInt;
21234:  Func OsDosFileTimeToDateTime( DosTime : LongInt) : TDateTime;
21235:  Func OsDateTimeToDosFileTime( Value : TDateTime) : LongInt;
21236:  Func OsFileTimeToLocalFileTime( FileTime : LongInt) : LongInt;
21237:  Func OsLocalFileTimeToFileTime( FileTime : LongInt) : LongInt;
21238: end;
21239:
21240: Proc SIRegister_TWebBrowser(CL: TPSPascalCompiler);
21241: begin
21242:   //with RegClassS(CL,'TOleControl', 'TWebBrowser') do
21243:  with ClassN(CL.FindClass('TOleControl'),'TWebBrowser') do begin
21244:   RegisterMethod(Proc GoBack;
21245:  Proc GoForward;
21246:   Proc GoHome;
21247:   Proc GoSearch;
21248:   Proc Navigate7( const URL : WideString);;
21249:   Proc Navigate8( const URL : WideString; const Flags : OleVariant);;
21250:   Proc Navigate9(const URL:WideString;const Flags:OleVariant;const TargetFrameName: OleVariant);;
21251:    Proc Navigate10(const URL:WideString;const Flags:OleVariant;const TargetFrameName:OleVariant;var
      PostData:OleVariant);
21252:   Proc Navigate11(const URL:WideString;const Flags:OleVariant;const TargetFrameName:OleVariant;var
      PostData:OleVariant;const Headers:OleVariant);
21253:   Proc Refresh;
21254:   Proc Refresh212;;
21255:   Proc Refresh213( var Level : OleVariant);;
21256:   Proc Stop;
21257:   Proc Quit;
21258:   Proc ClientToWindow( var pcx : SYSINT; var pcy : SYSINT);
21259:   Proc PutProperty( const Property_ : WideString; vtValue : OleVariant);
21260:   Func GetProperty( const Property_ : WideString) : OleVariant;
21261:   Proc Navigate214( var URL : OleVariant);;
21262:   Proc Navigate215( var URL : OleVariant; var Flags : OleVariant);;
21263:   Proc Navigate216(var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVar);
21264:   Proc Navigate217(var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVariant;var PostData:
      OleVariant);
21265:   Proc Navigate218(var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVar;var
      PostData:OleVariant;var Headers:OleVariant);
21266:  Func QueryStatusWB( cmdID : OLECMDID) : OLECMDF;
21267:  Proc ExecWB19( cmdID : OLECMDID; cmdexecopt : OLECMDEXECOPT);;
21268:  Proc ExecWB20(cmdID:OLECMDID;cmdexecopt: OLECMDEXECOPT; var pvaIn : OleVariant);;
21269:  Proc ExecWB21(cmdID:OLECMDID;cmdexecopt:OLECMDEXECOPT;var pvaIn:OleVariant;var pvaOut:OleVariant);
21270:  Proc ShowBrowserBar22( var pvaClsid : OleVariant);;
21271:  Proc ShowBrowserBar23( var pvaClsid : OleVariant; var pvarShow : OleVariant);;
21272:  Proc ShowBrowserBar24(var pvaClsid:OleVariant;var pvarShow:OleVariant;var pvarSize: OleVariant);;
21273:  ControlInterface', 'IWebBrowser2', iptr);
21274:  RegisterProperty(DefaultInterface, IWebBrowser2', iptr);
21275:  Application', 'IDispatch', iptr);
21276:  Parent', 'IDispatch', iptr);
21277:  Container', 'IDispatch', iptr);
21278:  Document', 'IDispatch', iptr);
21279:  TopLevelContainer', 'WordBool', iptr);
21280:  type_', 'WideString', iptr);
21281:  LocationName', 'WideString', iptr);
21282:  LocationURL', 'WideString', iptr);
21283:  Busy', 'WordBool', iptr);
21284:  Name', 'WideString', iptr);
21285:  FullName', 'WideString', iptr);
21286:  Path', 'WideString', iptr);
21287:  Visible', 'WordBool', iptrw);
21288:  StatusBar', 'WordBool', iptrw);
21289:  StatusText', 'WideString', iptrw);
21290:  ToolBar', 'Int', iptrw);
21291:  MenuBar', 'WordBool', iptrw);
21292:  FullScreen', 'WordBool', iptrw);
21293:  Offline', 'WordBool', iptrw);
21294:  Silent', 'WordBool', iptrw);
21295:  RegisterAsBrowser', 'WordBool', iptrw);
21296:  RegisterAsDropTarget', 'WordBool', iptrw);
21297:  TheaterMode', 'WordBool', iptrw);
21298:  AddressBar', 'WordBool', iptrw);
21299:  Resizable', 'WordBool', iptrw);
21300:  OnStatusTextChange', 'TWebBrowserStatusTextChange', iptrw);
21301:  OnProgressChange', 'TWebBrowserProgressChange', iptrw);
21302:  OnCommandStateChange', 'TWebBrowserCommandStateChange', iptrw);
21303:  OnDownloadBegin', 'TNotifyEvent', iptrw);
21304:  OnDownloadComplete', 'TNotifyEvent', iptrw);
21305:  OnTitleChange', 'TWebBrowserTitleChange', iptrw);
21306:  OnPropertyChange', 'TWebBrowserPropertyChange', iptrw);
```

```
21307:  OnBeforeNavigate2', 'TWebBrowserBeforeNavigate2', iptrw);
21308:  OnNewWindow2', 'TWebBrowserNewWindow2', iptrw);
21309:  OnNavigateComplete2', 'TWebBrowserNavigateComplete2', iptrw);
21310:  OnDocumentComplete', 'TWebBrowserDocumentComplete', iptrw);
21311:  OnQuit', 'TNotifyEvent', iptrw);
21312:  OnVisible', 'TWebBrowserOnVisible', iptrw);
21313:  OnToolBar', 'TWebBrowserOnToolBar', iptrw);
21314:  OnMenuBar', 'TWebBrowserOnMenuBar', iptrw);
21315:  OnStatusBar', 'TWebBrowserOnStatusBar', iptrw);
21316:  OnFullScreen', 'TWebBrowserOnFullScreen', iptrw);
21317:  OnTheaterMode', 'TWebBrowserOnTheaterMode', iptrw);
21318:  OnWindowSetResizable', 'TWebBrowserWindowSetResizable', iptrw);
21319:  OnWindowSetLeft', 'TWebBrowserWindowSetLeft', iptrw);
21320:  OnWindowSetTop', 'TWebBrowserWindowSetTop', iptrw);
21321:  OnWindowSetWidth', 'TWebBrowserWindowSetWidth', iptrw);
21322:  OnWindowSetHeight', 'TWebBrowserWindowSetHeight', iptrw);
21323:  OnWindowClosing', 'TWebBrowserWindowClosing', iptrw);
21324:  OnClientToHostWindow', 'TWebBrowserClientToHostWindow', iptrw);
21325:  OnSetSecureLockIcon', 'TWebBrowserSetSecureLockIcon', iptrw);
21326:  OnFileDownload', 'TWebBrowserFileDownload', iptrw);
21327:  OnNavigateError', 'TWebBrowserNavigateError', iptrw);
21328:  OnPrintTemplateInstantiation', 'TWebBrowserPrintTemplateInstantiation', iptrw);
21329:  OnPrintTemplateTeardown', 'TWebBrowserPrintTemplateTeardown', iptrw);
21330:  OnUpdatePageStatus', 'TWebBrowserUpdatePageStatus', iptrw);
21331:  OnPrivacyImpactedStateChange', 'TWebBrowserPrivacyImpactedStateChange', iptrw);
21332:  end;
21333: end;
21334:
21335: Proc SIRegister_xutils(CL: TPSPascalCompiler);
21336: begin
21337:   //TypeS('PshortString', '^ShortString // will not work;
21338:  'EXIT_DOSERROR','LongInt').SetInt( 2);
21339:  'EXIT_ERROR','LongInt').SetInt( 1);
21340:  'adCmdTxt','LongInt').SetInt($00000001);
21341:  'adExecNoRecords','LongInt').SetInt($00000080);
21342:  {** Byte order mark: UTF-8 encoding signature }
21343:  ('BOM_UTF8','String').SetString( #$EF#$BB#$BF);
21344:  {** Byte order mark: UTF-8 encoding signature }
21345:   BOM_UTF8      = #$EF#$BB#$BF;
21346:   {** Byte order mark: UCS-4 big endian encoding signature }
21347:   BOM_UTF32_BE = #00#00#$FE#$FF;
21348:   {** Byte order mark: UCS-4 little endian encoding signature }
21349:   BOM_UTF32_LE = #$FF#$FE#00#00;
21350:   BOM_UTF16_BE = #$FE#$FF;
21351:   BOM_UTF16_LE = #$FF#$FE;
21352:   //const adCmdTxt = $00000001; //adExecNoRecords = $00000080;
21353:  Func AnsiFileExists( const FName :Str) :Bool;
21354:  Func AnsiDirectoryExists( DName :Str) :Bool;
21355:  Proc SwapLong2( var x : longword);
21356:  Proc SwapWord2( var x : word);
21357:  Func UpString( s :Str) :Str;
21358:  Func LowString( s :Str) :Str;
21359:  Func AddDoubleQuotes( s :Str) :Str;
21360:  Func RemoveDoubleQuotes( s :Str) :Str;
21361:  Proc StreamErrorProcedure( var S : TStream);
21362:  Func StrToken( var Text :Str; Delimiter : Char; UseQuotes:boolean):Str;
21363:  Func StrGetNextLine( var Text :Str) :Str;
21364:  Func TrimLeftx( const S :Str) :Str;
21365:  Func TrimRightx( const S :Str) :Str;
21366:  Func hexstr( val : longint; cnt : byte) :Str;
21367:  Func decstr( val : longint; cnt : byte) :Str;
21368:  Func decstrunsigned( l : longword; cnt : byte) :Str;
21369:  Func boolstr( val :Bool; cnt : byte) :Str;
21370:  Func CompareByte( buf1, buf2 : pchar; len : longint) : Int;
21371:  Func Trimx( const S :Str) :Str;
21372:  Func Printf2( const s :Str; var Buf:Str; size : word) :Str;
21373:  Func Printfx( const s :Str; var Buf:Str; size : word) :Str;
21374:  Func FillTo( s :Str; tolength : Int) :Str;
21375:  Func stringdup( const s :Str) :Str;
21376:  //Proc stringdispose( var p : pShortstring);
21377:  Func EscapeToPascal( const s :Str; var code : Int) :Str;
21378:  Func ValDecimal( const S :Str; var code : Int) : longint;
21379:  Func ValUnsignedDecimal( const S :Str; var code : Int) : longword;
21380:  Func ValOctal( const S :Str; var code : Int) : longint;
21381:  Func ValBinary( const S :Str; var code : Int) : longint;
21382:  Func ValHexadecimal( const S :Str; var code : Int) : longint;
21383:  Func CleanString( const s :Str) :Str;
21384:  Func ChangeFileExt2( const FileName, Extension :Str) :Str;
21385:  Func fillwithzero( s :Str; newlength : Int) :Str;
21386:  Func removenulls( const s :Str) :Str;
21387:  //ConstantN('WhiteSpace','Char').SetString( ' ' or  #10 or  #13 or  #9);
21388: end;
21389:
21390: Proc SIRegister_ietf(CL: TPSPascalCompiler);
21391: begin
21392:  Func mime_isvalidcontenttype( const s : shortstring) :Bool;
21393:  Func langtag_isvalid( const s :Str) :Bool;
21394:  Func langtag_split( const s :Str; var primary, sub :Str) :Bool;
21395:  URI_START_DELIMITER_CHAR','String').SetString( '<;
```

```
21396:  URI_END_DELIMITER_CHAR','String').SetString( '>;
21397:  URI_SCHEME_NAME_EMAIL','String').SetString( 'mailto;
21398:  URI_SCHEME_SEPARATOR','String').SetString( ':;
21399:  Func uri_split( url :Str; var scheme, authority, path, query :Str) :Bool;
21400:  Func urn_isvalid( s : shortstring) :Bool;
21401:  Func urn_isvalidnid( nid :Str) :Bool;
21402:  Func urn_split( urn :Str; var urnidstr, nidstr, nssstr :Str) :Bool;
21403:  Func urn_pathsplit( path :Str; var namespace, nss :Str) :Bool;
21404:  Func http_pathsplit( path :Str; var directory, name :Str) :Bool;
21405:  Func file_pathsplit( path :Str; var directory, name :Str) :Bool;
21406: end;
21407:
21408: Proc SIRegister_dateutilreal(CL: TPSPascalCompiler);
21409: begin
21410:   TypeS('TDatetimeReal', 'real;
21411:   TypeS(TDateInfo', 'record DateTime : TDateTime; UTC :Bool; end;
21412:   //TypeS('float', 'real;
21413:    TypeS('big_Int_t', 'int64;
21414:  'DayMondayR','LongInt').SetInt( 1);
21415:  ConstantN('DayTuesdayR','LongInt').SetInt( 2);
21416:  'DayWednesdayR','LongInt').SetInt( 3);
21417:  'DayThursdayR','LongInt').SetInt( 4);
21418:  'DayFridayR','LongInt').SetInt( 5);
21419:  'DaySaturdayR','LongInt').SetInt( 6);
21420:  'DaySundayR','LongInt').SetInt( 7);
21421:  Func CurrentYearreal : word;
21422:  Func Datereal : TDatetimeReal;
21423:  Func DateOfreal( const AValue : TDatetimeReal) : TDatetimeReal;
21424:  Func DateTimeToStrreal( DateTime : TDatetimeReal) :Str;
21425:  Func DateToStrreal( date : TDatetimeReal) :Str;
21426:  Func DayOfreal( const AValue : TDatetimeReal) : Word;
21427:  Func DaysBetweenreal( const ANow, AThen : TDatetimeReal) : Int;
21428:  Proc DecodeDatereal( Date : TDatetimeReal; var Year, Month, Day : Word);
21429:  Proc DecodeDateTimereal(const AValue:TDatetimeReal;var Year,Month,Day,Hour,Minute,Second,
       MilliSecond:Word);
21430:  Proc DecodeTimereal( Time : TDatetimeReal; var Hour, Min, Sec, MSec : Word);
21431:  Func HourOfreal( const AValue : TDatetimeReal) : Word;
21432:  Func IncDayreal(const AValue:TDatetimeReal;const ANumberOfDays:Int): TDatetimeReal;
21433:  Func IncHourreal(const AValue:TDatetimeReal;const ANumberOfHours:longint):TDatetimeReal;
21434:  Func IncMilliSecondreal(const AValue:TDatetimeReal; const ANumberOfMilliSeconds: big_Int_t):TDatetimeReal;
21435:  Func IncMinutereal(const AValue: TDatetimeReal; const ANumberOfMinutes: big_Int_t) : TDatetimeReal;
21436:  Func IncSecondreal(const AValue: TDatetimeReal; const ANumberOfSeconds: big_Int_t) : TDatetimeReal;
21437:  Func IncWeekreal(const AValue:TDatetimeReal;const ANumberOfWeeks:Int): TDatetimeReal;
21438:  Func IsPMreal( const AValue : TDatetimeReal) :Bool;
21439:  Func IsValidDatereal( const AYear, AMonth, ADay : Word) :Bool;
21440: Func IsValidDateTimereal(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSec:Word):Bool;
21441:  Func IsValidTimereal( const AHour, AMinute, ASecond, AMilliSecond : Word) :Bool;
21442:  Func MinuteOfreal( const AValue : TDatetimeReal) : Word;
21443:  Func MonthOfreal( const AValue : TDatetimeReal) : Word;
21444:  Func Nowreal : TDatetimeReal;
21445:  Func SameDatereal( const A, B : TDatetimeReal) :Bool;
21446:  Func SameDateTimereal( const A, B : TDatetimeReal) :Bool;
21447:  Func SameTimereal( const A, B : TDatetimeReal) :Bool;
21448:  Func SecondOfreal( const AValue : TDatetimeReal) : Word;
21449:  Func Timereal : TDatetimeReal;
21450:  Func GetTimereal : TDatetimeReal;
21451:  Func TimeOfreal( const AValue : TDatetimeReal) : TDatetimeReal;
21452:  Func TimeToStrreal( Time : TDatetimeReal) :Str;
21453:  Func Todayreal : TDatetimeReal;
21454:  Func TryEncodeDatereal( Year, Month, Day : Word; var Date : TDatetimeReal) :Bool;
21455:  Func TryEncodeTimereal(Hour,Min,Sec MSec: Word; var Time : TDatetimeReal) :Bool;
21456:  Func TryEncodeDateTimereal(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond:Word;var
       AValue:TDatetimeReal):Boolean;
21457:  Func TryStrToDatereal( const S :Str; var Value : TDatetimeReal) :Bool;
21458:  Func TryStrToDateTimereal( const S :Str; var Value : TDatetimeReal) :Bool;
21459:  Func TryStrToTimereal( const S :Str; var Value : TDatetimeReal) :Bool;
21460:  Func YearOfreal( const AValue : TDatetimeReal) : Word;
21461:  Func parsetimeISO(timestr:str;var hourval,minval,secval:word;var offsethourval,offsetminval:Int;var
       UTC:bool):bool;
21462:  Func parsedateISO( datestr :Str; var yearval, monthval, dayval : word) :Bool;
21463:  Func datetojd( year, month, day, hour, minute, second, millisecond : word) : float;
21464:  Proc jdtodate(jday:float; var year, month, day, hour, minute, second, msec : word);
21465:  Func converttoisotime( timestr :Str) :Str;
21466:  Func AdobeDateToISODate( s :Str) :Str;
21467:  Func RFC822ToISODateTime( s :Str) :Str;
21468:  Proc getdatedos( var year, month, mday, wday : word);
21469:  Proc gettimedos( var hour, minute, second, sec100 : word);
21470: end;
21471:
21472: Proc SIRegister_dateext4(CL: TPSPascalCompiler);
21473: begin
21474:  TypeS('tfiletime2', 'record LowDateTime: longword; HighDateTime: longword; end;
21475:  // TypeS('big_Int_t', 'int64;
21476:  Func DosToWinTime( DTime : longint; var Wtime : TFileTime) : longbool;
21477:  Func WinToDosTime( const Wtime : TFileTime; var DTime : longint) : longbool;
21478:  Func TryStrToDateTimeExt(const S:str;var Value:TDateTime;var UTC:boolean):Bool);
21479:  Func TryEncodeDateAndTimeToStr(const Year,Month,Day,Hour,Minute,Second,MilliSecond:word;UTC:boolean;var
       AValue:str):boolean;
21480:  Func DateTimeToStrExt( DateTime : TDateTime; utc :Bool) :Str;
```

```
21481:  Proc GetCurrentDate( var Year, Month, Day, DayOfWeek : Int);
21482:  Proc GetCurrentTime2( var Hour, Minute, Second, Sec100 : Int);
21483:  Func TryUNIXToDateTimeExt(unixtime:big_Int_t;var DateTime:TDateTime;var UTC:bool): bool;
21484:  Func TryFileTimeToDateTimeExt( ftime : tfiletime; var DateTime : TDateTime; var UTC :Bool): bool;
21485:  Proc JulianToGregorian(JulianDN:big_Int_t;var Year,Month,Day:Word);;
21486: end;
21487:
21488:
21489: Proc SIRegister_locale(CL: TPSPascalCompiler);
21490: begin
21491:  Func GetISODateString( Year, Month, Day : Word) :Str;
21492:  Func GetISODateStringBasic( Year, Month, Day : Word) :Str;
21493:  Func IsValidISODateString( datestr : shortstring; strict :Bool) :Bool;
21494:  Func IsValidISODateStringExt(datestr:shortstring;strict:bool;var Year,Month,Day:word):bool;
21495:  Func IsValidISOTimeString( timestr : shortstring; strict :Bool) :Bool;
21496:  Func IsValidISOTimeStringExt(timestr:shortstring;strict:boolean;var hour,min,sec:word;var offhour,offmin:
     smallint):bool;
21497:  Func IsValidISODateTimeString( str : shortstring; strict :Bool) :Bool;
21498:  Func GetISOTimeString( Hour, Minute, Second : Word; UTC :Bool) : shortstring;
21499:  Func GetISOTimeStringBasic(Hour,Minute,Second: Word; UTC :Bool) : shortstring;
21500:  Func GetISODateTimeString(Year,Month,Day,Hour,Minute,Second:Word;UTC:Bool):shortstring;
21501:  Proc UNIXToDateTime2(epoch:big_Int_t; var year,month,day,hour,minute,second: Word);
21502:  Func GetCharEncoding( alias :Str; var _name :Str) : Int;
21503:  Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
21504:  Func MicrosoftLangageCodeToISOCode( langcode : Int) :Str;
21505:  CHAR_ENCODING_UTF8','LongInt').SetInt( 0);
21506:  ConstantN(CHAR_ENCODING_UNKNOWN',LongInt').SetInt( - 1);
21507:  'CHAR_ENCODING_UTF32BE','LongInt').SetInt( 1);
21508:  'CHAR_ENCODING_UTF32LE','LongInt').SetInt( 2);
21509:  'CHAR_ENCODING_UTF16LE','LongInt').SetInt( 3);
21510:  'CHAR_ENCODING_UTF16BE','LongInt').SetInt( 4);
21511:  'CHAR_ENCODING_BYTE','LongInt').SetInt( 5);
21512:  'CHAR_ENCODING_UTF16','LongInt').SetInt( 6);
21513:  'CHAR_ENCODING_UTF32','LongInt').SetInt( 7);
21514: end;
21515:
21516: Proc SIRegister_Strings(CL: TPSPascalCompiler);
21517: begin
21518:  DelphiFunction('Func StrLenPchar( Str : PChar) : longint;
21519:  Func StrEndPchar( Str : PChar) : PChar;
21520:  Func StrMovePchar( Dest, Source : Pchar; l : Longint) : pchar;
21521:  Func StrCopyPchar( Dest, Source : PChar) : PChar;
21522:  Func StrECopyPchar( Dest, Source : PChar) : PChar;
21523:  Func StrLCopyPchar( Dest, Source : PChar; MaxLen : Longint) : PChar;
21524:  Func StrPCopyPchar( Dest : PChar; Source :Str) : PChar;
21525:  Func StrCatPchar( Dest, Source : PChar) : PChar;
21526:  Func strlcatPchar( dest, source : pchar; l : Longint) : pchar;
21527:  Func StrCompPchar( Str1, Str2 : PChar) : Int;
21528:  Func StrICompPchar( Str1, Str2 : PChar) : Int;
21529:  Func StrLCompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Int;
21530:  Func StrLICompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Int;
21531:  Func StrScanPchar( Str : PChar; Ch : Char) : PChar;
21532:  Func StrRScanPchar( Str : PChar; Ch : Char) : PChar;
21533:  Func StrPosPchar( Str1, Str2 : PChar) : PChar;
21534:  Func StrUpperPchar( Str : PChar) : PChar;
21535:  Func StrLowerPchar( Str : PChar) : PChar;
21536:  Func StrPasPchar( Str : PChar) :Str;
21537:  Func StrNewPchar( P : PChar) : PChar;
21538:  Proc StrDisposePchar( P : PChar);
21539: end;
21540:
21541: Proc SIRegister_crc_checks(CL: TPSPascalCompiler);
21542: begin
21543:  Func UpdateCrc32_2( InitCrc : longword; b : byte) : longword;
21544:  Func UpdateCrc16_2( InitCrc : word; b : byte) : word;
21545:  Func UpdateAdler32( InitAdler : longword; b : byte) : longword;
21546:  Func UpdateFletcher8( InitFletcher : word; b : byte) : word;
21547:  Func UpdateCRC( InitCrc : word; b : byte) : word;
21548: end;
21549:
21550: Proc SIRegister_extdos(CL: TPSPascalCompiler);
21551: begin
21552:  TypeS('utf8char', 'char;
21553:   TypeS('putf8char', 'pchar;
21554:   TypeS('utf16char', 'word;
21555:   TypeS('ucs4char', 'longword;
21556:   //TypeS('pucs4char', '^ucs4char // will not work;
21557:   TypeS('ucs2char', 'word;
21558:   TypeS('pucs2char', 'ucs2char; // will not work;
21559:   //TypeS('pucs2char', '^ucs2char // will not work;
21560:   TypeS('utf8string', 'string;
21561:  // TypeS('putf8shortstring', '^shortstring // will not work;
21562:   TypeS('_USER_INFO_2', record usri2_name : pucs2char; usri2_password : '
21563:    +pucs2char; usri2_password_age : DWORD; usri2_priv : DWORD; usri2_home_dir '
21564:    +: pucs2char; usri2_comment : pucs2char; usri2_flags : DWORD; usri2_script_'
21565:    +path : pucs2char; usri2_auth_flags : DWORD; usri2_full_name : pucs2char; u'
21566:    +sri2_usr_comment : pucs2char; usri2_parms : pucs2char; usri2_workstations '
21567:    +: pucs2char; usri2_last_logon : DWORD; usri2_last_logoff : DWORD; usri2_ac'
21568:    +ct_expires : DWORD; usri2_max_storage : DWORD; usri2_units_per_week : DWOR'
```

```
21569:    +D; usri2_logon_hours : pchar; usri2_bad_pw_count : DWORD; usri2_num_logons'
21570:    +:DWORD; usri2_logon_server:pucs2char;usri2_country_code:DWORD;usri2_code_page:DWORD; end;
21571:   //TypeS('P_USER_INFO_2', '^_USER_INFO_2 // will not work;
21572:   TypeS('SE_OBJECT_TYPE', '( SE_UNKNOWN_OBJECT_TYPE, SE_FILE_OBJECT, SE_S'
21573:    +'ERVICE, SE_PRINTER, SE_REGISTRY_KEY, SE_LMSHARE, SE_KERNEL_OBJECT, SE_WIND'
21574:    +'OW_OBJECT, SE_DS_OBJECT, SE_DS_OBJECT_ALL, SE_PROVIDER_DEFINED_OBJECT, SE_'
21575:    +'WMIGUID_OBJECT );
21576:   //TypeS('PPSID', '^PSID // will not work;
21577:   TypeS('ASSOCF', 'DWORD;
21578:   TypeS('ASSOCSTR', '( ASSOCSTR_NONE, ASSOCSTR_COMMAND, ASSOCSTR_EXECUTAB'
21579:    +'LE, ASSOCSTR_FRIENDLYDOCNAME, ASSOCSTR_FRIENDLYAPPNAME, ASSOCSTR_NOOPEN, A'
21580:    +'SSOCSTR_SHELLNEWVALUE,ASSOCSTR_DDECOMMAND,ASSOCSTR_DDEIFEXEC,ASSOCSTR_DDEAPPLICATION,
     ASSOCSTR_DDETOPIC);
21581:   TypeS('TCHAR', 'ucs2char;
21582:   {TypeS('TFileTime', 'FILETIME;
21583:   TypeS('TWin32FindDataW', '_WIN32_FIND_DATAW;
21584:   TypeS('TCHAR', 'ucs2char;
21585:   TypeS('TFileTime', 'FILETIME;
21586:   TypeS('TWin32FindDataW', '_WIN32_FIND_DATAW;   }
21587:   //dev: array[0..127] of char;
21588:   {** Unique file serial number, this may change from one boot to the next.}
21589:   //ino: array[0..127] of char;
21590:
21591: TypeS('tresourceattribute', '( attr_any, attr_readonly, attr_hidden, at'
21592:    +'tr_system, attr_archive, attr_link, attr_directory, attr_temporary, attr_e'
21593:    +'ncrypted,attr_no_indexing,attr_device,attr_extended,attr_compressed,attr_offline, attr_sparse);
21594:   TFileAssociation', 'record appname : utf8string; exename : utf8string; end;
21595:   tresourceattributes', 'set of tresourceattribute;
21596:   TypeS('TFileStats', record name : utf8string; size : big_Int_t; ow'
21597:    +ner : utf8string; ctime : TDateTime; mtime : TDateTime; atime : TDateTime;'
21598:    +nlink : Int; attributes : tresourceattributes; association : tfileass'
21599:    +ociation; streamcount:Int;accesses:Int; utc :Bool; dev: array[0..127] of char; '
21600:    +'ino: array[0..127] of char; comment : utf8string; dirstr : utf8string; end;
21601:  TSearchRecExt', 'record Stats : TFileStats; FindHandle : THandle'
21602:    +; W32FindData  TWin32FindData;IncludeAttr: longint; SearchAttr:TResourceAttributes; end;
21603:   Func GetFileOwner( fname : putf8char) : utf8string;
21604:   GetFileATime( fname : putf8char; var atime : TDateTime) : Int;
21605:   GetFileMTime( fname : putf8char; var mtime : TDateTime) : Int;
21606:   GetFileCTime( fname : putf8char; var ctime : TDateTime) : Int;
21607:   GetFilesizeExt( fname : putf8char) : big_Int_t;
21608:   GetFileAttributesExt( fname : putf8char) : tresourceattributes;
21609:   GetFilestats( fname : putf8char; var stats : TFileStats) : Int;
21610:   DirectoryExistsExt( DName : utf8string) :Bool;
21611:   FileExistsExt( const FName : utf8string) :Bool;
21612:   GetCurrentDirectoryExt( var DirStr : utf8string) :Bool;
21613:   SetCurrentDirectoryExt( const DirStr : utf8string) :Bool;
21614:   SetFileATime( fname : putf8char; newatime : tdatetime) : Int;
21615:   SetFileMTime( fname : putf8char; newmtime : tdatetime) : Int;
21616:   SetFileCTime( fname : putf8char; newctime : tdatetime) : Int;
21617:   FindFirstEx(path:putf8char; attr:tresourceattributes;var SearchRec:TSearchRecExt):int;
21618:   FindNextEx( var SearchRec : TSearchRecExt) : Int;
21619:   Proc FindCloseEx( var SearchRec : TSearchRecExt);
21620:   GetUserFullName( account : utf8string) : utf8string;
21621:   GetLoginConfigDirectory : utf8string;
21622:   GetGlobalConfigDirectory : utf8string;
21623:   GetLoginHomeDirectory : utf8string;
21624:  // ucs4strnewstr(str:Str; srctype:Str): pucs4char;;
21625:  // ucs4strnewucs4(src: pucs4char): pucs4char;;
21626: end;
21627:
21628: Proc SIRegister_xBase(CL: TPSPascalCompiler);
21629: begin
21630:  _INADDR_ANY','LongWord').SetUInt($00000000);
21631:  //CL.AddConstantN('DCX_EXCLUDERGN','LongWord').SetUInt( $40);
21632:  INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
21633:  CL.AddConstantN('INVALID_FILE_SIZE','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21634:  INVALID_FILE_ATTRIBUTES','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21635:  INVALID_FILE_TIME','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21636:  INVALID_REGISTRY_KEY','LongWord').SetInt( DWORD ( - 1 ));
21637:  INVALID_VALUE','LongWord').SetInt( DWORD ( - 1 ));
21638:  SleepQuant','LongInt').SetInt( 1 * 60 * 1000);
21639:  MMaxChars','LongInt').SetInt( 250);
21640:   CL.AddTypeS('TBase64Table', '( bsBase64, bsUUE, bsXXE );
21641:   CL.AddTypeS('TWOHandleArray', 'array[0..64 - 1] of THandle;;
21642:  CL.AddTypeS('TWOHandleArray2', 'array[0..MAXIMUM_WAIT_OBJECTS - 1] of THandle;;
21643:  //type TWOHandleArray = array[0..MAXIMUM_WAIT_OBJECTS - 1] of THandle;
21644:   // MAXIMUM_WAIT_OBJECTS = 64;
21645:  SIRegister_TMimeCoder(CL);
21646:   CL.AddTypeS('TSocketOption', '(soBroadcast, soDebug, soDontLinger, soDontRou'
21647:    +'te, soKeepAlive, soOOBInLine, soReuseAddr, soNoDelay, soBlocking,soAcceptConn );
21648:   TSocketOptions', 'set of TSocketOption;
21649:   //TSocketClass', 'class of TSocket;
21650:   SIRegister_TSocketX(CL);
21651:   TObjProcX', 'Procedure;
21652:   TForEachProc', 'Proc ( P : _Pointer);
21653:   //PFileInfo', '^TFileInfo // will not work;
21654:   TFileInfo', 'record Attr : DWORD; Size : DWORD; Time : DWORD; end;
21655:   TuFindData', 'record Info : TFileInfo; FName :Str; end;
21656:   TCreateFileMode', '( cRead, cWrite, cFlag, cEnsureNew, cTruncate'
```

```
21657:    +, cEXisting, cShareAllowWrite, cShareDenyRead,cOverlapped,cRandomAccess,cSequentialScan,cDeleteOnClose
      );
21658:   TCreateFileModeSet', 'set of TCreateFileMode;
21659:   //PCharSet', '^TCharSet // will not work;
21660:   TCharSetX', 'set of Char;
21661:   //PCharArray', '^TCharArray // will not work;
21662:   //PByteArray', '^TByteArray // will not work;
21663:   //PIntArray', '^TIntArray // will not work;
21664:   //PDwordArray', '^TDwordArray // will not work;
21665:   //PvIntArr', '^TvIntArr // will not work;
21666:   //TvIntArr', 'record Arr : PIntArray; Cnt : Integer; end;
21667:   //PBoolean', '^Boolean // will not work;
21668:   //PItemList', '^TItemList // will not work;
21669:   TThreadMethodX', 'Procedure;
21670:   TThreadPriorityXX,(tpIdle tpLowest,tpLower,tpNormal,tpHigher,tpHighest,tpTimeCritical);
21671:   SIRegister_TThreadX(CL);
21672:   CL.AddClassN(CL.FindClass('TOBJECT'),'TAdvObject;
21673:   SIRegister_TAdvObject(CL);
21674:   SIRegister_TAdvCpObject(CL);
21675:   //TAdvClass', 'class of TAdvObject;
21676:   //TCollClass', 'class of TColl;
21677:   SIRegister_TColl(CL);
21678:   SIRegister_TSortedColl(CL);
21679:   SIRegister_TStringColl(CL);
21680:  Func AddRightSpaces( const S :Str; NumSpaces : Integer) :Str;
21681:  Proc AddStr( var S :Str; C : char);
21682:  Proc Add_Str( var S : ShortString; C : char);
21683:  Func CompareStrX( const S1, S2 :Str) : Integer;
21684:  Func CopyLeft( const S :Str; I : Integer) :Str;
21685:  Proc DelDoubles( const St :Str; var Source :Str);
21686:  Proc DelFC( var s :Str);
21687:  Proc DelLC( var s :Str);
21688:  Func DelLeft( const S :Str) :Str;
21689:  Func DelRight( const S :Str) :Str;
21690:  Func DelSpaces( const s :Str) :Str;
21691:  Proc DeleteLeft( var S :Str; I : Integer);
21692:  Func DigitsOnly( const AStr :Str) :Bool;
21693:  Proc DisposeStr( P :Str);
21694:  Func ExpandFileNameX( const FileName :Str) :Str;
21695:  Func ExtractFilePathX( const FileName :Str) :Str;
21696:  Func ExtractDir( const S :Str) :Str;
21697:  Func ExtractFileRootX( const FileName :Str) :Str;
21698:  Func ExtractFileExtX( const FileName :Str) :Str;
21699:  Func ExtractFileNameX( const FileName :Str) :Str;
21700:  Func ExtractFileDriveX( const FileName :Str) :Str;
21701:  Func ExtractFileDirX( const FileName :Str) :Str;
21702:  Proc FSplit( const FName :Str; var Path, Name, Ext :Str);
21703:  Proc FillCharSetX( const AStr :Str; var CharSet : TCharSet);
21704:  Func GetWrdStrictUC( var s, w :Str) :Bool;
21705:  Func GetWrdStrict( var s, w :Str) :Bool;
21706:  Func GetWrdD( var s, w :Str) :Bool;
21707:  Func GetWrdA( var s, w :Str) :Bool;
21708:  Func GetWrd( var s, w :Str; c : char) :Bool;
21709:  Func Hex2X( a : Byte) :Str;
21710:  Func Hex4X( a : Word) :Str;
21711:  Func Hex8X( a : DWORD) :Str;
21712:  Func Int2Hex( a : Integer) :Str;
21713:  Func Int2StrX( L : Integer) :Str;
21714:  Func ItoS( I : Integer) :Str;
21715:  Func ItoSz( I, Width : Integer) :Str;
21716:  Func LastDelimiterX( const Delimiters, S :Str) : Integer;
21717:  Func LowerCaseX( const S :Str) :Str;
21718:  Func MakeFullDir( const D, S :Str) :Str;
21719:  Func MakeNormName( const Path, Name :Str) :Str;
21720:  Func MonthE( m : Integer) :Str;
21721:  Func NewStr( const S :Str) :Str;
21722:  Func ReplaceX( const Pattern, ReplaceString:Str; var S:Str):Bool;
21723:  Func StoI( const S :Str) : Integer;
21724:  Func StrEnds( const S1, S2 :Str) :Bool;
21725:  Func StrRightX( const S :Str; Num : Integer) :Str;
21726:  Func UpperCaseX( const S :Str) :Str;
21727:  Func WipeChars( const AStr, AWipeChars :Str) :Str;
21728:  Func _Val( const S :Str; var V : Integer) :Bool;
21729:  Func ProcessQuotes( var s :Str) :Bool;
21730:  Func UnpackPchars( var s :Str) :Bool;
21731:  Func UnpackUchars( var s :Str) :Bool;
21732:  Func __alpha( c : char) :Bool;
21733:  Func __ctl( c : char) :Bool;
21734:  Func __digit( c : char) :Bool;
21735:  Func __extra( c : char) :Bool;
21736:  Func __national( c : char) :Bool;
21737:  Func __pchar( c : char) :Bool;
21738:  Func __reserved( c : char) :Bool;
21739:  Func __safe( c : char) :Bool;
21740:  Func __uchar( c : char) :Bool;
21741:  Func __unsafe( c : char) :Bool;
21742:  Func Buf2Str( const Buffer:Str) :Str;
21743:  //Proc Clear( var Buf, Count : Integer);
21744:  Func CompareMem( P1, P2 : _Pointer; Length : Integer) :Bool;
```

```
21745:  Proc FreeObjectX( var O: TObject);
21746:  Proc LowerPrec( var A, B : Integer; Bits : Byte);
21747:  Func MemEqu( const A, B, Sz : Integer) :Bool;
21748:  Func MaxIX( A, B : Integer) : Integer;
21749:  Func MinIX( A, B : Integer) : Integer;
21750:  Func MaxDX( A, B : DWORD) : DWORD;
21751:  Func MinDX( A, B : DWORD) : DWORD;
21752:  Func NulSearch( const Buffer:Str) : Integer;
21753:  Func NumBits( I : Integer) : Integer;
21754:  //Proc XAdd( var Critical, Normal);
21755:  //Proc XChg( var Critical, Normal);
21756:  Func CreateEvtA : DWORD;
21757:  Func CreateEvt( Initial :Bool) : DWORD;
21758:  Func SignaledEvt( id : DWORD) :Bool;
21759:  Func WaitEvt( const id : TWOHandleArray; Timeout : DWORD) : DWORD;
21760:  //Func WaitEvtA(nCount: Integer;lpHandles: PWOHandleArray; Timeout:DWORD): DWORD;
21761:  Func ClearHandle( var Handle : THandle) :Bool;
21762:  Proc CloseHandles( const Handles : array of DWORD);
21763:  Func FileExistsX( const FName :Str) :Bool;
21764:  Func FindExecutableX( FileName, Directory : PChar; Result : PChar) : HINST;
21765:  Func GetEnvVariable( const Name :Str) :Str;
21766:  Func GetFileNfo(const FName:Str;var Info: TFileInfo; NeedAttr:Boolean):Bool;
21767:  Func GetFileNfoByHandle( Handle : DWORD; var Info : TFileInfo) :Bool;
21768:  Func ZeroHandle( var Handle : THandle) :Bool;
21769:  Func _CreateFile( const FName :Str; Mode : TCreateFileModeSet) : DWORD;
21770:  Func _CreateFileSecurity(const
       FName:str;Mode:TCreateFileModeSet;lpSecurityAttributes:PSecurityAttributes):DWORD;
21771:  Func _GetFileSize( const FName :Str) : DWORD;
21772:  Func _MatchMaskBody( AName, AMask :Str; SupportPercent :Bool) :Bool;
21773:  Func _MatchMask(const AName:str; AMask :Str;SupportPercent:Boolean):Boolean;
21774:  Func MatchMask( const AName, AMask :Str) :Bool;
21775:  Func SysErrorMsg( ErrorCode : DWORD) :Str;
21776:  Func CreateRegKeyX( const AFName :Str) : HKey;
21777:  Func OpenRegKeyEx( const AName :Str; AMode : DWORD) : HKey;
21778:  Func OpenRegKey( const AName :Str) : DWORD;
21779:  Func ReadRegBin(Key: DWORD; const rvn :Str; Bin: _Pointer; Sz : DWORD) :Bool;
21780:  Func ReadRegInt( Key : DWORD; const AStrName :Str) : DWORD;
21781:  Func ReadRegString( Key : DWORD; const AStrName :Str) :Str;
21782:  Func WriteRegBin(Key:DWORD;const rvn :Str; Bin: _Pointer; Sz : DWORD):Bool;
21783:  Func WriteRegInt( Key : DWORD; const AStrName :Str; AValue : DWORD) :Bool;
21784:  Func WriteRegString( Key : DWORD; const AStrName, AStr :Str) :Bool;
21785:  Func AddrInet( i : DWORD) :Str;
21786:  Func GetHostNameByAddr( Addr : DWORD) :Str;
21787:  Func Inet2addr( const s :Str) : DWORD;
21788:  Func InetAddr( const s :Str) : DWORD;
21789:  Proc GlobalFail;
21790:  Func _LogOK( const Name :Str; var Handle : DWORD) :Bool;
21791:  Proc xBaseDone;
21792:  Proc xBaseInit;
21793:  Proc uCvtSetFileTime( T : DWORD; var L, H : DWORD);
21794:  Func uCvtGetFileTime( L, H : DWORD) : DWORD;
21795:  Func uGetSystemTime : DWORD;
21796:  Func VlX( const s :Str) : DWORD;
21797:  Func VlHX( const s :Str) : DWORD;
21798:  Func StrAsg( const Src :Str) :Str;
21799:  SIRegister_TResetterThread(CL);
21800:  CServerVersion','String').SetString( '1.93;
21801:  CServerProductName','String').SetString( 'TinyWeb;
21802: end;
21803:
21804: Proc SIRegister_ImageHistogram(CL: TPSPascalCompiler);
21805: begin
21806:   THistogramChannel', '( hclGray, hclRed, hclGreen, hclBlue );
21807:   THistogramArray', 'array[0..255] of Cardinal;;
21808:   //THistogramArray = array[0..255] of Cardinal;
21809:   SIRegister_THistogram(CL);
21810:  Proc AdjustBitmap(Bmp: TBitmap;Channel: THistogramChannel; Tolerance :Card);
21811:  Proc AdjustBitmap1(Bmp:TBitmap;Hist:THistogram;Channel:THistogramChannel;Tolerance:Card);
21812:  Proc AdjustBitmap2(Bmp: TBitmap; Low, High : Byte; Channel : THistogramChannel);
21813: end;
21814:
21815: Proc SIRegister_cCipherRSA(CL: TPSPascalCompiler);
21816: begin
21817:  CL.AddClassN(CL.FindClass('TOBJECT'),'ERSA;
21818:   TRSAPublicKey', 'record KeySize : Integer; Modulus : HugeWord; Exponent : HugeWord; end;
21819:   TRSAPrivateKey', 'record KeySize : Integer; Modulus : HugeWord; '
21820:   Exponent : HugeWord; PublicExponent : HugeWord; Prime1 : HugeWord; Prime2 '
21821:   : HugeWord;Phi:HugeWord; Exponent1:HugeWord; Exponent2:HugeWord; Coefficient:HugeWord; end;
21822:   CL.AddTypeS('TRSAMessage', 'HugeWord;
21823:  TRSAEncryptionType', '( rsaetPKCS1, rsaetOAEP );
21824:  Proc RSAPublicKeyInit( var Key : TRSAPublicKey);
21825:  Proc RSAPublicKeyFinalise( var Key : TRSAPublicKey);
21826:  Proc RSAPublicKeyAssign( var KeyD : TRSAPublicKey; const KeyS : TRSAPublicKey);
21827:  Proc RSAPublicKeyAssignHex(var Key:TRSAPublicKey;const KeySize:Int;const HexMod,HexExp:Str
21828: Proc RSAPublicKeyAssignBuf(var Key:TRSAPublicKey;const KeySize:Int;const ModBuf:str;const
       ModBufSize:Integer;const ExpBuf:str;const ExpBufSize:Integer;const ReverseByteOrder:Bool);
21829: Proc RSAPublicKeyAssignBufStr(var Key:TRSAPublicKey;const KeySize:Integer;const ModBuf,ExpBuf:Ansistr);
21830:  Proc RSAPrivateKeyInit( var Key : TRSAPrivateKey);
21831:  Proc RSAPrivateKeyFinalise( var Key : TRSAPrivateKey);
```

```
21832:  Proc RSAPrivateKeyAssign( var KeyD : TRSAPrivateKey; const KeyS : TRSAPrivateKey);
21833:  Proc RSAPrivateKeyAssignHex(var Key:TRSAPrivateKey;const KeySize:Integer;const HexMod, HexExp:Ansistr);
21834:  Proc RSAPrivateKeyAssignBuf(var Key:TRSAPrivateKey;const KeySize:Integer;const ModBuf:Str; const
       ModBufSize:Integer;const ExpBuf:Str; const ExpBufSize:Integer; const ReverseByteOrder:Boolean);
21835:  Proc RSAPrivateKeyAssignBufStr(var Key:TRSAPrivateKey;const KeySize:Integer;const ModBuf,ExpBuf:Ansistr);
21836:  Proc RSAGenerateKeys(const KeySize:Int;var PrivateKey:TRSAPrivateKey;var PublicKey:TRSAPublicKey);
21837:  Func RSACipherMessageBufSize(const KeySize: Integer) : Integer;
21838:  Proc RSAEncodeMessagePKCS1(const KeySize:Integer;const Buf:str;const BufSize:Integer;var EncodedMessage:
       TRSAMessage);
21839:  Proc RSAEncodeMessageOAEP(const KeySize:Integer;const Buf:str;const BufSize:Integer;var EncodedMessage:
       TRSAMessage);
21840:  Proc RSAEncryptMessage(const PublicKey:TRSAPublicKey;const PlainMessage:TRSAMessage;var
       CipherMessage:TRSAMessage);
21841:  Func RSACipherMessageToBuf(const KeySize:Integer;const CipherMessage:TRSAMessage;var CipherBuf:str;const
       CipherBufSize:Int):Int;
21842:  Func RSAEncryptC(const EncryptionType:TRSAEncryptionType; const PublicKey:TRSAPublicKey; const
       PlainBuf:str;const PlainBufSize:Int;var CipherBuf:str;const CipherBufSize:Int):Int;
21843:  Func RSAEncryptStr(const EncryptionType:TRSAEncryptionType;const PublicKey:TRSAPublicKey;const
       Plain:Ansistr): Ansistr;
21844:  Proc RSACipherBufToMessage(const KeySize:Integer;const CipherBuf:str;const CipherBufSize:Int;var
       CipherMessage:TRSAMessage);
21845:  Proc RSADecryptMessage(const PrivateKey:TRSAPrivateKey;const CipherMessage:TRSAMessage;var
       EncodedMessag:TRSAMessage);
21846:  Func RSADecodeMessagePKCS1(const KeySize:Integer;const EncodedMessage:HugeWord;var Buf:str;const BufSize:
       Integer):Integer;
21847:  Func RSADecodeMessageOAEP(const KeySize:Integer;const EncodedMessage:HugeWord;var Buf:str;const BufSize:
       Integer):Integer;
21848:  Func RSADecryptC( const EncryptionType : TRSAEncryptionType; const PrivateKey : TRSAPrivateKey; const
       CipherBuf :Str; const CipherBufSize : Integer; var PlainBuf :Str; const PlainBufSize : Integer) : Integer;
21849:  Func RSADecryptStr(const EncryptionType:TRSAEncryptionType;const PrivateKey:TRSAPrivateKey;const
       Cipher:Ansistr):Ansistr;
21850:  Proc SelfTestRSA;
21851:  //CL.AddDelphiFunction('Proc Profile;
21852: end;
21853:
21854: // add to V 4.2.6.10
21855: Proc SIRegister_Streams(CL: TPSPascalCompiler);
21856: begin
21857:  SIRegister_TBaseStream(CL);
21858:   SIRegister_TFilterStream(CL);
21859:   SIRegister_TSeekableStream(CL);
21860:   //CL.AddTypeS('TFileAttributes2', 'set of ( sfaArchive, sfaHidden, sfaSystem, sf'
21861:    //+'aReadOnly, sfaTemporary, sfaCompressed, sfaDirectory );
21862: TFileAttribute2,(sfaArchive,sfaHidden,sfaSystem,sfaReadOnly,sfaTemporary,sfaCompressed, sfaDirectory);
21863:  TFileAttributes2, set of TFileAttribute2');
21864:   TfsMode','(sfsRead, sfsWrite, sfsCreate, sfsShareRead, sfsShareWrite);
21865:   TfsModes', 'set of TfsMode');
21866: // CL.AddTypeS('TfsModes','set of( fsRead, fsWrite,fsCreate,fsShareRead,fsShareWrite );
21867:  fsReset','LongInt).Value.ts32 := ord(fsRead) or ord(fsWrite);
21868:  fsShared','LongInt).Value.ts32:= ord(fsShareRead) or ord(fsShareWrite);
21869:   SIRegister_TFileStream2(CL);
21870:   SIRegister_TDummyStream(CL);
21871:  Proc CopyStream( Source, Dest: TBaseStream; const Size : Integer);
21872:  Func GetLastErrorText :Str');
21873: end;
21874:
21875: Proc SIRegister_CromisStreams(CL: TPSPascalCompiler);
21876: begin
21877:  Proc WriteToStreamAsString( const Stream : TStream; const Content :Str);
21878:  Proc WriteToStreamAsUnicode( const Stream : TStream; const Content :Str);
21879:  Proc WriteToStreamAsUTF80( const Stream : TStream; const Content :Str);
21880:  Proc WriteToStreamAsUTF81( const Stream : TStream; const Content: Ansistr);
21881:  Func ReadFromStreamAsUnicode( const Stream : TStream) :Str');
21882:  Func ReadFromStreamAsString( const Stream : TStream) :Str');
21883:  Func ReadFromStreamAsUTF8( const Stream : TStream) : Ansistr');
21884:  CL.AddConstantN('cDefaultSearchSize','LongInt').SetInt( 50);
21885:   SIRegister_TNamesEnumerator(CL);
21886:   SIRegister_IStorageStream(CL);
21887:   SIRegister_TStreamStorage(CL);
21888: end;
21889:
21890: // new functions on 4.2.6.10
21891: Func ArcTan3(const X : Extended) : Extended');  //bugfix of ArcTan()
21892: Func WMIConnect2(WBemLocator:ISWBemLocator;namespace,Server,account,password:str): ISWBemServices;
21893: Func WMIConnect3(WBemLocator:ISWBemLocator;Server,namespace,account,password:str): ISWBemServices;
21894: Func CLSIDFromProgID(pszProgID: widestring; out clsid: TGUID): Longint'); //stdcall;
21895: Func ProgIDFromCLSID(const clsid:TGUID; out pszProgID: widestring): Longint'); //stdcall;
21896: Func StringFromCLSID(const clsid:TGUID; out psz: widestring): longint'); //stdcall;
21897: //{$EXTERNALSYM CLSIDFromString}
21898:  Func CLSIDFromString(psz: widestring; out clsid: TGUID): longint'); //stdcall;
21899: Func StringFromGUID2(const guid: TGUID;psz:widestring;cbMax:Integer):Integer'); //stdcall;
21900: Func ProgIDExists(const ProgID:WideString):Boolean');
21901: Func ByteArraytostring(const bin: array of byte):Str;
21902: Func ByteArraytoHex(const bin: array of byte):Str');
21903: Proc SaveFile(Contents : Ansistr; const FileName:Ansistr; Append :Bool);
21904: Proc SaveToFile(Contents: Ansistr;const FileName:Ansistr; Append :Bool);
21905: Func LoadFromFile( const FileName : Ansistr) : Ansistr');
21906: Func VariantIsObject(const V: Variant):Bool;
21907: Func DiskSpaceInfo(const Drive:str; out AvailBytes,TotalBytes,FreeBytes:Int64):Boolean;
```

```
21908: Func DriveDisplayName(const Drive:Str):Str;
21909: Proc DriveDisplayNames(const List: TStrings);
21910: Proc Log( const Line:Str; amemo: Tmemo);
21911: begin   amemo.Lines.Add( Line)  end;
21912: Proc LogFmtX( const Line:Str; amemo: TMemo; const Args: array of const);
21913: begin
21914:   Log( Format( Line, Args), amemo)
21915: end;
21916:
21917: Proc SIRegister_USha256(CL: TPSPascalCompiler);
21918: begin
21919:  type TSHA256HASH', 'array[0..7] of Cardinal');
21920:  type TSHAChunk', 'array[0..7] of Cardinal');
21921:  //TSHA256HASH = array[0..7] of Cardinal;
21922:   //TChunk = array[0..15] of Cardinal;
21923:  Func CalcDoubleSHA256( Msg : Ansistr) : TSHA256HASH;
21924:  Func CalcSHA256( Msg : Ansistr) : TSHA256HASH;
21925:  Func CalcSHA2561( Stream : TStream) : TSHA256HASH;
21926:  Func SHA256ToStr( Hash : TSHA256HASH) :Str;
21927:  Func CanBeModifiedOnLastChunk(MessageTotalLength:Int64;var startBytePos:int):Bool;
21928:  Proc PascalCoinPrepareLastChunk(const messageToHash:Ansistr;var stateForLastChunk:TSHA256HASH;var
       bufferForLastChunk : TSHAChunk);
21929:  Func ExecuteLastChunk(const stateForLastChunk:TSHA256HASH;const bufferForLastChunk:
       TSHAChunk;nPos:Integer;nOnce,Timestamp:Card):TSHA256HASH');
21930:  Func ExecuteLastChunkAndDoSha256(const stateForLastChunk:TSHA256HASH;const
       bufferForLastChunk:TSHAChunk;nPos:Integer;nOnce,Timestamp:Card):TSHA256HASH');
21931:  Proc PascalCoinExecuteLastChunkAndDoSha256(const stateForLastChunk:TSHA256HASH;const
       bufferForLastChunk:TSHAChunk;nPos:Integer;nOnce,Timestamp:Card;var ResultSha256:AnsiStr);
21932:  Func Sha256HashToRaw(const hash : TSHA256HASH): Ansistr');
21933: Func GetSHA256( Msg : Ansistr) :Str;
21934: Func GetDriveNumber(const Drive:Str): Integer;
21935: Func HardDiskSerial(const Drive:Str): DWORD;
21936: Func IsDriveReady2(const Drive:Str):Bool;
21937: Func Touchfile(const FileName:Str):Bool;
21938: Func URLFromShortcut(const Shortcut:Str):Str;
21939:
21940: Func GetSHA256(Msg: Ansistr):Str; //overload;
21941: var Stream: TMemoryStream;
21942: begin
21943:   Stream:= TMemoryStream.Create;
21944:   try
21945:     Stream.WriteBuffer(PAnsiChar(Msg)^,Length(Msg));
21946:     Stream.Position:= 0;
21947:     Result:= SHA256ToStr(CalcSHA256(Stream));
21948:   finally
21949:     Stream.Free;
21950:   end;
21951: end;
21952:
21953: Code Example Binary file:
21954:    sr:= filetostring(Exepath+'maXbox4.exe')
21955:    writeln(SHA256ToStr(CalcSHA256(sr)))
21956:    stringtofile(Exepath+'maXbox4.exe',sr,false); //append = false
21957:
21958: Proc SIRegister_uTPLb_BinaryUtils(CL: TPSPascalCompiler);
21959: begin
21960:  Func SwapEndien_u32(Value: uint32) : uint32');
21961:  Func SwapEndien_s64(Value: int64) : int64');
21962:  Func SwapEndien_u64(Value: uint64) : uint64');
21963:  Func RotateLeft1Bit_u32( Value : uint32) : uint32');
21964:  Proc Read_BigEndien_u32_Hex( const Value :Str; BinaryOut: TStream);
21965:  Func Get_TP_LockBox3_HINSTANCE : HMODULE');
21966: end;
21967:
21968: Proc SIRegister_Series(CL: TPSPascalCompiler);
21969: begin
21970:  CL.AddConstantN('PiDegree','Double').setExtended( Pi / 180.0);
21971:  Tee_CircledShadowColor','LongWord).SetUInt( TColor ( $A0A0A0 ));
21972:  CL.AddTypeS('TTreatNullsStyle', '( teetnDontPaint, teetnSkip, teetnIgnore );
21973:  SIRegister_TCustomLineSeries(CL);
21974:  CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomSeries');
21975:  CL.AddTypeS(TSeriesClickPointerEvent','Procedure(Sender:TCustomSeries;ValueIndex:Int;X,Y:Int;
21976:  TCustomSeriesStack','( cssNone, cssOverlap, cssStack, cssStack100 );
21977:  TOnGetPointerStyle','Func ( Sender : TChartSeries; ValueIndex: Integer) : TSeriesPointerStyle');
21978:  SIRegister_TCustomSeries(CL);
21979:   SIRegister_TLineSeries(CL);
21980:   SIRegister_THorizLineSeries(CL);
21981:   SIRegister_TPointSeries(CL);
21982:   CL.AddTypeS(TMultiArea','( mamaNone,maStacked, maStacked100 );
21983:   SIRegister_TAreaSeries(CL);
21984:   SIRegister_THorizAreaSeries(CL);
21985:  TMultiBar,(teembNone,teembSide,teembStacked,teembStacked100,teembSideAll,teembSelfStack);
21986:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomBarSeries');
21987:   TBarStyle,(teebsRectangle, teebsPyramid, teebsInvPyramid, teebsCilinder,teebsEllipse, teebsArrow,
       teebsRectGradient,teebsCone,teebsBevel,teebsSlantCube,teebsDiamond,teebsInvArrow, teebsInvCone);
21988:   TGetBarStyleEvent,Procedure(Sender:TCustomBarSeries;ValueIndex:Integer;var TheBarStyle: TBarStyle);
21989:   SIRegister_TBarSeriesGradient(CL);
21990:   SIRegister_TCustomBarSeries(CL);
21991:   SIRegister_TBarSeries(CL);
```

```
21992:   SIRegister_THorizBarSeries(CL);
21993:   SIRegister_TCircledShadow(CL);
21994:   SIRegister_TCircledSeries(CL);
21995:   TPieAngle', 'record StartAngle : Double; MidAngle : Double; EndAngle : Double; end');
21996:   TPieAngles', 'array of TPieAngle');
21997:   SIRegister_TSliceValueList(CL);
21998:   TPieOtherStyle', '( piepoNone, piepoBelowPercent, piepoBelowValue );
21999:   SIRegister_TPieOtherSlice(CL);
22000:   SIRegister_TPieMarks(CL);
22001:   TMultiPie', '( teempAutomatic, teempDisabled );
22002:   SIRegister_TPieSeries(CL);
22003:   SIRegister_TFastLineSeries(CL);
22004:  //CL.AddConstantN('bsCylinder','').SetString( bsCilinder);
22005:  Proc RegisterTeeStandardSeries');
22006:  Proc TeePointerDrawLegend(Pointer:TSeriesPointer;AColor:TColor;const Rect:TRect;DrawPen:Bool;
       AStyle:TSeriesPointerStyle);
22007: end;
22008:
22009: Proc SIRegister_UJSONFunctions(CL: TPSPascalCompiler);
22010: begin
22011:   CL.AddTypeS('TJSONValue', 'TJSONData');
22012:   SIRegister_TPCJSONData(CL);
22013:   //CL.AddTypeS('TPCJSONDataClass', 'class of TPCJSONData');
22014:   SIRegister_TPCJSONVariantValue(CL);
22015:   SIRegister_TPCJSONNameValue(CL);
22016:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPCJSONArray');
22017:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPCJSONObject');
22018:   SIRegister_TPCJSONList(CL);
22019:   SIRegister_TPCJSONArray(CL);
22020:   SIRegister_TPCJSONObject(CL);
22021:   CL.AddClassN(CL.FindClass(TOBJECT),EPCParametresError');
22022: end;
22023:
22024: Proc SIRegister_uTPLb_Hash(CL: TPSPascalCompiler);
22025: begin
22026:   CL.AddTypeS(TOnHashProgress,Function(sender:TObject;CountBytesProcessed:int64):bool;
22027:   SIRegister_IHash(CL);
22028:   SIRegister_IHash_TestAccess(CL);
22029:   SIRegister_TSimpleHash(CL);
22030:   SIRegister_THash(CL);
22031: end;
22032:
22033: Proc SIRegister_UTime(CL: TPSPascalCompiler);
22034: begin
22035:  //Func TzSpecificLocalTimeToSystemTime( lpTimeZoneInformation:PTimeZoneInformation; var lpLocalTime,
       lpUniversalTime : TSystemTime) : BOOL');
22036:  //Func SystemTimeToTzSpecificLocalTime( lpTimeZoneInformation:PTimeZoneInformation;var lpUniversalTime,
       lpLocalTime : TSystemTime) : BOOL');
22037:  Func DateTime2UnivDateTime(d : TDateTime) : TDateTime');
22038:  Func UnivDateTime2LocalDateTime(d: TDateTime): TDateTime');
22039:  Func UnivDateTimeToUnix(dtDate: TDateTime : Longint');
22040:  Func UnixToUnivDateTime(USec: Longint): TDateTime');
22041:  Func UnixTimeToLocalElapsedTime( USec : Longint): Ansistr');
22042:  Func DateTimeElapsedTime(dtDate: TDateTime): Ansistr');
22043: end;
22044:
22045: Proc SIRegister_uTPLb_BlockCipher(CL: TPSPascalCompiler);
22046: begin
22047:   SIRegister_IBlockCodec(CL);
22048:   SIRegister_IBlockCipher(CL);
22049:   SIRegister_TBlockChainLink(CL);
22050:   CL.AddTypeS(TChainingFeature', '( cfNoNounce, cfKeyStream, cfAutoXOR, cf8bit);
22051:   CL.AddTypeS(TChainingFeatureSet', 'set of TChainingFeature');
22052:   SIRegister_IBlockChainingModel(CL);
22053:   SIRegister_IBlockCipherSelector(CL);
22054: end;
22055:
22056: Proc SIRegister_XMLDoc(CL: TPSPascalCompiler);
22057: begin
22058:   FindInterface('IUNKNOWN'),IXMLNode, 'IXMLNode');
22059:   TOBJECT'),'TXMLNode');
22060:   TOBJECT'),'TXMLNodeList');
22061:   TOBJECT'),'TXMLNodeCollection');
22062:   TOBJECT'),'TXMLDocument');
22063:   CL.AddTypeS(TNodeListOperation','( nlInsert, nlRemove, nlCreateNode );
22064:   CL.AddTypeS(TNodeListNotification', 'Proc ( Operation : TNodeListOperat'
22065:    +ion; var Node :IXMLNode; const IndexOrName: OleVariant; BeforeOperation :Bool);
22066:   SIRegister_TXMLNodeList(CL);
22067:    CL.AddTypeS(TAsyncEventHandler,procedure(Sender:TObject;AsyncLoadState:Integer) of Object);
22068:   //CL.AddTypeS('TXMLNodeClass', 'class of TXMLNode');
22069:   TXMLNodeArray', 'array of TXMLNode');
22070:   TNodeClassInfo','record NodeName:DOMString;NamespaceURI:DOMString;NodeClass:TXMLNode; end');
22071:   CL.AddTypeS(TNodeClassArray', 'array of TNodeClassInfo');
22072:   //CL.AddTypeS('TXMLNodeCollectionClass', 'class of TXMLNodeCollection');
22073:   CL.AddTypeS(TNodeChange', '( ncUpdateValue, ncInsertChild, ncRemoveChild, nc'
22074:    +AddAttribute, ncRemoveAttribute );
22075:   SIRegister_IXMLNodeAccess(CL);
22076:   SIRegister_TXMLNode(CL);
22077:   SIRegister_TXMLNodeCollection(CL);
```

```
22078:   TNodeChangeEvent','Proc ( const Node : IXMLNode; ChangeType : TNodeChange);
22079:   TXMLPrologItem','( xmlpVersion, xmlpEncoding, xmlpStandalone );
22080:   TXMLDocumentSource','( xdsNone, xdsXMLProperty, xdsXMLData, xdsFile, xdsStream);
22081:   SIRegister_IXMLDocumentAccess(CL);
22082:   SIRegister_TXMLDocument(CL);
22083: Func CreateDOMNode(Doc:IDOMDocument;const NameOrData:DOMString;NodeType:TNodeType;const
       AddlData:DOMString):IDOMNode;
22084: Func DetectCharEncoding( S : TStream) : TXmlEncodingType');
22085: Proc CheckEncoding(var XMLData: DOMString; const ValidEncodings: array of string);
22086: Func XMLStringToWideString( const XMLString :Str) : WideString;
22087: Func FormatXMLData( const XMLData : DOMString) : DOMString;
22088: Func LoadXMLDocument3( const FileName : DOMString) : IXMLDocument');
22089: Func LoadXMLData( const XMLData : DOMString) : IXMLDocument;
22090: Func LoadXMLData2( const XMLData :Str) : IXMLDocument;
22091: Func NewXMLDocument( Version : DOMString) : IXMLDocument');
22092: Proc XMLDocError( const Msg :Str);
22093: Proc XMLDocError2( const Msg :Str; const Args : array of const);
22094: Func CreateXMLDocument()
22095: end;
22096:
22097: Proc SIRegister_uTPLb_Codec(CL: TPSPascalCompiler);
22098: begin
22099:   SIRegister_TSimpleCodec(CL);
22100:   SIRegister_ICodec_TestAccess(CL);
22101:   SIRegister_TCodec(CL);
22102: end;
22103:
22104: Proc SIRegister_uTPLb_StreamCipher(CL: TPSPascalCompiler);
22105: begin
22106:   CL.AddTypeS(TAlgorithmicFeature', '( afStar, afCryptographicallyWeak, afNotI'
22107:    +mplementedYet, afForTestOnly, afForRunTimeOnly, afEncumberedByPatent, afEn'
22108:    +cumberedByCopyRight, afOpenSourceSoftware, afCommercial, afCompressor, afC'
22109:    +onverter, afBlockAdapter, afDisplayNameOnKeySize, afDoesNotNeedSalt, afAsymetric );
22110:   CL.AddTypeS(TAlgorithmicFeatureSet', 'set of TAlgorithmicFeature');
22111:   SIRegister_ICryptoGraphicAlgorithm(CL);
22112:   SIRegister_TSymetricKey(CL);
22113:   SIRegister_IStreamEncryptor(CL);
22114:   SIRegister_IStreamDecryptor(CL);
22115:   SIRegister_IStreamCipher(CL);
22116:   SIRegister_IisBase64Converter(CL);
22117: end;
22118:
22119: Proc SIRegister_MidasCon(CL: TPSPascalCompiler);
22120: begin
22121:   SIRegister_TRemoteServer(CL);
22122:   CL.AddTypeS(TConnectType', '( ctDCOM, ctSockets, ctOLEnterprise );
22123:   SIRegister_TMidasConnection(CL);
22124: end;
22125:
22126: Proc SIRegister_IXmlDocument2(CL: TPSPascalCompiler);
22127: begin
22128:   //with RegInterfaceS(CL,'IXmlNode', 'IXmlDocument') do
22129:   with CL.AddInterface(CL.FindInterface('IXmlBase'),IXmlNode2,'IXmlDocument2') do begin
22130:   Func Get_DocumentElement : IXmlElement', cdRegister);
22131:   Func Get_BinaryXML : RawByteString', r);
22132:   Func Get_PreserveWhiteSpace :Bool', );
22133:   Proc Set_PreserveWhiteSpace( aValue :Bool)', );
22134:   Func Get_OnTagEnd : THookTag', cdRegister);
22135:   Proc Set_OnTagEnd( aValue : THookTag)', cdRegister);
22136:   Func Get_OnTagBegin : THookTag', cdRegister);
22137:   Proc Set_OnTagBegin( aValue : THookTag)', cdRegister);
22138:   Func NewDocument(const aVersion,anEncoding:TXmlString;aRootElementNameID:NativeInt): IXmlElement;
22139:   Func NewDocument1(const aVersion,anEncoding,aRootElementName:TXmlString): IXmlElement;
22140:   Func CreateElement( aNameID : NativeInt) : IXmlElement;',);
22141:   Func CreateElement1(const aName : TXmlString) : IXmlElement;',);
22142:   Func CreateText(const aData : TXmlString) : IXmlText',);
22143:   Func CreateCDATASection( const aData : TXmlString) : IXmlCDATASection',);
22144:   Func CreateComment( const aData : TXmlString) : IXmlComment',);
22145:   Func CreateProcessingInstruction(const aTarget,aData: TXmlString): IXmlProcessingInstruction;',);
22146:   Func CreateProcessingInstruction1(aTargetID:NativeInt;const aData:TXmlString):IXmlProcessingInstruction);
22147:   Proc LoadXML( const aXML : RawByteString)',);
22148:   Proc LoadBinaryXML( const aXML : RawByteString)',);
22149:   Proc Load( aStream : TStream);',);
22150:   Proc Load1( const aFileName :Str);',);
22151:   Proc LoadResource( aType, aName : PChar)',);
22152:   Proc Save( aStream : TStream);',);
22153:   Proc Save1( const aFileName :Str);',);
22154:   Proc SaveBinary( aStream : TStream; anOptions : LongWord);',);
22155:   Proc SaveBinary1( const aFileName :Str; anOptions : LongWord);',);
22156:   end;
22157: end;
22158:
22159: Proc SIRegister_OmniXMLUtils(CL: TPSPascalCompiler);
22160: begin
22161:  // XmlString = WideString;
22162:  CL.AddTypeS('XmlString', 'WideString');
22163:  Func XMLStrToReal38( nodeValue : XmlString; var value : real) :Bool;
22164:  Func XMLStrToReal39( nodeValue : XmlString) : real;
22165:  Func XMLStrToRealDef( nodeValue : XmlString; defaultValue : real) : real');
```

```
22166:  Func XMLStrToExtended40( nodeValue : XmlString; var value : extended):Bool;
22167:  Func XMLStrToExtended41( nodeValue : XmlString) : extended;
22168:  Func XMLStrToExtendedDef( nodeValue : XmlString; defaultValue : extended): extended;
22169:  Func XMLStrToCurrency42( nodeValue : XmlString; var value : Currency):Bool;
22170:  Func XMLStrToCurrency43( nodeValue : XmlString) : Currency;
22171:  Func XMLStrToCurrencyDef( nodeValue : XmlString; defaultValue : Currency): Currency;
22172:  Func XMLStrToInt44( nodeValue : XmlString; var value : integer) :Bool;
22173:  Func XMLStrToInt45( nodeValue : XmlString) : integer;
22174:  Func XMLStrToIntDef( nodeValue : XmlString; defaultValue : integer) : integer');
22175:  Func XMLStrToInt6446( nodeValue : XmlString; var value : int64) :Bool;
22176:  Func XMLStrToInt6447( nodeValue : XmlString) : int64;
22177:  Func XMLStrToInt64Def( nodeValue : XmlString; defaultValue : int64) : int64');
22178:  Func XMLStrToBool48( nodeValue : XmlString; var value :Bool) :Bool;
22179:  Func XMLStrToBool49( nodeValue : XmlString) :Bool;
22180:  Func XMLStrToBoolDef( nodeValue : XmlString; defaultValue :Bool):Bool;
22181:  Func XMLStrToDateTime50( nodeValue : XmlString; var value : TDateTime):Bool;
22182:  Func XMLStrToDateTime51( nodeValue : XmlString) : TDateTime;
22183:  Func XMLStrToDateTimeDef(nodeValue : XmlString; defaultValue : TDateTime): TDateTime;
22184:  Func XMLStrToDate52( nodeValue : XmlString; var value : TDateTime):Bool;
22185:  Func XMLStrToDate53( nodeValue : XmlString) : TDateTime;
22186:  Func XMLStrToDateDef( nodeValue : XmlString; defaultValue : TDateTime) : TDateTime;
22187:  Func XMLStrToTime54( nodeValue : XmlString; var value : TDateTime):Bool;
22188:  Func XMLStrToTime55( nodeValue : XmlString) : TDateTime;
22189:  Func XMLStrToTimeDef( nodeValue : XmlString; defaultValue : TDateTime): TDateTime;
22190:  Func XMLStrToBinary( nodeValue : XmlString; const value : TStream):Bool');
22191:  Func XMLRealToStr( value : real; precision : byte) : XmlString');
22192:  Func XMLExtendedToStr( value : extended) : XmlString');
22193:  Func XMLCurrencyToStr( value : Currency) : XmlString');
22194:  Func XMLIntToStr( value : integer) : XmlString');
22195:  Func XMLInt64ToStr( value : int64) : XmlString');
22196:  Func XMLBoolToStr( value :Bool; useBoolStrs :Bool) : XmlString');
22197:  Func XMLDateTimeToStr2( value : TDateTime) : XmlString');
22198:  Func XMLDateTimeToStrEx( value : TDateTime) : XmlString');
22199:  Func XMLDateToStr2( value : TDateTime) : XmlString');
22200:  Func XMLTimeToStr( value : TDateTime) : XmlString');
22201:  Func XMLBinaryToStr( value : TStream) : XmlString');
22202:  Func XMLVariantToStr( value : Variant) : XmlString');
22203:  end;
22204:
22205: Proc SIRegister_IdCoderHeader(CL: TPSPascalCompiler);
22206: begin
22207:  CL.AddTypeS('TTransfer', ( bit7, bit8, iso2022jp );
22208: Func EncodeAddressItem(EmailAddr:TIdEmailAddressItem; const HeaderEncoding: Char;
22209:   TransferHeader: TTransfer; MimeCharSet:Str):Str;
22210: Func EncodeHeader(const Header:Str; specials : CSET2; const HeaderEncoding: Char;
22211:  TransferHeader: TTransfer; MimeCharSet:Str):Str;
22212: Func Encode2022JP(const S:Str):Str;
22213: Func EncodeAddress(EmailAddr:TIdEMailAddressList;const HeaderEncoding:Char;
22214:   TransferHeader: TTransfer; MimeCharSet:Str):Str;
22215: Func DecodeHeader(Header:Str):str;
22216: Func Decode2022JP(const S:Str):Str;
22217: Proc DecodeAddress(EMailAddr : TIdEmailAddressItem);
22218: Proc DecodeAddresses(AEMails :Str; EMailAddr : TIdEmailAddressList);
22219: Proc InitializeISO(var TransferHeader:TTransfer;var HeaderEncoding:char;var CharSet:str);
22220: end;
22221:
22222: Proc SIRegister_FannNetwork(CL: TPSPascalCompiler);
22223: begin
22224:  // CL.AddTypeS('PCardinalArray', '^CardinalArray // will not work');
22225:   CL.AddConstantN('DLL_FILE','String').SetString( 'fannfloat.dll');
22226:   CL.AddTypeS('fann_type', 'single');
22227:   Fann_Type_Array', 'array [0..65535] of fann_type');
22228:   TFann_Type_Array', 'array [0..65535] of fann_type');
22229:   TFann_Type_Array2', 'array of fann_type');
22230:   TFann_Type_Array3', 'array of single');
22231:   TTrainingAlgorithm', '( taFANN_TRAIN_INCREMENTAL, taFANN_TRAIN_B'
22232:    +ATCH, taFANN_TRAIN_RPROP, taFANN_TRAIN_QUICKPROP );
22233:   CL.AddTypeS(TActivationFunction', '( afFANN_LINEAR, afFANN_THRESHOLD, afFANN'
22234:    +'_THRESHOLD_SYMMETRIC, afFANN_SIGMOID, afFANN_SIGMOID_STEPWISE, afFANN_SIGM'
22235:    +'OID_SYMMETRIC, afFANN_SIGMOID_SYMMETRIC_STEPWISE, afFANN_GAUSSIAN, afFANN_'
22236:    +'GAUSSIAN_SYMMETRIC, afFANN_GAUSSIAN_STEPWISE, afFANN_ELLIOT, afFANN_ELLIOT'
22237:    +'_SYMMETRIC, afFANN_LINEAR_PIECE, afFANN_LINEAR_PIECE_SYMMETRIC, afFANN_SIN'
22238:    +'_SYMMETRIC, afFANN_COS_SYMMETRIC, afFANN_SIN, afFANN_COS );
22239:   SIRegister_TFannNetwork(CL);  //CL.AddDelphiFunction('Proc Register');
22240: end;
22241:
22242: Proc SIRegister_TFannNetwork(CL: TPSPascalCompiler);
22243: begin
22244:   //with RegClassS(CL,'TComponent', 'TFannNetwork') do
22245:  with CL.AddClassN(CL.FindClass('TComponent'),'TFannNetwork') do begin
22246:   Constructor Create( Aowner : TComponent);
22247:   Proc Build( );
22248:   Proc Free');
22249:   Proc UnBuild( );
22250:   Func Train( Input : array of fann_type; Output : array of fann_type) : single');
22251:   Proc TrainOnFile(FileName:str; MaxEpochs:Card;DesiredError: Single);
22252:   Proc Run(Inputs: array of fann_type; var Outputs : TFann_Type_Array2);
22253:   Proc Run2(Inputs: array of fann_type; var Outputs : array of fann_type);
22254:   Proc Run3( Inputs : array of fann_type; var Outputs : TFann_Type_Array3);
```

```
22255:   Proc Run4( Inputs : array of fann_type; var Outputs : TFann_Type_Array3);
22256:   Proc SaveToFile( FileName :Str);
22257:   Proc LoadFromFile( Filename :Str);
22258:   FannObject', 'PFann', iptr);
22259:   Layers', 'TStrings', iptrw);
22260:   LearningRate', 'Single', iptrw);
22261:   ConnectionRate', 'Single', iptrw);
22262:   LearningMometum', 'single', iptrw);
22263:   MSE', 'Single', iptr);
22264:   TrainingAlgorithm', 'TTrainingAlgorithm', iptrw);
22265:   ActivationFunctionHidden', 'TActivationFunction', iptrw);
22266:   ActivationFunctionOutput', 'TActivationFunction', iptrw);
22267:  end;
22268: end;
22269: TFannNetwork encapsulates the Fast Artificial Neural Network. fann.sourceforge.net
22270:
22271: Proc SIRegister_RTLDateTimeplus(CL: TPSPascalCompiler);
22272: begin
22273:   SIRegister_TNullableDateTime(CL);
22274:  Func IsValidDate2( const Value :Str; out ADate : TDatetime) :Bool;
22275:  Func IsValidTime2( const Value :Str; out ATime : TDateTime) :Bool;
22276:  Func IsValidDateTime2( const Value :Str; out ADateTime : TDateTime) :Bool;
22277:  Proc ChangeSystemTime( const Value : TDateTime);
22278:  Func TryChangeSystemTime( const Value : TDateTime) :Bool');
22279: end;
22280:
22281: // new functions on V 4.2.8.10  III /IV  - V 4.5.8.10 IV
22282: Func IsUTF8File(const FileName:Str):Bool;
22283: Func IsUTF16File(const FileName:Str):Bool;
22284: Proc RichEditToCanvas(RichEdit: TRichEdit; Canvas: TCanvas; PixelsPerInch:Integer);
22285: --> RichEditToCanvas(RichEdit1, Image1.Canvas, Self.PixelsPerInch);
22286: Func {$I ActiveDEP:Card; }
22287: Func {$I ActivateDEP:Card;}
22288: Proc WriteXMLFile(doc: TXMLDocument; const AFileName:Str);
22289: Proc WriteXML(Element: TDOMNode; const AFileName:Str);
22290: Func ReadXMLFile(const AFilename:Str) :  TXMLDocument;
22291: Proc WriteXMLFile(doc: TXMLDocumentDOM; const AFileName:Str);
22292: Proc WriteXML(Element: TDOMNode; const AFileName:Str);
22293: Proc WriteXMLFileStream(ADoc : TXMLDocument; AStream : TStream);
22294:
22295: Func CountColors(const Bitmap: Graphics.TBitmap): Integer;
22296: var  Flags: array[Byte, Byte] of Classes.TBits;
22297:   I, J, K: Integer;
22298:   RowIn: PRGBTripleArray;
22299: begin
22300:   // Be sure bitmap is 24-bits/pixel
22301:   Assert(Bitmap.PixelFormat = Graphics.pf24Bit);
22302:
22303: Func SndPlaySound(const AFilename: pchar; flags:Card):Bool;
22304: Proc PlaySound1(const AFilename:Str);
22305: Func mciSendCommand(mciID, umessage, dwparam1, dwparam2:Card):Card;
22306: Func ProxyActive:Bool;
22307: Func GetProxyServer:Str;
22308: Func GetProxyOverride:Str;
22309: Func GetProxyServerIP:Str;
22310: Func GetProxyServerPort: Integer;
22311: Func GetHtml( const Url :Str) :Str;
22312: Func GetHtm12( const Url, UserAgent :Str) :Str;
22313: Proc FormatXMLFile(const XmlFile:str);
22314: Func FormatXMLFile2String(const XmlFile:str):Str;
22315: Func selectXMLNode(xnRoot: IXmlNode; const nodePath: WideString): IXmlNode;
22316: Func HTMLEncode3(const Data:Str):Str; //vfast
22317: Func IsSpecialFolderSupported(CSIDL: Integer):Bool;
22318: Func SpecialFolderPath(CSIDL: Integer):Str;
22319: Func DataSetToXML(const ADataSet: TDataSet):Str;
22320: Func PackageParams( Params : TParams; Types : TParamTypes) : OleVariant');
22321: Proc UnpackParams( const Source : OleVariant; Dest : TParams);
22322: Proc ActivateDEP2;
22323: Func JavaToDelphiDateTime(const dt: int64): TDateTime;
22324: Func DelphiToJavaDateTime(const dt: TDateTime): int64;
22325: Func GetTimeBias: integer;
22326: Proc WrapLines(VAR a, b:Str; len: Integer);
22327: Func DelphiDateTimeToISO8601Date(dt: TDateTime):Str;
22328: Proc BackgroundWorkerWaitFor(isworking:Bool; fWindow: HWND);
22329:
22330: uPSI_CromisStreams,
22331: uPSI_Streams,
22332: uPSI_BitStream,
22333: uPSI_UJSONFunctions.pas
22334: uPSI_uTPLb_BinaryUtils.pas
22335: uPSI_USha256.pas
22336: uPSI_uTPLb_HashDsc.pas
22337: uPSI_uTPLb_Hash.pas
22338: SIRegister_Series(X);  //4.2.6.10
22339: uPSI_Series,
22340: unit uPSI_mimeinln;
22341: unit uPSI_UTime; //unit UTime;
22342: RIRegister_TStreamStorage(X);
22343: RIRegister_TNamesEnumerator(X);
```

```
22344: 1283 uPSI_uTPLb_StreamCipher.pas
22345: 1284 uPSI_uTPLb_BlockCipher.pas
22346: 1285 uPSI_uTPLb_Asymetric.pas
22347: 1286 uPSI_uTPLb_CodecIntf.pas
22348: 1287 uPSI_uTPLb_Codec.pas
22349: 1288 uPSI_ADOInt.pas
22350: 1289 uPSI_MidasCon.pas
22351: 1290 uPSI_XMLDoc.pas
22352: 1291 uPSI_XMLIntf.pas
22353: 1292 uPSI_ProxyUtils.pas
22354: 1293 unit uPSI_maxXMLUtils2;
22355: 1294 unit_StDict_Routines(S: TPSExec);
22356: 1295 unit uPSI_Hashes2
22357: 1296 unit uPSI_IdCoderHeader;
22358:
22359: Func TRestRequest_createStringStreamFromStringList(strings:TStringList):TStringStream;
22360: {A simple Oscilloscope using  TWaveIn class.
22361:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
22362:  http://www.retroarchive.org/garbo/pc/turbopas/index.html
22363: uses Forms,
22364:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
22365:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
22366:   uColorFunctions in 'uColorFunctions.pas',
22367:   AMixer in 'AMixer.pas',
22368:   uSettings in 'uSettings.pas',
22369:   UWavein4 in 'UWavein4.pas',
22370:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
22371:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
22372:   BSpectrum in unit uPSI_BSpectrum; {SpectraLibrary}
22373:   1250 unit uPSI_U_Oscilloscope4_2
22374:
22375: // new functions on V 4.6.3.10
22376: Proc SIRegister_cHash(CL: TObject);
22377: var sha: TSha256hash2;
22378: begin
22379:  DigestToHexBuf( const Digest :Str; const Size : Integer; const Buf);
22380:  DigestToHex( const Digest :Str; const Size : Integer) : Ansistr');
22381:  Digest128Equal( const Digest1, Digest2 : T128BitDigest) :Bool');
22382:  Digest160Equal( const Digest1, Digest2 : T160BitDigest) :Bool');
22383:  Digest224Equal( const Digest1, Digest2 : T224BitDigest) :Bool');
22384:  Digest256Equal( const Digest1, Digest2 : T256BitDigest) :Bool');
22385:  Digest384Equal( const Digest1, Digest2 : T384BitDigest) :Bool');
22386:  Digest512Equal( const Digest1, Digest2 : T512BitDigest) :Bool');
22387:    hashNoError','LongInt').SetInt( 0);
22388:    hashInternalError','LongInt').SetInt( 1);
22389:    hashInvalidHashType','LongInt').SetInt( 2);
22390:    hashInvalidBuffer','LongInt').SetInt( 3);
22391:    hashInvalidBufferSize','LongInt').SetInt( 4);
22392:    hashInvalidDigest','LongInt').SetInt( 5);
22393:    hashInvalidKey','LongInt').SetInt( 6);
22394:    hashInvalidFileName','LongInt').SetInt( 7);
22395:    hashFileOpenError','LongInt').SetInt( 8);
22396:    hashFileSeekError','LongInt').SetInt( 9);
22397:    hashFileReadError','LongInt').SetInt( 10);
22398:    hashNotKeyedHashType','LongInt').SetInt( 11);
22399:    hashTooManyOpenHandles','LongInt').SetInt( 12);
22400:    hashInvalidHandle','LongInt').SetInt( 13);
22401:    hashMAX_ERROR','LongInt').SetInt( 13);
22402:  GetHashErrorMessage( const ErrorCode : LongWord) : PChar');
22403: /SIRegister_EHashError(CL);
22404:  SecureClear( var Buf :Str; const BufSize : Integer);
22405:  SecureClear512( var Buf : T512BitBuf);
22406:  SecureClear1024( var Buf : T1024BitBuf);
22407:  SecureClearStr2( var S : Ansistr);       -------------------
22408:  CalcChecksum32( const Buf : Ansistr) : LongWord');
22409:  CalcXOR8( const Buf : Ansistr) : Byte');
22410:  CalcXOR16( const Buf : Ansistr) : Word');
22411:  CalcXOR32( const Buf : Ansistr) : LongWord');
22412:  CRC16Init( var CRC16 : Word);
22413:  CRC16Byte( const CRC16 : Word; const Octet : Byte) : Word');
22414:  CRC16Buf( const CRC16 : Word; const Buf :Str; const BufSize : Integer) : Word');
22415:  CalcCRC16( const Buf : Ansistr) : Word');
22416:  SetCRC32Poly( const Poly : LongWord);
22417:  CRC32Init( var CRC32 : LongWord);
22418:  CRC32Byte( const CRC32 : LongWord; const Octet : Byte) : LongWord');
22419:  CRC32Buf( const CRC32 : LongWord; const Buf :Str; const BufSize : Integer): LongWord');
22420:  CRC32BufNoCase(const CRC32: LongWord; const Buf:Str; const BufSize: Integer): LongWord');
22421:  CalcCRC32( const Buf : Ansistr) : LongWord');
22422:  Adler32Init( var Adler32 : LongWord);
22423:  Adler32Byte( const Adler32 : LongWord; const Octet : Byte) : LongWord');
22424:  Adler32Buf( const Adler32:LongWord;const Buf:str; const BufSize:Integer): LongWord');
22425:  CalcAdler32( const Buf : Ansistr) : LongWord');
22426:  ELFInit( var Digest : LongWord);
22427:  ELFBuf( const Digest : LongWord; const Buf :Str; const BufSize : Integer):LongWord');
22428:  CalcELF( const Buf : Ansistr) : LongWord');
22429:  IsValidISBN2( const S : Ansistr) :Bool'); -------------------------
22430:  IsValidLUHN( const S : Ansistr) :Bool');
22431:  KnuthHash( const S : Ansistr) : LongWord');
22432:  MD5InitDigest( var Digest : T128BitDigest);
```

```
22433:  MD5Buf( var Digest : T128BitDigest; const Buf :Str; const BufSize : Integer);
22434:  MD5FinalBuf(var Digest:T128BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22435:  CalcMD5( const Buf : Ansistr) : T128BitDigest');
22436:  MD5DigestAsString( const Digest : T128BitDigest) : Ansistr');
22437:  MD5DigestToHex( const Digest : T128BitDigest) : Ansistr');
22438:  SHA1InitDigest( var Digest : T160BitDigest);
22439:  SHA1Buf( var Digest : T160BitDigest; const Buf :Str; const BufSize : Integer);
22440:  SHA1FinalBuf(var Digest:T160BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22441:  CalcSHA1( const Buf : Ansistr) : T160BitDigest');
22442:  SHA1DigestAsString( const Digest : T160BitDigest) : Ansistr');
22443:  SHA1DigestToHex( const Digest : T160BitDigest) : Ansistr');
22444:  SHA224InitDigest( var Digest : T256BitDigest);
22445:  SHA224Buf( var Digest : T256BitDigest; const Buf:Str;const BufSize : Integer);
22446: SHA224FinalBuf(var Digest:T256BitDigest;const Buf:str;const BufSize:Integer;const TotalSize:Int64;var
       OutDigest:T224BitDigest);
22447:   CalcSHA224( const Buf : Ansistr) : T224BitDigest');
22448:  SHA224DigestAsString( const Digest : T224BitDigest) : Ansistr');
22449:  SHA224DigestToHex( const Digest : T224BitDigest) : Ansistr');
22450:  SHA256InitDigest( var Digest : T256BitDigest);
22451:  SHA256Buf( var Digest : T256BitDigest; const Buf :Str; const BufSize : Integer);
22452:  SHA256FinalBuf(var Digest:T256BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64;
22453:  CalcSHA2562( const Buf : Ansistr) : T256BitDigest');  --------------------------
22454:  SHA256DigestAsString( const Digest : T256BitDigest) : Ansistr');
22455:  SHA256DigestToHex( const Digest : T256BitDigest) : Ansistr');
22456:  SHA384InitDigest( var Digest : T512BitDigest);
22457:  SHA384Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22458:  SHA384FinalBuf(var Digest:T512BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64;var
       OutDigest:T384BitDigest);
22459:  CalcSHA384( const Buf : Ansistr) : T384BitDigest');
22460:  SHA384DigestAsString( const Digest : T384BitDigest) : Ansistr');
22461:  SHA384DigestToHex( const Digest : T384BitDigest) : Ansistr');
22462:  SHA512InitDigest( var Digest : T512BitDigest);
22463:  SHA512Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22464:  SHA512FinalBuf(var Digest:T512BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22465:  CalcSHA512( const Buf : Ansistr) : T512BitDigest');
22466:  SHA512DigestAsString( const Digest : T512BitDigest) : Ansistr');
22467:  SHA512DigestToHex( const Digest : T512BitDigest) : Ansistr');
22468:  HMAC_MD5Init(const Key:str;const KeySize:Integer;var Digest:T128BitDigest;var K:T512BitBuf);
22469:  HMAC_MD5Buf( var Digest : T128BitDigest; const Buf :Str; const BufSize : Integer);
22470:  HMAC_MD5FinalBuf(const K:T512BitBuf;var Digest T128BitDigest;const Buf:str;const BufSize:Integer;const
       TotalSize:Int64);
22471:  CalcHMAC_MD5( const Key, Buf : Ansistr) : T128BitDigest');
22472:   HMAC_SHA1Init(const Key:str;const KeySize:Int;var Digest:T160BitDigest;var K:T512BitBuf);
22473:   HMAC_SHA1Buf( var Digest : T160BitDigest; const Buf :Str; const BufSize : Integer);
22474: HMAC_SHA1FinalBuf(const K:T512BitBuf;ar Digest:T160BitDigest;const Buf:Str;const BufSize:Int;const
       TotalSize Int64);
22475:  CalcHMAC_SHA1( const Key, Buf : Ansistr):T160BitDigest;
22476:  HMAC_SHA256Init(const Key:str;const KeySize:Int;var Digest:T256BitDigest;var K:T512BitBuf);
22477:  HMAC_SHA256Buf( var Digest : T256BitDigest; const Buf :Str; const BufSize : Integer);
22478:  HMAC_SHA256FinalBuf(const K :T512BitBuf;var Digest: T256BitDigest;const Buf:Str; const
       BufSize:Integer;const TotalSize:Int64);
22479:  CalcHMAC_SHA256( const Key, Buf : Ansistr) : T256BitDigest');
22480:  HMAC_SHA512Init(const Key:str;const KeySize:Int;var Digest:T512BitDigest;var K:T1024BitBuf);
22481:  HMAC_SHA512Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22482:  HMAC_SHA512FinalBuf(const K:T1024BitBuf;var Digest:T512BitDigest;const Buf:str;const
       BufSize:Integer;const TotalSize:Int64);
22483:  CalcHMAC_SHA512( const Key, Buf: Ansistr): T512BitDigest');
22484:  AHash(CL);
22485:  CL.AddTypeS('THashClass', 'class of AHash');
22486:  TChecksum32Hash(CL);
22487:   TXOR8Hash(CL);
22488:   TXOR16Hash(CL);
22489:   TXOR32Hash(CL);
22490:   TCRC16Hash(CL);
22491:   TCRC32Hash(CL);
22492:   TAdler32Hash(CL);
22493:   TELFHash(CL);
22494:   TMD5Hash(CL);
22495:   TSHA1Hash(CL);
22496:   TSHA256Hash2(CL);  //------------!!
22497:   TSHA512Hash(CL);
22498:   THMAC_MD5Hash(CL);
22499:   THMAC_SHA1Hash(CL);
22500:   THMAC_SHA256Hash(CL);
22501:   THMAC_SHA512Hash(CL);
22502:   THashType','(hashChecksum32,hashXOR8,hashXOR16,hashXOR32,hashCRC16,hashCRC32,hashAdler32, hashELF,
       hashMD5,hashSHA1 hashSHA256,hashSHA512,hashHMAC_MD5,hashHMAC_SHA1,hashHMAC_SHA256,hashHMAC_SHA512'+');
22503:  GetHashClassByType( const HashType : THashType) : THashClass');
22504:  GetDigestSize( const HashType : THashType) : Integer');
22505:  CalculateHash(const HashType:THashType;const Buf:AnsiStr;const Digest:str;const Key:AnsiStr;
22506:  HashString(const S: Ansistr; const Slots:LongWord; const CaseSensitive:Boolean):LongWord');
22507:  SelfTestHash');
22508: end;
22509:
22510: Unit UMatrix
22511: Proc Determinant( Dimen : integer; Data : TNmatrix; var Det : Float; var Error : byte);
22512: Proc Inverse2( Dimen : integer; Data : TNmatrix; var Inv : TNmatrix; var Error : byte);
22513:  Proc Gaussian_Elimination(Dimen:integer;Coefficients:TNmatrix;Constants:TNvector;var
       Solution:TNvector;var Error:byte);
```

```
22514:  Proc Partial_Pivoting(Dimen:integer;Coefficients:TNmatrix;Constants:TNvector;var Solution:TNvector;var
        Error:byte);
22515:  Proc LU_Decompose(Dimen:integer;Coefficients:TNmatrix;var Decomp:TNmatrix;var Permute:TNmatrix;var
        Error:byte);
22516:  Proc LU_Solve(Dimen:integer;var Decomp TNmatrix;Constants:TNvector;var Permute:TNmatrix;var
        Solution:TNvector;var Error:byte);
22517:  Proc Gauss_Seidel( Dimen: integer; Coefficients: TNmatrix; Constants: TNvector; Tol: Float; MaxIter:
        integer;var Solution:TNvector;var Iter:integer;var Error: byte);
22518: end;
22519:
22520: Proc SIRegister_uTPLb_RSA_Primitives(CL: TPSPascalCompiler);
22521: begin
22522:  //CL.AddDelphiFunction('Func I2OSP( x : THugeCardinal; xLen : integer; XStream : TStream; Pool :
        TMemoryStreamPool) :Bool');
22523:  //'Func OS2IP( XStream : TStream; xLen : integer; var x : THugeCardinal; Pool : TMemoryStreamPool;
        MaxBits : integer) :Bool');
22524:  Proc MGF1( mgfSeed : TStream; maskLen :Card; mask : TStream);
22525:   CL.AddTypeS('TLongOpResult', '( opPass, opFail, opAbort );
22526:  Func RSAES_OAEP_ENCRYPT( n, e : THugeCardinal; M, C : TMemoryStream) :Bool');
22527:  Func RSAES_OAEP_ENCRYPT_MaxByteLen( n : THugeCardinal) : integer');
22528:  Func RSAES_OAEP_DECRYPT( d, n : THugeCardinal; C, M : TStream) :Bool');
22529:  Func EMSA_PSS_ENCODE(M:TStream;emBits,
        sLen:integer;EM:TStream;CheckAbortFunc:TOnHashProgress):TLongOpResult;
22530:  Func RSASSA_PSS_SIGN(d,n:THugeCardinal;M,S:TStream;CheckAbortFunc:TOnHashProgress): TLongOpResult;
22531:  Func EMSA_PSS_VERIFY(M:TStream;emBits,sLen integer;EM:TStream;
        CheckAbortFunc:TOnHashProgress):TLongOpResult);
22532:  Func RSASSA_PSS_VERIFY(n,e THugeCardinal; M, S : TStream; CheckAbortFunc : TOnHashProgress) :
        TLongOpResult;
22533:  Func Generate_RSA_SymetricKey(n,e:THugeCardinal;CipherStream: TStream;const SymetricCipher:IBlockCipher)
        : TSymetricKey;
22534:  Func Extract_RSA_SymetricKey(d,n:THugeCardinal;CipherStream: TStream;const
        SymetricCipher:IBlockCipher):TSymetricKey;
22535: end;
22536:
22537: Proc SIRegister_DXUtil(CL: TPSPascalCompiler);
22538: begin
22539:  Proc SAFE_RELEASE( var i);
22540:  Proc SAFE_DELETE( var Obj);
22541:  Func DXUtil_GetDXSDKMediaPathCb( szDest : PChar; cbDest : Integer) : HRESULT');
22542:  Func DXUtil_FindMediaFileCch(strDestPath:PChar;cchDest:Integer;strFilename:PChar):HRESULT;
22543:  Func DXUtil_FindMediaFileCb(szDestPath: PChar; cbDest Integer;strFilename PChar):HRESULT'
22544:  Func DXUtil_WriteStringRegKey(hKey_ :HKEY; strRegName : PChar; strValue : PChar):HRESULT;
22545:   CL.AddTypeS('TIMER_COMMAND', 'DWORD');
22546:  'TIMER_RESET','LongInt').SetInt( 0);
22547:  'TIMER_START','LongInt').SetInt( 1);
22548:  'TIMER_STOP','LongInt').SetInt( 2);
22549:  'TIMER_ADVANCE','LongInt').SetInt( 3);
22550:  'TIMER_GETABSOLUTETIME','LongInt').SetInt( 4);
22551:  'TIMER_GETAPPTIME','LongInt').SetInt( 5);
22552:  'TIMER_GETELAPSEDTIME','LongInt').SetInt( 6);
22553:  Func DXUtil_Timer( command : TIMER_COMMAND) : Single');
22554:  Func DXUtil_ConvertGenericStringToAnsiCch(strDestination:PAnsiChar;const
        tstrSource:PChar;cchDestChar:Intr): HRESULT');
22555:  Func DXUtil_ConvertAnsistrToGenericCch(tstrDestination:PChar;const strSource:PAnsiChar; cchDestChar:Int):
        HRESULT');
22556:  Func DXUtil_ConvertGenericStringToAnsiCb(strDestination:PAnsiChar;const tstrSource:PChar;cbDestChar:Int:
        HRESULT');
22557:  Func DXUtil_ConvertAnsistrToGenericCb(tstrDestination:PChar;const
        strSource:PChar;cbDestChar:Integer):HRESULT;
22558:  CL.AddTypeS(TArrayListType', '( AL_VALUE, AL_REFERENCE );
22559:  SIRegister_CArrayList(CL);
22560:  Func GETTIMESTAMP : DWORD');
22561: end;
22562:
22563: Proc SIRegister_DCPbase64(CL: TPSPascalCompiler);
22564: begin
22565: Func Base64EncodeStr(const Value: Ansistr): Ansistr');
22566: Func Base64DecodeStr(const Value: Ansistr): Ansistr');
22567: Func Base64Encode( pInput :Str; pOutput :Str; Size : longint): longint;
22568: Func Base64Decode( pInput :Str; pOutput :Str; Size : longint): longint;
22569: end;
22570:
22571: Proc SIRegister_FlyFilesUtils(CL: TPSPascalCompiler);
22572: begin
22573:  Func GetCaseSensitiveFileName( const FileName :Str; RootPath :Str) :Str');
22574:  CL.AddConstantN('OTGDeivceCount','LongInt').SetInt( 16);
22575:  CL.AddConstantN('UsbDiskStartIndex','LongInt').SetInt( 255);
22576:  DeleteDirectories_WaitMinSecond','LongInt').SetInt( 2000);
22577:  Func isPathCanUseNow( const PathOrDir :Str; const Default :Bool) :Bool');
22578:  Func TestPathCanWrite( const PathOrDir :Str) :Bool');
22579:  Func GetSDCardPath( Index : Integer) :Str');
22580:  Func FindSDCardSubPath( SubPath :Str; Index : Integer) :Str');
22581:  Func GetAppPath :Str');
22582:  Func BuildFileListInAPath(const Path:str;const Attr:Int;const List:TStrings;JustFile:Bool):Bool;
22583:  Func BuildFileListInAPath1(const Path:str; const Attr:Integer;ustFile:Bool):Str;
22584:  Func GetFileNamesFromDirectory(const DirName:str;const SearchFilter:str;const FileAttribs:Int;const
        isIncludeSubDirName:Bool;const Recursion:Bool;const FullName:Bool):str;
22585:  Func DeleteDirectoryByEcho(const Source:str;AbortOnFailure:Bool;YesToAll:Bool;WaitMinSecond:Integer):Bool;
22586:  Func GetTotalSpaceSize( Path :Str) : UInt64');
```

```
22587:  Func GetAvailableSpaceSize( Path :Str) : UInt64');
22588:  Func GetFreeSpaceSize( Path :Str) : UInt64');
22589:  Func GetTotalMemorySize : UInt64');
22590:  Func GetFreeMemorySize : UInt64');
22591:  Func IsPadOrPC :Bool');
22592:  Func OpenFileOnExtApp(const FileName:str;Https:Bool):Bool');
22593:  Func NowGMT_UTC : TDateTime');
22594:  Func EncodeURLWithSchemeOrProtocol( const URL :Str):Str');
22595:  Func GetVolumePaths :Str');
22596:  Func GetExternalStoragePath :Str');
22597:  Func GetExterStoragePath :Str');
22598:  Func GetInnerStoragePath :Str');
22599:  Func GetIsExternalStorageRemovable :Bool');
22600:  Func GetScreenClientInches : Single');
22601:  Func IsCanFindJavaClass( const NamePath :Str) :Bool');
22602:  Func IsCanFindJavaMethod(const MethodName,Signature:str;const CalssNamePath:str):Bool;
22603:  Func IsCanFindJavaStaticMethod(const MethodName,Signature:str;const CalssNamePath:Str):Bool;
22604:   CL.AddTypeS(TGetFileNameLIsternerMethod','Procedure(const IsOK:Boolean;const FileName:str);
22605:  Func OpenFileDialog(Title,FileExtension:str; GetFileNameCallBack: TGetFileNameLIsternerMethod):Boolean;
22606:  Func CheckPermission( const APermissionName :Str) :Bool');
22607:  CL.AddConstantN('C_android_permission_EXTERNAL_STORAGE',
       'String').SetString('android.permission.WRITE_EXTERNAL_STORAGE');
22608:  Func CanWriteExterStorage :Bool');
22609:  Proc UpdateAlbum( FileNames :Str);
22610:  Func ReadNoSizeFileToString( const AFileName :Str) :Str');
22611:  Func ReadFileToString( const AFileName :Str) :Str');
22612: end;
22613:
22614: 466_kmean_cluster_correlation_test3.txt
22615:  Proc SIRegister_LatLonDist(CL: TPSPascalCompiler);
22616: begin
22617:  Func EllipticalDistance(llat1,llon1,llat2,llon2:extended;units:integer): extended');
22618:  Proc VDirectLatLon(DGLAT1,DGLON1,DFAZ,S EXTENDED;Units:integer;var DGLAT2,DGLON2,DBAZ : EXTENDED);
22619:  Func VInverseDistance(dlat1,dlon1,dlat2,dlon2:ext;var AzimuthInit,AzimuthFinal:ext;units:int):ext;
22620:  Func RhumbDistance(llat1,llon1,llat2,llon2:extended;units:integer;var Azimuth:extended):extended;
22621:  Proc RhumbLatLon(LAT1,LON1 Azimuth1,Dist:EXTENDED;Units:integer;var LAT2,LON2:EXTENDED);
22622:  Func ApproxEllipticalDistance(llat1,llon1,llat2,llon2: extended;units:integer):extended;
22623:  Func ApproxRhumbDistance(llat1,llon1,llat2,llon2: extended;units:integer; var Azimuth: extended):
       extended'
22624:  Proc ApproxRhumbLatLon(LAT1,LON1,Azimuth1,Dist:EXTENDED;Units integer;var LAT2,LON2: EXTENDED);
22625:  Func convertunits( fromindex,ToIndex:integer;fromvalue:extended):extended');
22626: end;
22627:
22628: Proc SIRegister_PXLTiming(CL: TPSPascalCompiler);
22629: begin
22630: CL.AddConstantN('OverrunDeltaLimit','Extended').setExtended( 8.0);
22631:  TTimerEvent', 'Proc ( const Sender : TObject);
22632:  SIRegister_TMultimediaTimer2(CL);
22633:  CL.AddTypeS(TSystemTimerValue, 'UInt64');
22634:  Func TimerValueInBetween(const Value1,Value2: TSystemTimerValue):TSystemTimerValue;
22635:  Func TickCountInBetween( const Value1, Value2:Card) :Card');
22636:  Func GetSystemTimerValue: TSystemTimerValue');
22637:  Func GetSystemTickCount:Card');
22638:  Func GetSystemTimeValue: Double');
22639:  Proc MicroSleep( const Microseconds : UInt64);
22640: end;
22641:
22642: (* === compile-time registration functions === *)
22643: (*----------------------------------------------------------------------------*)
22644: Proc SIRegister_TMultimediaTimer2(CL: TPSPascalCompiler);
22645: begin
22646: //with RegClassS(CL,'TOBJECT', 'TMultimediaTimer') do
22647: type  TMultimediaTimer2 = PXLTiming.TMultimediaTimer;
22648:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TMultimediaTimer2') do begin
22649:   Constructor Create');
22650:   Proc Process');
22651:   Proc Reset');
22652:   Proc NotifyTick( AllowSleep :Bool);
22653:   Delta', 'Double', iptr);
22654:   Latency', 'Double', iptr);
22655:   FrameRate', 'Integer', iptr);
22656:   Speed', 'Double', iptrw);
22657:   MaxFPS', 'Integer', iptrw);
22658:   Enabled', 'Boolean', iptrw);
22659:   SingleCallOnly', 'Boolean', iptrw);
22660:   OnTimer', 'TTimerEvent', iptrw);
22661:   OnProcess', 'TTimerEvent', iptrw);
22662:  end;
22663: end;
22664:
22665: Proc SIRegister_TNetCom(CL: TPSPascalCompiler);
22666: begin
22667:   //with RegClassS(CL,'TOBJECT', 'TNetCom') do
22668:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TNetCom') do begin
22669:   Constructor Create');
22670:   Proc Free');
22671:   Func Initialize :Bool');
22672:   Proc Finalize');
22673:   Func ResolveHost( const Host :Str) :Str');
```

```
22674:   Func ResolveIP( const IPAddress :Str) :Str');
22675:   Func SendStr(const Host:Str;const Port:Integer;const Data:str;const Size:Int):Bool;
22676:   Func Send2(const Host:str;const Port:Integer;const Data:str;const Size:Int):Bool;
22677:   Func Send(const Host:str;const Port:Integer;const Data:str;const Size:Int):Bool;
22678:   Proc Update');
22679:   Proc ResetStatistics');
22680:    LocalIP', 'String', iptr);
22681:    Initialized', 'Boolean', iptr);
22682:    Broadcast', 'Boolean', iptrw);
22683:    LocalPort', 'Integer', iptrw);
22684:    /('OnReceive', 'TReceiveEvent', iptrw);
22685:    UpdateRefreshTime', 'Integer', iptrw);
22686:    BytesReceived', 'Integer', iptr);
22687:    BytesSent', 'Integer', iptr);
22688:    SentPackets', 'Integer', iptr);
22689:    ReceivedPackets', 'Integer', iptr);
22690:    BytesPerSec', 'Integer', iptr);
22691:   end;
22692: end;
22693:
22694: (*----------------------------------------------------------------------------*)
22695: Proc SIRegister_PXLNetComs(CL: TPSPascalCompiler);
22696: begin
22697:   CL.AddConstantN('MaximumPacketSize','LongInt').SetInt( 8166);
22698:   CL.AddTypeS('TSocket', 'Integer');   //delphisocks as longint
22699:   CL.AddTypeS('TInAddr', 'record S_addr :Card; end');
22700:   SIRegister_TNetCom(CL);
22701: end;
22702:
22703: unit uPSI_cTCPConnection.pas
22704: Proc RIRegister_cTCPConnection_Routines(S: TPSExec);
22705: Func TCPGetTick : LongWord');
22706: Func TCPTickDelta( const D1, D2 : LongWord) : Integer');
22707: Func TCPTickDeltaW( const D1, D2 : LongWord) : LongWord');
22708:
22709: Proc SIRegister_cTCPConnection(CL: TPSPascalCompiler);
22710: begin
22711:   CL.AddClassN(CL.FindClass('TOBJECT'),'ETCPConnection');
22712:   CL.FindClass('TOBJECT'),'TTCPConnection');
22713:   CL.AddTypeS('TTCPLogType', '( tlDebug, tlInfo, tlError );
22714:   TTCPConnectionProxyState,(prsInit,prsNegotiating,prsFiltering,prsFinished,prsError,prsClosed);
22715:   SIRegister_TTCPConnectionProxy(CL);
22716:   SIRegister_TTCPConnectionProxyList(CL);
22717:   TTCPConnectionState','(cnsInit,cnsProxyNegotiation,cnsConnected,cnsClosed );
22718: TTCPConnectionTransferState,record LastUpdate:LongWord;ByteCount:Int64;TransferRate:LongWord; end;
22719:   TAnsiCharSet', 'set of AnsiChar');
22720:   TTCPConnectionNotifyEvent', 'Proc ( Sender : TTCPConnection);
22721:   TTCPConnectionStateChangeEvent', 'Proc ( Sender : TTCPConnection; State: TTCPConnectionState);
22722:  TTCPConnectionLogEvent','Procedure(Sender:TTCPConnection;LogType:TTCPLogType;LogMsg:str; LogLevel:Int;
22723:   SIRegister_TTCPConnection(CL);
22724:   //CL.AddTypeS('TTCPConnectionClass', 'class of TTCPConnection');
22725:  Func TCPGetTick : LongWord');
22726:  Func TCPTickDelta( const D1, D2 : LongWord) : Integer');
22727:  Func TCPTickDeltaW( const D1, D2 : LongWord) : LongWord');
22728: end;
22729:
22730: Proc RIRegister_cTCPConnection_Routines(S: TPSExec);
22731: begin
22732:  S.RegisterDelphiFunction(@TCPGetTick, 'TCPGetTick', cdRegister);
22733:  S.RegisterDelphiFunction(@TCPTickDelta, 'TCPTickDelta', cdRegister);
22734:  S.RegisterDelphiFunction(@TCPTickDeltaW, 'TCPTickDeltaW', cdRegister);
22735: end;
22736:
22737: var HTMLWindow2: IHTMLWindow2;
22738: const HTMLStrGoogle   'string').SetString(HTMLStrGoogle);     //javascript
22739:
22740: Proc GoogleMapsFormCreate(Sender:TObject;aform:TForm;webbrowser1:TWebbrowser;HTMLStr:str);
22741: Proc GoogleMapsGotoLocationClick(Sender: TObject; LatitudeText, LongitudeText:Str );
22742: begin
22743:   HTMLWindow2.execScript(Format('GotoLatLng(%s,%s)',[LatitudeText,LongitudeText]),'JavaScript');
22744: end;
22745:
22746: Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress:Str);
22747: var address :Str;
22748: begin
22749:   address:= vaddress;
22750:   address:=StringReplace(StringReplace(Trim(address),#13,' ',[rfReplaceAll]),#10,' ',[rfReplaceAll]);
22751:   HTMLWindow2.execScript(Format('codeAddress(%s)',[QuotedStr(address)]), 'JavaScript');
22752: end;
22753:  Func TaskbarHandle: THandle');
22754:  Func TrayHandle: THandle');
22755:  Proc DOSWipeFile(fn:Str);
22756:  Proc DOSFileWipe(fn:Str);
22757:  Func ExecAndWait(sExe, sCommandLine:Str):Bool;
22758:  Proc GoogleMapsFormCreate(Sender:TObject;aform:TForm; webbrowser1:TWebbrowser;HTMLStr,navdoc:str);
22759:  Proc GoogleMapsGotoLocationClick(Sender:TObject;LatitudeText,LongitudeText:Str );
22760:  Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress:Str);
22761:  Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress:Str);
22762:  Proc BitmapToMetafile(const Bmp: {Graphics.}TBitmap; const EMF: {Graphics.}TMetafile);
```

```
22763:
22764: Proc SIRegister_DrBobCGI(CL: TPSPascalCompiler);
22765: begin
22766:   CL.AddTypeS('TBobRequestMethod','(bobUnknown, bobGet, bobPost );
22767:  Func bobValue( const Field : ShortString; Convert :Bool) : ShortString');
22768:  Func CookieValue( const Field : ShortString) : ShortString');
22769:  Func getCGIEnvValues:Str;
22770: end;
22771:
22772: Func getCGIEnvValues:Str;
22773:   begin
22774:     result:= inttoStr(ContentLength)+';'+RemoteAddress+';'+HttpUserAgent+';'+
22775:     Authorization+';'+ScriptName+'.'
22776:   end;
22777:
22778: Proc SIRegister_OverbyteIcsCharsetUtils(CL: TPSPascalCompiler);
22779: begin
22780:  CL.AddConstantN('ERR_CP_NOTMAPPED','LongInt').SetInt( MAX_CODEPAGE + 1);
22781:  'ERR_CP_NOTAVAILABLE','LongInt').SetInt( MAX_CODEPAGE + 2);
22782:   ^'CP_US_ASCII','LongInt').SetInt( 20127);
22783:   CL.AddTypeS('CsuString', 'String');
22784:   CL.AddTypeS('TMimeCharset', '( CS_DEFAULT, CS_NOTMAPPED, UTF_8, WIN_1250, WIN'
22785:    +'_1251, WIN_1252, WIN_1253, WIN_1254, WIN_1255, WIN_1256, WIN_1257, WIN_125'
22786:    +'8, ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6,'
22787:    +' ISO_8859_7, ISO_8859_8, ISO_8859_8_i, ISO_8859_9, ISO_8859_13, ISO_8859_1'
22788:    +'5, ISO_2022_JP, ISO_2022_JP_1, ISO_2022_JP_2, ISO_2022_KR, ISO_2022_CN, X_'
22789:    +'CP50227, EUC_JP, GB_2312_80, GB_2312, HZ_GB_2312, GB_18030, EUC_CN, KOI8_R'
22790:    +', KOI8_U, UTF_16LE, UTF_16BE, UTF_7, SHIFT_JIS, BIG_5, KOREAN_HANGUL, EUC_'
22791:    +'KR, WIN_874, IBM_037, IBM_437, IBM_500, IBM_850, IBM_852, IBM_855, IBM_857'
22792:    +', IBM_00858, IBM_860, IBM_861, IBM_862, IBM_863, IBM_864, IBM_865, IBM_866'
22793:    +', IBM_869, IBM_870, IBM_1026, IBM_01047, IBM_01140, IBM_01141, IBM_01142, '
22794:    +'IBM_01143, IBM_01144, IBM_01145, IBM_01146, IBM_01147, IBM_01148, IBM_0114'
22795:    +'9, MACINTOSH, UTF_32LE, UTF_32BE, US_ASCII, T_61, CS_LAST_ITEM );
22796:   CL.AddTypeS('TMimeCharsets', 'set of TMimeCharset');
22797:   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');
22798:   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');
22799:    (*_cpinfoexA = record
22800:         MaxCharSize         : UINT;     { max length (bytes) of a char }
22801:         DefaultChar         : array[0..MAX_DEFAULTCHAR - 1] of Byte;    { default character }
22802:         LeadByte            : array[0..MAX_LEADBYTES - 1] of Byte;      { lead byte ranges }
22803:         UnicodeDefaultChar  : WideChar;
22804:         CodePage            : UINT;
22805:         CodePageName        : array[0..MAX_PATH] of AnsiChar;
22806:     end;   *)
22807: //CL.AddTypeS('PCharsetInfo', '^TCharsetInfo // will not work');
22808:   CL.AddTypeS(TCharsetInfo', 'record MimeCharset : TMimeCharset; CodePage : Lo'
22809:    +ngWord; MimeName : CsuString; FriendlyName :Str; end');
22810:   CL.AddTypeS('TCharsetInfos', 'array of TCharsetInfo');
22811:   //CL.AddTypeS('LPCPINFOEXA', '^CPINFOEXA // will not work');
22812:   CL.AddTypeS('CPINFOEXA', '_cpinfoexA');
22813:   CL.AddTypeS('TCpInfoExA', 'CPINFOEXA');
22814:   //CL.AddTypeS('PCpInfoExA', 'LPCPINFOEXA');
22815:   //CL.AddTypeS('LPCPINFOEXW', '^CPINFOEXW // will not work');
22816:   CL.AddTypeS('CPINFOEXW', '_cpinfoexW');
22817:   CL.AddTypeS('TCpInfoExW', 'CPINFOEXW');
22818:   //CL.AddTypeS('PCpInfoExW', 'LPCPINFOEXW');
22819:   //CL.AddTypeS('PCpInfoEx', 'PCpInfoExW');
22820:   CL.AddTypeS('CPINFOEX', 'CPINFOEXW');
22821:   CL.AddTypeS('TCpInfoEx', 'TCpInfoExW');
22822:   //CL.AddTypeS('LPCPINFOEX', 'LPCPINFOEXW');
22823:   //CL.AddTypeS('PCpInfoEx', 'PCpInfoExA');
22824:   CL.AddTypeS('CPINFOEX', 'CPINFOEXA');
22825:   CL.AddTypeS('TCpInfoEx', 'TCpInfoExA');
22826:   //CL.AddTypeS('LPCPINFOEX', 'LPCPINFOEXA');
22827:   SIRegister_TCodePageObj(CL);
22828:  CL.AddDelphiFunction('Func CodePageToMimeCharset( ACodePage : LongWord) : TMimeCharset');
22829:  Func CodePageToMimeCharsetString( ACodePage : LongWord) : CsuString');
22830:  Func GetMimeInfo0( AMimeCharSet : TMimeCharset) : PCharSetInfo;
22831:  //&&CL.AddDelphiFunc('Func GetMimeInfo1(const AMimeCharSetStr:CsuString):PCharSetInfo;
22832:  //CL.AddDelphiFunction('Func GetMimeInfo2( ACodePage : LongWord) : PCharSetInfo;
22833:  Func MimeCharsetToCharsetString( AMimeCharSet : TMimeCharset) : CsuString');
22834:  ('Func ExtractMimeName( PInfo : PCharSetInfo) : CsuString');
22835:  Func MimeCharsetToCodePage3( AMimeCharSet : TMimeCharset) : LongWord;
22836:  Func MimeCharsetToCodePage4(const AMimeCharSetStr:CsuString;out ACodePage:LongWord):Bool;
22837:  Func MimeCharsetToCodePageDef( const AMimeCharSetString : CsuString) : LongWord');
22838:  Func MimeCharsetToCodePageEx5(const AMimeCharSetStr:CsuString;out ACodePage:LongWord):Bool;
22839:  Func MimeCharsetToCodePageExDef( const AMimeCharSetString : CsuString) : LongWord');
22840:  Func IsValidAnsiCodePage( ACodePage : LongWord) :Bool');
22841:  Func IcsIsValidCodePageID( ACodePage : LongWord) :Bool');
22842:  Func IsSingleByteCodePage( ACodePage : LongWord) :Bool');
22843:  Proc GetSystemCodePageList( AOwnsObjectList : TObjectList);
22844:  Func AnsiCodePageFromLocale( ALcid : LCID) : LongWord');
22845:  Func OemCodePageFromLocale( ALcid : LCID) : LongWord');
22846:  Func GetThreadAnsiCodePage : LongWord');
22847:  Func GetThreadOemCodePage : LongWord');
```

```
22848:  Func GetUserDefaultAnsiCodePage : LongWord');
22849:  Func GetUserDefaultOemCodePage : LongWord');
22850:  Func GetCPInfoExA( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEXA) : BOOL');
22851:  Func GetCPInfoExW( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEXW) : BOOL');
22852:  Func GetCPInfoEx( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEX) : BOOL');
22853:  Proc GetFriendlyCharsetList(Items:TStrings;IncludeList: TMimeCharsets; ClearItems : Bool);
22854:  Proc GetMimeCharsetList( Items : TStrings; IncludeList: TMimeCharsets; ClearItems: Bool);
22855: end;
22856:
22857: Proc SIRegister_OverbyteIcsMimeUtils(CL: TPSPascalCompiler);
22858: begin
22859:  TMimeUtilsVersion','LongInt').SetInt( 802);
22860:  'mimuutilsCopyRight','String').SetString( ' MimeUtils (c) 2003-2014 F. Piette V8.02 ');
22861:  'SmtpDefaultLineLength','LongInt').SetInt( 76);
22862:  'SMTP_SND_BUF_SIZE','LongInt').SetInt( 2048);
22863:  'icsRegContentType','String').SetString( 'MIME\Database\Content Type');
22864:  //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)( '.' ));
22865:  //CL.AddConstantN('CrLfSet','TSysCharSet').SetString(ord(AnsiChar)(#13)or ord(AnsiChar)(#10));
22866:  //CL.AddConstantN('QuotedCharSet','TSysCharSet').SetString(ord(AnsiChar)('?' )or ord(AnsiChar)('=' )or
        ord(AnsiChar)' ' )or ord(AnsiChar)( '_' ));
22867:  //CL.AddConstantN('BreakCharsSet','TSysCharSet').SetString(ord(AnsiChar)( #9 ) or ord(AnsiChar) (#32 )or
        ord(AnsiChar)(';')or ord(AnsiChar)(',')or ord(AnsiChar)('>')or ord(AnsiChar)(']'));
22868:  Func EncodeQuotedPrintable0( const S : RawByteString) :Str;
22869:  Func EncodeQuotedPrintable1(const S:UnicodeString; ACodePage: LongWord) :UnicodeString;
22870:  Func EncodeQuotedPrintable2( const S : UnicodeString) : UnicodeString;
22871:  Func DecodeQuotedPrintable3( const S : RawByteString) : RawByteString;
22872:  Func DecodeQuotedPrintable4(const S : UnicodeString; ACodePage:LongWord):UnicodeString;
22873:  Func DecodeQuotedPrintable5( const S : UnicodeString) : UnicodeString;
22874:  Func SplitQuotedPrintableString( const S :Str) :Str');
22875:  Func FilenameToContentType( FileName :Str) :Str');
22876:  Func Base64Encode6( const Input : Ansistr) : Ansistr;
22877:  Func Base64Encode7( const Input : PAnsiChar; Len : Integer) : Ansistr;
22878:  Func Base64Encode8( const Input : UnicodeString; ACodePage : LongWord):UnicodeString;
22879:  Func Base64Encode9( const Input : UnicodeString) : UnicodeString;
22880:  Func Base64Encode10( Input :StrBuilder) :StrBuilder;
22881:  Func Base64Decode11( const Input: Ansistr) : Ansistr;
22882:  Func Base64Decode12( const Input: UnicodeString; ACodePage : LongWord):UnicodeString;
22883:  Func Base64Decode13( const Input: UnicodeString) : UnicodeString;
22884:  ('Func Base64Decode14( Input :StrBuilder) :StrBuilder;
22885:  Func InitFileEncBase64( const FileName :Str; ShareMode : Word) : TStream');
22886:  Func DoFileEncBase64( var Stream : TStream; var More :Bool) : Ansistr');
22887:  Func DoFileEncQuotedPrintable( var Stream : TStream; var More :Bool) : Ansistr');
22888:  Func DoTextFileReadNoEncoding( var Stream : TStream; var More :Bool) : Ansistr');
22889:  Func DoFileLoadNoEncoding( var Stream : TStream; var More :Bool) :Str');
22890:  Func Base64EncodeEx15(const Input:RawByteString;MaxCol:Int;var
        cPos:Integer;CodePage:LongWord;IsMultiByteCP:Bool):RawByteString;
22891:  Func Base64EncodeEx16(const Input:UnicodeString;MaxCol:Integer;var cPos:Int):UnicodeString;
22892:  Func Base64EncodeEx17(const Input:UnicodeString;MaxCol:Integer;var cPos:Integer;ACodePage:LongWord):
        UnicodeString;
22893:  Proc EndFileEncBase64( var Stream : TStream);
22894:  Proc DotEscape( var S :Str; OnlyAfterCrLf :Bool);
22895:  //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;
22896:  Func IcsWrapTextEx19(const Line:Str; const BreakStr :Str; const BreakingChars :
        TSysCharSet;MaxCol:Integer;QuoteChars:TSysCharSet;var cPos:Integer;ForceBreak Boolean):str;
22897:  Func UnFoldHdrLine( const S :Str) :Str');
22898:  Func NeedsEncoding20( const S : Ansistr) :Bool;
22899:  Func NeedsEncoding21( const S : UnicodeString) :Bool;
22900:  Func NeedsEncodingPChar( S : PChar) :Bool');
22901:  Func HdrEncodeInLine22(const Input: RawByteString; Specials : TSysCharSet; EncType : AnsiChar; const
        CharSet : Ansistr; MaxCol : Integer; DoFold :Bool; CodePage: LongWord; IsMultiByteCP :Bool) :
        RawByteString;
22902:  Func HdrEncodeInLine23( const Input : UnicodeString; Specials : TSysCharSet; EncType : WideChar; const
        CharSet : UnicodeString; MaxCol:Integer; DoFold:Bool; Codepage: LongWord; IsMultiByteCP :Bool) :
        UnicodeString;
22903:  Func HdrEncodeInLineEx( const Input : UnicodeString; Specials : TSysCharSet; EncType : WideChar;
        CodePage:LongWord;MaxCol:Integer;DoFold:Bool;IsMultiByteCP:Bool): RawByteString');
22904:  Func StrEncodeQP24( const Input:RawByteString; MaxCol:Integer;Specials:TSysCharSet; CodePage : LongWord;
        IsMultibyteCP:Bool) :Str;
22905:  Func StrEncodeQP25(const Input:UnicodeString;MaxCol:Integer;Specials:TSysCharSet;ACodePage:LongWord;
        IsMultibyteCP:Bool):UnicodeString;
22906:  Func StrEncodeQPEx26(const Buf:RawByteString; MaxCol:Integer; Specials:TSysCharSet; ShortSpace:Bool;var
        cPos:Int;DoFold:Bool;CodePage:LongWord;IsMultibyteCP:Bool):RawByteString;
22907:  Func StrEncodeQPEx27(const Buf:UnicodeString;MaxCol:Integer;Specials:TSysCharSet;ShortSpace:Boolean;var
        cPos: Integer;DoFold:Boolean):UnicodeString;
22908:  Proc FoldHdrLine28( HdrLines : TStrings; const HdrLine :Str);
22909:  Proc FoldHdrLine29(HdrLines:TStrings;const HdrLine:RawByteString;
        ACodePage:LongWord;IsMultiByteCP:Boolean);
22910:  Func FoldString30(const Input:RawByteString; BreakCharsSet:TSysCharSet; MaxCol:Integer;
        ACodePage:LongWord; IsMultiByteCP:Bool):RawByteString;
22911:  Func FoldString31(const Input:UnicodeString;BreakCharsSet:TSysCharSet;MaxCol:Int):UnicodeString;
22912:  Func CalcBase64AttachmentGrow( FileSize : Int64) : Int64');
22913:  //Func EncodeMbcsInline32(CodePage:LongWord;const
        Charset:str;EncType:Char;Body:PWideChar;Len:Integer;DoFold:Boolean;MaxLen  Integer): Ansistr;
22914:  //Func EncodeMbcsInline33(CodePage:LongWord;const
        Charset:str;EncType:Char;Body:PAnsiChar;Len:Integer;DoFold:Boolean;MaxLen: Integer): Ansistr;
```

```
22915:  Func ContentTypeGetExtn( const Content :Str; var CLSID :Str) :Str');
22916:  'Func ContentTypeFromExtn( const Extension :Str) :Str');
22917:   CL.AddTypeS('TMimeTypeSrc','( MTypeList, MTypeOS, MTypeMimeFile, MTypeKeyFile, MTypeRes );
22918:  SIRegister_TMimeTypesList(CL);
22919: end;
22920:
22921:  Returns the MIME name of a Windows code page identifier or an empty     }
22922: { string if the code page identifier is not mapped.                       }
22923: Func  CodePageToMimeCharsetString(ACodePage: LongWord): CsuString;
22924:
22925: Proc SIRegister_TMimeTypesList(CL: TPSPascalCompiler);
22926: begin
22927:   //with RegClassS(CL,'TComponent', 'TMimeTypesList') do
22928:   with CL.AddClassN(CL.FindClass('TComponent'),'TMimeTypesList') do begin
22929:     Constructor Create( AOwner : TComponent);
22930:     Proc Free');
22931:     Func LoadTypeList :Bool');
22932:     Proc Clear');
22933:     Func CountExtn : integer');
22934:     Func CountContent : integer');
22935:     Func AddContentType(const AExtn, AContent:Str) :Bool');
22936:     Func LoadWinReg :Bool');
22937:     Func LoadFromOS :Bool');
22938:     Func LoadFromList :Bool');
22939:     Func LoadMimeFile( const AFileName :Str) :Bool');
22940:     Func LoadFromResource( const AResName :Str) :Bool');
22941:     Func LoadFromFile( const AFileName :Str) :Bool');
22942:     Func SaveToFile( const AFileName :Str) :Bool');
22943:     Proc LoadContentTypes( AList : TStrings);
22944:     Proc AddContentTypes( AList : TStrings);
22945:     Proc GetContentTypes( AList : TStrings);
22946:     Func TypeFromExtn( const AExtn :Str) :Str');
22947:     Func TypeFromFile( const AFileName :Str) :Str');
22948:     Func TypeGetExtn( const AContent :Str) :Str');
22949:     RegisterProperty('LoadOSonDemand', 'boolean', iptrw);
22950:     RegisterProperty('MimeTypesFile', 'string', iptrw);
22951:     RegisterProperty('DefaultTypes', 'TStringList', iptrw);
22952:     RegisterProperty('MimeTypeSrc', 'TMimeTypeSrc', iptrw);
22953:     RegisterProperty('UnknownType', 'string', iptrw);
22954:   end;
22955: end;
22956:
22957: Proc SIRegister_uWebSocket(CL: TPSPascalCompiler);
22958: begin
22959:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWebSocketConnection');
22960:   CL.AddTypeS(TWebSocketMessageEvent','Proc (AConnection:TWebSocketConnection;const AMessage:str);
22961:   TWebSocketConnectEvent', 'Proc ( AConnection : TWebSocketConnection);
22962:   TWebSocketDisconnectEvent', 'Proc ( AConnection : TWebSocketConnection);
22963:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWebSocketException');
22964:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWebSocketHandshakeException');
22965:   SIRegister_TWebSocketRequest(CL);
22966:   SIRegister_TWebSocketConnection(CL);
22967:   SIRegister_TWebSocketServer(CL);
22968: end;
22969:
22970: (*----------------------------------------------------------------------------*)
22971: Proc SIRegister_OverbyteIcsUrl(CL: TPSPascalCompiler);
22972: begin
22973:  CL.AddConstantN('IcsUrlVersion','LongInt').SetInt( 800);
22974:  CL.AddConstantN('uriCopyRight','String').SetString(' TIcsURL (c) 1997-2012 F. Piette V8.00 ');
22975:  Proc icsParseURL( const URL :Str; var Proto, User, Pass, Host, Port, Path:Str);
22976:  Func icsPosn( const s, t :Str; count : Integer) : Integer');
22977:  Func icsUrlEncode( const S:str;DstCodePage:LongWord) :Str');
22978:  Func icsUrlDecode0(const S:str;SrcCodePage:LongWord;DetectUtf8:Boolean):str;
22979:  Func icsUrlDecode1(const S:RawByteString;SrcCodePage:LongWord;DetectUtf8:Bool):UnicodeString;
22980:  Func icsUrlEncodeToA( const S:Str; DstCodePage : LongWord) : Ansistr');
22981: end;
22982:
22983: (*----------------------------------------------------------------------------*)
22984: Proc SIRegister_OverbyteIcsLogger(CL: TPSPascalCompiler);
22985: begin
22986:  CL.AddConstantN('TIcsLoggerVersion','LongInt').SetInt( 800);
22987:   IcsCopyRight','String').SetString( ' IcsLogger (c) 2005-2012 by François PIETTE V8.00 ');
22988:   CL.AddClassN(CL.FindClass('TOBJECT'),'ELoggerException');
22989:   CL.AddTypeS('TLogOption',(loDestEvent,loDestFile,loDestOutDebug,loAddStamp,loWsockErr,loWsockInfo,
      loWsockDump,loSslErr,loSslInfo,loSslDump,loProtSpecErr,loProtSpecInfo, loProtSpecDump);
22990:   TLogOptions', 'set of TLogOption');
22991:   TLogFileOption', '( lfoAppend, lfoOverwrite );
22992:   TLogFileEncoding', '( lfeUtf8, lfeUtf16 );
22993:   CL.AddConstantN(LogAllOptErr','LongInt).Value.ts32:=ord(loWsockErr) or ord(loSslErr) or
      ord(loProtSpecErr);
22994:   TNTEventType', '( etError, etWarning, etInformation, etAuditSuccess, etAuditFailure );
22995:   TIcsLogEvent', 'Proc ( Sender : TObject; LogOption : TLogOption; const Msg :Str);
22996:   SIRegister_TIcsLogger(CL);
22997: end;
22998:
22999: Proc MoveCardinal2String(const Source:Card; SrcStartIdx: Integer; var Dest:Str;
23000:   DstStartIdx: Integer; Count: Integer);
23001: Proc MoveString2Cardinal(const Source:Str; SrcStartIdx: Integer; var Dest:Card;
```

```
23002:   DstStartIdx: Integer; Count: Integer);
23003: Proc MoveCardinal(const Source:Card; var Dest:Card; Count: Integer);
23004: Func ExplodeStr(S:Str; Delimiter: Char; Index: Integer):Str; //overload;
23005:   var I: Integer; P, P1: PChar;
23006: Func ImplodeStr(S:Str; Delimiter: Char):Str;
23007: Func ExplodeStr(S:Str; Delimiter: Char; Index: Integer):Str; //overload;
23008: Func InSetWord(n:word):boolean;
23009:
23010:      borland video frame
23011: <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>
23012:   resourcestring
23013:     { Charsets, user-friendly names, known from OE and IE }
23014:     sArabicISO                  = 'Arabic (ISO)'; //28596
23015:     sArabicWindows              = 'Arabic (Windows)'; //1256
23016:     sBalticISO                  = 'Baltic (ISO)'; //28594
23017:     sBalticWindows              = 'Baltic (Windows)'; //1257
23018:     sCentralEuropeanISO         = 'Central European (ISO)'; //28592
23019:     sCentralEuropeanWindows     = 'Central European (Windows)'; //1250
23020:     sChineseTraditionalBig5     = 'Chinese Traditional (Big5)'; //950
23021:     sChineseSimplifiedGB18030   = 'Chinese Simplified (GB18030)'; //54936
23022:     sChineseSimplifiedGB2312    = 'Chinese Simplified (GB2312)';//936
23023:     sChineseSimplifiedHZ        = 'Chinese Simplified (HZ)'; //52936
23024:     sCyrillicISO                = 'Cyrillic (ISO)'; //28595
23025:     sCyrillicKOI8R              = 'Cyrillic (KOI8-R)'; //20866
23026:     sCyrillicKOI8U              = 'Cyrillic (KOI8-U)'; //21866
23027:     sCyrillicWindows            = 'Cyrillic (Windows)'; //1251
23028:     sEstonianISO                = 'Estonian (ISO)'; //28603
23029:     sGreekISO                   = 'Greek (ISO)'; //28597
23030:     sGreekWindows               = 'Greek (Windows)'; //1253
23031:     sHebrewISOLogical           = 'Hebrew (ISO-Logical)'; //38598
23032:     sHebrewISOVisual            = 'Hebrew (ISO-Visual)'; //28598
23033:     sHebrewWindows              = 'Hebrew (Windows)'; //1255
23034:     sJapaneseJIS                = 'Japanese (JIS)'; //932
23035:     sKorean                     = 'Korean'; //949
23036:     sKoreanEUC                  = 'Korean (EUC)'; //51949
23037:     sLatin9                     = 'Latin 9 (ISO)'; //28605
23038:     sThaiWindows                = 'Thai (Windows)'; //874
23039:     sTurkishISO                 = 'Turkish (ISO)'; //28599
23040:     sTurkishWindows             = 'Turkish (Windows)'; //1254
23041:     sUnicodeUTF7                = 'Unicode (UTF-7)'; //65001
23042:     sUnicodeUTF8                = 'Unicode (UTF-8)'; //65000
23043:     sVietnameseWindows          = 'Vietnamese (Windows)'; //1258
23044:     sWesternEuropeanISO         = 'Western European (ISO)'; //28591
23045:     sWesternEuropeanWindows     = 'Western European (Windows)'; //1252
23046:
23047: Proc SIRegister_psULib(CL: TPSPascalCompiler); // Prometheus
23048: begin
23049:  'clMoneyGreenps','LongWord').SetUInt( TColor ( $00C0DCC0 ));
23050:  ('clMedGrayps','LongWord').SetUInt( TColor ( $00A4A0A0 ));
23051:  ('clRxCreamps','LongWord').SetUInt( TColor ( $00A6CAF0 ));
23052:  ('clCreamps','LongWord').SetUInt( TColor ( $00F0FBFF ));
23053:  ('clRxSkyBlueps','LongWord').SetUInt( TColor ( $00FFFBF0 ));
23054:  ('clSkyBlueps','LongWord').SetUInt( TColor ( $00F0CAA6 ));
23055:  ('clInfoBkps','LongWord').SetUInt( TColor ( $02E1FFFF ));
23056:  ('clNoneps','LongWord').SetUInt( TColor ( $02FFFFFF ));
23057:  ('clSystemColor','LongWord').SetUInt( $FF000000);
23058:  ('MinutesPerDay','LongInt').SetInt( 24 * 60);
23059:  ('DefaultBeepDelay','LongInt').SetInt( 500);
23060:  ('cDoNotBringToTop','String').SetString( '-LaunchInBackground');
23061:   CL.AddTypeS('TFilenameEvent', 'Proc ( Sender : TObject; var Filename :Str);
23062:  //CL.AddTypeS('TWriteDebugProc','Procedure(const LogName,RoutineName,Commen:str;DebugData:array of
       string);
23063:   CL.AddTypeS('TMaxIntSize', '( mbsInt8, mbsWord8, mbsInt16, mbsWord16, mbsInt32, mbsWord32 );
23064:  Proc WriteDebugps(const LogName,RoutineName,Comment:str;DebugData:array of string);
23065:  Func Betweenps( Value, BoundA, BoundB : integer; AllowFlip :Bool) :Bool');
23066:  Func BooleanMatchsp( Value1 :Bool; Value2 : integer) :Bool;
23067:  Func BooleanMatch1ps( Value1 : integer; Value2 :Bool) :Bool;
23068:  Func BooleanMatch2ps( Value1, Value2 : integer) :Bool;
23069:  Func IDMatchps( RequiredID, TestID : integer) :Bool;
23070:  Func IDMatch1ps( RequiredID : TField; TestID : integer) :Bool;
23071:  Func IDMatch2ps( RequiredID, TestID : TField) :Bool;
23072:  Func Blendps( Color1, Color2 : TColor; Weight1 : integer; Weight2 : integer) : TColor');
23073:  Func MergeRGBps( Red, Green, Blue : integer) : TColor');
23074:  Proc SplitRGBps( Color : TColor; var Red, Green, Blue : integer);
23075:  Func VisibleContrastps( BackgroundColor : TColor) : TColor');
23076:  Func SortedStrps( Value :Str) :Str');
23077:  Func IntSortStrps( const Value : integer) :Str');
23078:  //Func AllAssignedps( Values : array of pointer) :Bool');
23079:  Func GetTokenps( var SourceStr :Str; Delim :Str) :Str');
23080:  Func URLizeps( SourceStr :Str) :Str');
23081:  Func UnURLizeps( const SourceStr :Str) :Str');
23082:  Func UnQuoteStrps( const Value :Str; QuoteChar : char) :Str');
23083:  Proc SetAndSaveBoolps( var OldVar :Bool; Value :Bool; var SaveVar :Bool);
23084:  Proc SetAndSaveIntps( var OldVar : integer; Value : integer; var SaveVar : integer);
23085:  Proc SetAndSaveStrps( var OldVar :Str; const Value :Str; var SaveVar :Str);
23086:  Func GetCharFromVKeyps( VKey : word) :Str');
23087:  Func IsControlKeyDownps :Bool');
23088:  Proc PushScreenCursorps( const aCursor : TCursor);
```

```
23089:  Proc PopScreenCursorps');
23090:  Func PeekScreenCursorps : TCursor');
23091:  Func CharIsNumericSymbolps( aChar : char) :Bool');
23092:  Func PrecisionMultiplierps( DecimalPrecision : byte) :Card');
23093:  Func TextToFloatValps(const aText:str;const DecimalPrecision:byte;const DefaultValue:extended):extended;
23094: Func TextToWordValps(const aText:str;const DefaultVal:cardi;const MaxIntSize:TMaxIntSize):cardi;
23095: Func TextToIntValps(const aText:str;const DefaultVal:int;const MaxIntSize:TMaxIntSize):int;
23096:  Func BankersRoundingps( const Num : currency; const DecimalPrecision : byte) : currency');
23097:  Func RoundDownps( const Num : Extended; const DecimalPrecision : byte) : Extended');
23098:  Func RoundUpps( const Num : Extended; const DecimalPrecision : byte) : Extended');
23099:  Func RoundNearestps( const Num : Extended; const DecimalPrecision : byte) : Extended');
23100:  Func NumDigitsps( const aNum : integer) : integer');
23101:  Func ZeroToMaxps( Value : integer) : integer');
23102:  Proc CopyIfPrefixMatchps(Source,Dest:TStrings;Prefix:str;ClearDest:bool;ValuesOnly:bool);
23103:  Func HexToCharps( const aValue : byte) : AnsiChar');
23104:  Func CharToHexps( const aValue : AnsiChar) : byte');
23105:  Func HexToTextps( const Source : Ansistr) : Ansistr');
23106:  Func TextToHexps( const Source : Ansistr) : Ansistr');
23107:  Proc SaveGraphicToStreamps( aGraphic : TGraphic; aStream : TStream);
23108:  Func LoadGraphicFromStreamps( aStream : TStream) : TGraphic');
23109:  Func GraphicToTextps( aGraphic : TGraphic) :Str');
23110:  Func TextToGraphicps( aGraphicAsText :Str) : TGraphic');
23111:  Func GetCustNameStrps( aFirstName, aLastName :Str) :Str');
23112:  Func GetLongCustNameStrps( aTitle, aFirstName, aLastName :Str) :Str');
23113:  Func GetPriceFromMarginps( const aMargin :Str; aCost : currency): currency');
23114:  Func GetMarginFromPriceps( aPrice, aCost : currency) :Str');
23115:  Func GetDriveSerialNumps( Path :Str) : longint');
23116:  Func GetCCCardTypeFromNameps( aCardName :Str) : integer');
23117:  Func GetItemMarginPercentps( Price, Cost : Currency) : double');
23118:  Func ValidateCreditCardps( aCardNum :Str) :Bool');
23119:  Func ValidateCreditCardRangeps( const aCardNum, aLowNum, aHighNum :Str) :Bool');
23120:  Func HHMMToTimeps( Value :Str) : TTime');
23121:  Func MMDDToDateps( Value :Str) : TDate');
23122:  Func TimeToHHMMSSps( Value : TTime) :Str');
23123:  Func DateToYYYYMMDDps( Value : TDate) :Str');
23124:  Func FontColorToStringps( SourceFont : TFont; SourceColor : TColor) :Str');
23125:  Proc StringToFontps( aStr :Str; DestFont : TFont);
23126:  Proc StringToFontColorps( aStr :Str; DestFont : TFont; var DestColor : TColor);
23127:  Proc ShowDebugStringsps( DebugStrings : TStrings);
23128:  Proc ShowStackDumpps');
23129:  Proc RegisterFileExtensionps(const Extension,Description,DefaultActionName,DefaultActionDescription,
       DefaultActionCommand:str; IconIndex:integer);
23130:  Func LaunchCommandps(aCommand:str; aParams:str; WaitForTerminate:boolean):bool;
23131:  Func GetWindowsDirectoryStrps :Str');
23132:  Func IsRemoteSessionps :Bool');
23133:  Func IsWinXPps :Bool');
23134:  Proc AssignTextFileToStreamps( var ATextFile : TextFile; AStream : TStream);
23135:  Func DeleteFileMaskps( const Mask :Str) :Bool');
23136:  Func FileMaskCountps( const Mask :Str) : integer');
23137:  Func FileMaskExistsps( const Mask :Str) :Bool');
23138:  Func FileMaskListps( const Mask :Str) :Str;
23139:  Func FileMaskList1ps(const Mask:str; Dest : TStrings; ClearList :Bool) :Bool;
23140:  Func GetSafeNumericVariantps( Value : Variant) : Variant');
23141:  Func GetPasswordCharStrps( aStr :Str) :Str');
23142:  Func GetDriversLicenseHeightDisplayTextps( aStr :Str) :Str');
23143:  Func IncludePathDelimiterps( const aFilePath :Str) :Str');
23144:  Func ExcludePathDelimiterps( const aFilePath :Str) :Str');
23145:  //Func GetEnvironmentVariable( const Name :Str) :Str');
23146:  Func ArrayToStrps( SourceArray : array of string; Separator : AnsiChar) :Str');
23147:  Func psStrReplaceps( var S : Ansistr; const Search, Replace : Ansistr; Flags : TReplaceFlags):boolean;
23148:  Func StrReplaceInStringsps(StrObject:TStrings;const Search,Replace:Ansistr;Flags:TReplaceFlags):bool;
23149:  Proc ShowMessageps( aMsg :Str);
23150:  Proc ShowErrorMessageps( aMsg :Str);
23151:  Func StandardUserQueryps(aMsg:str;DialogType:Card;DialogCaption:Str):Card;
23152:  Func ParamValueps( ParamName :Str) :Str');
23153:  Func ParamFlagSetps( ParamName :Str) :Bool');
23154:  Func ConcatenateNonBlanksps( Strings : array of string) :Str');
23155:  Func ConcatenateWithDelimiterps( Strings : array of string; Delimiter : PChar) :Str');
23156:  Func GetMaxStringLengthps( Strings : array of string) : integer');
23157:  Proc PlayBeepps( BeepActionType : TMsgDlgType; NumBeeps : integer; MSDelay : integer);
23158:  Func DelTreeps( const DirectoryName :Str) :Bool');
23159:  Proc UpdateActionCaptionps( anAction : TAction; aCaption :Str);
23160:  Proc GetDirectoryListps( const DirectoryName :Str; StringList : TStrings);
23161:  Func DecodePathps( const Path :Str; PathVars : TStrings) :Str');
23162:  Func KillEXEps( ExeName :Str) :Bool');
23163:  Func GetWindowsDefaultPrinterNameps :Str');
23164:  Func GetBackupFilenameps( const OrigFilename :Str) :Str');
23165:  Func GetRestoreFilenameps( const BackupFilename :Str) :Str');
23166:  Func MakeBackupFileps( const OrigFilename :Str; OverwriteExisting :Bool):Str');
23167:  Func RestoreBackupFileps( const BackupFilename:str;OverwriteExisting:boolean):Str');
23168:  Func CalcDistanceps( Lat1, Long1, Lat2, Long2 : double) : double');
23169:  Func GetDelimiterCountps( aString, aDelimiter :Str) : integer');   //}
23170: end;
23171:
23172: (*----------------------------------------------------------------------------*)
23173: Proc SIRegister_psUFinancial(CL: TPSPascalCompiler);
23174: begin
23175:  CL.AddTypeS(TLoanType', '( ltAmortized, ltSimple );
23176:  CL.AddTypeS(TLoanPayType','( lptMonthly, lptBiMonthly, lptBiWeekly );
```

```
23177: TypeS(TLoanPaymentEvent','Procedure(PayDate:TDate;PayNum:int;PaymentAmt,Balance,InterestAmt,
       TotalInterestAmt,AmortizedAmt,TotalAmortizedAmt:currency;var ExtraPayment: currency);
23178:  Func
       LoanPaymentAmt(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;LoanType
       :TLoanType):currency');
23179:  Func LoanNumPayments(Principal,
       PaymentAmt:currency;InterestRate:extended;PayType:TLoanPayType;LoanType:TLoanType):int;
23180:  Func LoanTotalEstimate(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;
       LoanType:TLoanType):currency;
23181:  Func
       LoanInterestEstimate(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;
       LoanType:TLoanType):currency;
23182:  Proc AmortizationSchedule( OnLoanPayment:TLoanPaymentEvent;StartDate:TDate;Principal
       :currency;InterestRate:extended;TotalNumPay:integer;PayType TLoanPayType;LoanType:TLoanType);
23183: Proc AmortizationSchedulePaymentAmt(OnLoanPayment TLoanPaymentEvent;StartDate:TDate; Principal,
       PaymentAmt:currency;InterestRate:extended;TotalNumPay:int;PayType:TLoanPayType; LoanType:TLoanType);
23184: end;
23185:
23186: (*----------------------------------------------------------------------------*)
23187: Proc SIRegister_TSimpleRSS(CL: TPSPascalCompiler);
23188: begin
23189:   //with RegClassS(CL,'TComponent', 'TSimpleRSS') do
23190: with CL.AddClassN(CL.FindClass('TComponent'),'TSimpleRSS') do begin
23191:   Constructor Create( AOwner : TComponent);
23192:   Proc Free');
23193:   Proc SaveToFile( Filename :Str);
23194:   Func SaveToString :Str');
23195:   Proc SaveToStream( Stream : TStream);
23196:   Func SaveToStrings : TStrings;
23197:   Proc LoadFromHTTP( URL :Str)
23198:   Proc LoadFromString( S :Str);
23199:   Proc LoadFromFile( Filename :Str);
23200:   Proc LoadFromStream( Stream : TStream);
23201:   Proc LoadFromStrings( Strings : TStrings);
23202:   Proc GenerateXML;
23203:   Proc GenerateXML1( FeedType : TXMLTypeRSS);
23204:   Proc GenerateComponent');
23205:   Proc ClearXML');
23206:   Property('Channel', 'TRSSChannel', iptrw);
23207:   Property('Items', 'TRSSItems', iptrw);
23208:   Property('Version', 'string', iptrw);
23209:   Property('XMLType', 'TXMLTypeRSS', iptrw);
23210:   Property('XMLFile', 'TXMLDocument', iptr);
23211:   Property('IndyHTTP', 'TIdHTTP', iptrw);
23212:   Property('SimpleRSSVersion', 'String', iptr);
23213:   Property('OnCreate', 'TNotifyEvent', iptrw);
23214:   Property('OnGenerateXML', 'TNotifyEvent', iptrw);
23215:   Property('OnParseXML', 'TNotifyEvent', iptrw);
23216:  end;
23217: end;
23218:
23219: (*----------------------------------------------------------------------------*)
23220: Proc SIRegister_SimpleRSSTypes(CL: TPSPascalCompiler);
23221: begin
23222: CL.AddClassN(CL.FindClass('TOBJECT'),'ESimpleRSSException');
23223: CL.AddTypeS('TLanguagesRSS', '( langAF, langSQ, langEU, langBE, langBG, langCA, '
23224:  +'langZH_CN, langZH_TW, langHR, langCS, langDA, langNL, langNL_BE, langNL_NL'
23225:  +',langEN, langEN_AU, langEN_BZ, langEN_CA, langEN_IE, langEN_JM, langEN_NZ'
23226:  +',langEN_PH, langEN_ZA, langEN_TT, langEN_GB, langEN_US, langEN_ZW, langET'
23227:  +',langFO, langFI, langFR, langFR_BE, langFR_CA, langFR_FR, langFR_LU, lang'
23228:  +'FR_MC, langFR_CH, langGL, langGD, langDE, langDE_AT, langDE_DE, langDE_LI,'
23229:  +' langDE_LU, langDE_CH, langEL, langHAW, langHU, langIS, langIN, langGA, la'
23230:  +'ngIT, langIT_IT, langIT_CH, langJA, langKO, langMK, langNO, langPL, langPT'
23231:  +', langPT_BR, langPT_PT, langRO, langRO_MO, langRO_RP, langRU, langRU_MO, l'
23232:  +'angRU_RU, langSR, langSK, langSL, langES, langES_AR, langES_BO, langES_CL,'
23233:  +' langES_CO, langES_CR, langES_DO, langES_EC, langES_SV, langES_GT, langES_'
23234:  +'HN, langES_MX, langES_NI, langES_PA, langES_PY, langES_PE, langES_PR, lang'
23235:  +'ES_ES, langES_UY, langES_VE, langSV, langSV_FI, langSV_SE, langTR, langUK,langX );
23236: CL.AddTypeS('TXMLTypeRSS', '( xtRDFrss, xtRSSrss, xtAtomrss, xtiTunesrss );
23237: TContentTypeRSS', '( ctTextrss, ctHTMLrss, ctXHTMLrss );
23238: TEncodingTypeRSS', '( etBase64rss, etEscapedrss, etXMLrss );
23239: SIRegister_TRFC822DateTime(CL);
23240: SIRegister_TRSSChannelReq(CL);
23241: SIRegister_TRSSImageReq(CL);
23242: SIRegister_TRSSImageOpt(CL);
23243: SIRegister_TRSSImage(CL);
23244: SIRegister_TRSSCloud(CL);
23245: SIRegister_TRSSTextInput(CL);
23246: SIRegister_TRSSChannelCategory(CL);
23247: SIRegister_TRSSChannelCategories(CL);
23248: SIRegister_TRSSChannelSkipHours(CL);
23249: SIRegister_TRSSChannelSkipDays(CL);
23250: SIRegister_TRSSChannelOpt(CL);
23251: SIRegister_TRSSChannel(CL);
23252: SIRegister_TRSSItemSource(CL);
23253: SIRegister_TRSSItemEnclosure(CL);
23254: SIRegister_TRSSItemCategory(CL);
23255: SIRegister_TRSSItemCategories(CL);
23256: SIRegister_TRSSItemGUID(CL);
```

```
23257: SIRegister_TRSSAuthor(CL);
23258: SIRegister_TRSSItem(CL);
23259: SIRegister_TRSSItems(CL);
23260: CL.AddTypeS(TFormatSettingsRSS', 'record ShortDateFormat :Str; LongDateForm'
23261:       +at :Str; LongTimeFormat :Str; ShortTimeFormat :Str; end');
23262: end;
23263:
23264: RSS Feed Snippet:
23265: const RSS_NewsFeed = 'http://feeds.bbci.co.uk/news/world/rss.xml';
23266:  with TSimpleRSS.create(self) do begin
23267:    XMLType:= xtRDFrss;
23268:    IndyHTTP:= TIdHTTP.create(self);
23269:    LoadFromHTTP(RSS_NewsFeed);
23270:    //LoadFromHTTP(Climatefeed);
23271:    writeln('RSSVersion: '+Version)
23272:    writeln('SimpleRSSVersion: '+SimpleRSSVersion)
23273:    for it:= 0 to items.count-1 do
23274:      writeln(itoa(it)+': '+Items[it].title+': '+items[it].pubdate.getdatetime);
23275:  end;
23276:
23277: (*----------------------------------------------------------------------------*)
23278: Proc SIRegister_JTools(CL: TPSPascalCompiler);
23279: begin
23280:  Func ParseCmdLine( ACmdLine : PChar; List : TStrings; QuoteChar : char) :Bool');
23281: Func GetCmdSwitchValue(const Switch:str;SwitchChars:TSysCharSet;var Value:str;IgnoreCase:boolean):bool;
23282:  //Proc ShowError( const S :Str; const Args : array of const);
23283: end;
23284:
23285: Proc SIRegister_rfc1213util(CL: TPSPascalCompiler);
23286: begin
23287:  CL.AddConstantN('rfc1213version','String').SetString( '0.1.2');
23288:  Func UptimeToDays( const Uptime :Str) :Str');
23289:  Func getSNMP( const mib, community, host :Str; var ErrNo : Integer) :Str');
23290:  Func sysServicesString( const sysServicesStr :Str) :Str');
23291:  Func ipForwardingString( const ipForwarding :Str) :Str');
23292:  Func ipRouteTypeString( const ipRouteType :Str) :Str');
23293:  Func ipRouteProtoString( const ipRouteProto :Str) :Str');
23294:  Func ipNetToMediaTypeString( const ipNetToMediaType :Str) :Str');
23295:  Func GetColFromTableRow( const EntryCols : TStringList; index : integer) :Str');
23296:  Func ifTypeString( ifTypeStringNo :Str) :Str;
23297:  Func ifStatusString( ifStatusStringNo :Str) :Str;
23298:  Func tcpRtoAlgorithmString( const tcpRtoAlgorithm :Str) :Str;
23299:  Func tcpConnStateString( const tcpConnState :Str) :Str;
23300:  Func egpNeighStateString( const egpNeighState :Str) :Str;
23301:  Func egpNeighModeString( const egpNeighMode :Str) :Str;
23302:  Func egpNeighEventTriggerString( const egpNeighEventTrigger :Str) :Str;
23303:  Func snmpEnableAuthenTrapsString( const snmpEnableAuthenTraps :Str) :Str;
23304:  Func FormatMac( const Mac :Str) :Str;
23305:  Proc JumpToRegistryKey(Key:Str);
23306: end;
23307:
23308: Proc SIRegister_neuralab(CL: TPSPascalCompiler);
23309: begin
23310:  SIRegister_TABHash(CL);
23311:  Func ABKey( S : array of byte; Divisor : longint) : longint');
23312:  Func ABCmp( var X, Y : array of byte) :Bool');
23313:  Func ABGetNext1( var AB : array of byte; ST : word) : word');
23314:  Func ABCountDif( var X, Y : array of byte) : longint');
23315:  Func ABCountDifZero( var X : array of byte) : longint');
23316:  Proc ABAnd( var A, B : array of byte);
23317:  Func ABGetEqual( var Equal, X, Y : array of byte) : longint');
23318:  Proc ABShiftLogicalLeft( var X : array of byte);
23319:  Proc ABShiftLogicalRight( var X : array of byte);
23320:  Func ABGetDif( var Dif, X, Y : array of byte) : longint');
23321:  Func ABToString( var AB : array of byte) :Str');
23322:  Func ABToStringR( var AB : array of byte) :Str');
23323:  Proc ABClear( var AB : array of byte);
23324:  Proc ABFull( var AB : array of byte);
23325:  Proc ABBitOnPos( var AB : array of byte; POS : longint);
23326:  Proc ABBitOnPosAtPos( var AB : array of byte; X, Start, Len : longint);
23327:  Func ABReadBitOnPosAtPos( var AB : array of single; Start, Len:longint): longint');
23328:  Proc ABCopy( var A, B : array of byte);
23329:  Proc ABTriPascal( var A, B : array of byte);
23330:  Proc ABSet( var A : array of byte; B : array of byte);
23331: end;
23332:
23333: Proc SIRegister_neuralbit(CL: TPSPascalCompiler);
23334: begin
23335:   //CL.AddTypeS('TArrOf2BytesPtr', '^TArrOf2Bytes // will not work');
23336:   //CL.AddTypeS('TArrOf3BytesPtr', '^TArrOf3Bytes // will not work');
23337:   //CL.AddTypeS('TArrOf4BytesPtr', '^TArrOf4Bytes // will not work');
23338:   //CL.AddTypeS('TArrBytePtr', '^TLongByteArray // will not work');
23339:  Func POT( numero, elevado : extended) : extended');
23340:  Func LongintBitTest( Data : longint; P : longint) :Bool');
23341:  Func LongintBitFlip( Data : longint; P : longint) : longint');
23342:  Proc BAClear( var VARS : array of byte);
23343:  Proc BAMake1( var VARS : array of byte);
23344:  Func BARead( var A : array of byte; P : longint) : byte');
23345:  Proc BAFlip( var A : array of byte; P : longint);
```

```
23346:  Proc BAWrite( var A : array of byte; P : longint; Data : byte);
23347:  Func BATest( var A : array of byte; P : longint) :Bool');
23348:  Proc BASum( var x, y : array of byte);
23349:  Proc BASub( var x, y : array of byte);
23350:  Proc BAIncPos( var x : array of byte; POS : longint);
23351:  Proc BADecPos( var x : array of byte; POS : longint);
23352:  Proc BAInc( var x : array of byte);
23353:  Proc BADec( var x : array of byte);
23354:  Func BAToString( VARS : array of byte) :Str');
23355:  Func BAToFloat( var VARS : array of byte) : extended');
23356:  Proc PFloatToBA( var VARS : array of byte; Valor : extended);
23357:  Proc BANot( var VARS : array of byte);
23358:  Proc BAAnd( var r, x, y : array of byte);
23359:  Proc BAOr( var r, x, y : array of byte);
23360:  Proc BAXOr( var r, x, y : array of byte);
23361:  Func BAGrater( var x, y : array of byte) :Bool;
23362:  Func BALower( var x, y : array of byte) :Bool;
23363:  Func BAEqual( var x, y : array of byte) :Bool;
23364:  Proc BAPMul( var r, x, y : array of byte);
23365:  Func nnRAnd( A, B : extended) : extended;
23366:  Func nnROr( A, B : extended) : extended;
23367:  Func nnRNot( A : extended) : extended;
23368:  Func nnRXor( A, B : extended) : extended;
23369:  Func REqual( A, B : extended) : extended;
23370:  Proc RSum( x, y, z : extended; var R, C : extended);
23371:  Proc RegSum( var x, y : array of extended);
23372:  Func RegEqual( var x, y : array of extended) : extended');
23373:  Func RegOrdEqual( var x, y : array of extended) : extended');
23374:  Func RegToString( var VARS : array of extended) :Str');
23375:  Func ROrer( var VARS : array of extended) : extended');
23376:  Func RAnder( var VARS : array of extended) : extended');
23377:  Func RCNot( X : extended; var VARS : array of extended) : extended');
23378:  Func ROrMaxTerm( var VARS : array of extended; NumMaxTerm : longint) : extended');
23379:  Func ROrMaxTermStr( NumVars : longint; NumMaxTerm : longint) :Str');
23380:  Func RSatFunc( var VARS : array of extended; NumFunc : longint) : extended');
23381:  Func RSatFuncStr( NumVars : longint; NumFunc : longint) :Str');
23382:  Proc RRegen( var VARS : array of extended);
23383:  Proc RDegen( var VARS : array of extended);
23384:  Proc RDegenP( var VARS : array of extended; P : extended);
23385:  Proc nnClear( var VARS : array of extended);
23386:  Proc BARAnd( var R, A, B : array of byte);
23387:  Proc BAROr( var R, AUX, A, B : array of byte);
23388:  Proc BARNot( var R, A : array of byte);
23389: end;
23390:
23391: (*----------------------------------------------------------------------------*)
23392: Proc SIRegister_winsvc2(CL: TPSPascalCompiler);
23393: begin
23394:  CL.AddConstantN('SERVICE_AUTO_START','LongWord').SetUInt( $00000002);
23395:  CL.AddConstantN('SERVICE_CONFIG_DELAYED_AUTO_START_INFO','LongInt').SetInt( 3);
23396:  ('SERVICE_CONFIG_FAILURE_ACTIONS_FLAG','LongInt').SetInt( 4);
23397:  ('SERVICE_CONFIG_PREFERRED_NODE','LongInt').SetInt( 9);
23398:  ('SERVICE_CONFIG_PRESHUTDOWN_INFO','LongInt').SetInt( 7);
23399:  ('SERVICE_CONFIG_REQUIRED_PRIVILEGES_INFO','LongInt').SetInt( 6);
23400:  ('SERVICE_CONFIG_SERVICE_SID_INFO','LongInt').SetInt( 5);
23401:  ('SERVICE_CONFIG_TRIGGER_INFO','LongInt').SetInt( 8);
23402:  ('SC_MANAGER_ALL_ACCESS','LongWord').SetUInt($F003F);
23403:  ('SERVICE_ALL_ACCESS','LongWord').SetUInt($F01FF);
23404:  Func ChangeServiceType( ServiceName :Str; TypeID : DWord) :Bool');
23405:  Func GetServiceStatus2( ServiceName :Str; ErrorState :Bool) :Bool');
23406:  Func StartService2( ServiceName :Str) :Bool');
23407:  Func StopService2( ServiceName :Str) :Bool');
23408: end;
23409:
23410: Proc SIRegister_neuralbyteprediction(CL: TPSPascalCompiler);
23411: begin
23412:   CL.AddTypeS('TNeuralCountings', 'array of longint');
23413:   SIRegister_TNeuronGroupBase(CL);
23414:   SIRegister_TNeuronGroup(CL);
23415:   CL.AddTypeS('TNeuralNetwork', 'array of TNeuronGroup');
23416:   SIRegister_TStatePredictionClass(CL);
23417:   SIRegister_TLabeledState(CL);
23418:   SIRegister_TClassifier(CL);
23419:   SIRegister_TEasyLearnAndPredictClass(CL);
23420: end;
23421:
23422: Proc SIRegister_neuralcache(CL: TPSPascalCompiler);
23423: begin
23424:  CL.AddConstantN('NeuralMaxStates','LongInt').SetInt( 400000);
23425:  CL.AddTypeS('TNeuralState', 'TBytes');
23426:  TProcPred', 'Proc ( var ST : TBytes; Acao : byte);
23427:  SIRegister_TCacheMem(CL);
23428: end;
23429:
23430: (*----------------------------------------------------------------------------*)
23431: Proc SIRegister_USolarSystem(CL: TPSPascalCompiler);
23432: begin
23433:  CL.AddTypeS('TOrbitalElements',record N:Double;i:Double;w:Double;a:Double;e:Double;M:Double; end;
23434:   TOrbitalElementsData', 'record NConst : Double; NVar : Double; i'
```

```
23435:    +Const : Double; iVar : Double; wConst : Double; wVar : Double; aConst : Do'
23436:    +uble; aVar : Double; eConst : Double; eVar : Double; MConst : Double; MVar:Double; end');
23437:  CL.AddConstantN('cAUToKilometers','Extended').setExtended( 149.6e6);
23438:  CL.AddConstantN('cEarthRadius','LongInt').SetInt( 6400);
23439:  Func GMTDateTimeToJulianDay( const dt : TDateTime) : Double');
23440:  Func ComputeOrbitalElements(const oeData:TOrbitalElementsData;const d:Double): TOrbitalElements;
23441:  Func ComputePlanetPosition( const orbitalElements : TOrbitalElements) : TAffineVector;
23442:  Func ComputePlanetPosition1(const orbitalElementsData:TOrbitalElementsData;const d:Double):TAffineVector;;
23443: end;
23444:
23445: Proc SIRegister_TSearchAnagrams(CL: TPSPascalCompiler);
23446: begin
23447:   //with RegClassS(CL,'tcontrol', 'TSearchAnagrams') do
23448:  with CL.AddClassN(CL.FindClass('tcontrol'),'TSearchAnagrams') do begin
23449:    RegisterProperty('useabbrevs', 'boolean', iptrw);
23450:    RegisterProperty('useforeign', 'boolean', iptrw);
23451:    RegisterProperty('usecaps', 'boolean', iptrw);
23452:    RegisterProperty('tag', 'integer', iptrw);
23453:    Proc Init(newletters:str;NewMinLen,NewMaxLen:word;newa,newf,newc:bool;pubDic:TDic);
23454:    Func FindMissingLetter( const s :Str) : TTestWords');
23455:    Proc Findallwords( const s :Str; list : Tstrings);
23456:  end;
23457: end;
23458:
23459: unit U_Invertedtext;
23460:   Proc InitInvertedText(canvas:TCanvas; pagewidth,pageheight:integer);
23461:   Proc DrawInvertedText(Canvas:TCanvas;pagewidth,pageheight:integer;linenbr:ints:str);
23462:   Proc MemoTextFixUp(memo:TMemo {var text:str});
23463:
23464: unit hashunit
23465: Proc SIRegister_THashStr(CL: TPSPascalCompiler);
23466: begin
23467:  //with RegClassS(CL,'TObject', 'THashStr') do
23468:  with CL.AddClassN(CL.FindClass('TObject'),'THashStr') do begin
23469:    y('test', 'array of THashObject', iptrw);
23470:    y('maxhash', 'integer', iptrw);
23471:    y('maxloadfactor', 'single', iptrw);
23472:    y('maxused', 'integer', iptrw);
23473:    y('used', 'integer', iptrw);
23474:    y('nbrcollisions', 'integer', iptrw);
23475:    y('maxcollisions', 'integer', iptrw);
23476:    Constructor create( newmaxhash : integer; newmaxloading : single);
23477:    Proc Free');
23478:    Func exists( s :Str; var T : TObject) :Bool');
23479:    Func AddIfNotDup( s :Str; t : Tobject) :Bool');
23480:    Proc resethash');
23481:    Proc rehash');
23482:    Func hash( s :Str) : integer');
23483:  end;
23484: end;
23485:
23486: Proc SortString(var SortSt : Ansistr);
23487: SortSt:= 'this is maXbox4';  SortString2(SortSt);  >>> 4Xabhiimosstx
23488: * === compile-time registration functions === *)
23489: (*----------------------------------------------------------------------------*)
23490: Proc SIRegister_JsonsUtilsEx(CL: TPSPascalCompiler);
23491: begin
23492:  Func FixedFloatToStr( const Value : Extended) :Str');
23493:  Func FixedTryStrToFloat( const S :Str; out Value : Extended) :Bool');
23494:  Func FixedStrToFloat( const S :Str) : Extended');
23495:  Func __ObjectToJson( aObject : TObject) :Str');
23496:  Proc __jsonToObject( const aJSONString :Str; var aObject : TObject);
23497:   CL.AddTypeS('TObjectDynArray', 'array of TObject');
23498:   TStringDynArray2', 'array of string');
23499:   TIntegerDynArray2', 'array of Integer');
23500:  CL.AddConstantN('GLB_JSON_STD_DECIMALSEPARATOR','String').SetString( '.');
23501: end;
23502:
23503: Proc SIRegister_TJson(CL: TPSPascalCompiler);
23504: begin
23505:  //with RegClassS(CL,'TJsonBase', 'TJson') do
23506:  with CL.AddClassN(CL.FindClass('TJsonBase'),'TJson') do begin
23507:    Constructor Create');
23508:    Proc Free');
23509:    Proc Parse( JsonString :Str);
23510:    Func Stringify :Str');
23511:    Proc Assign( Source : TJsonBase);
23512:    Proc Delete21( const Index : Integer);
23513:    Proc Delete22( const Name :Str);
23514:    Proc Clear');
23515:    Func Get23( const Index : Integer) : TJsonValue;
23516:    Func Get24( const Name :Str) : TJsonValue;
23517:    Func Put25( const Value : TJsonEmpty) : TJsonValue;
23518:    Func Put26( const Value : TJsonNull) : TJsonValue;
23519:    Func Put27( const Value :Bool) : TJsonValue;
23520:    Func Put28( const Value : Integer) : TJsonValue;
23521:    Func Put29( const Value : Extended) : TJsonValue;
23522:    Func Put30( const Value :Str) : TJsonValue;
23523:    Func Put31( const Value : TJsonArray2) : TJsonValue;
```

```
23524:    Func Put32( const Value : TJsonObject2) : TJsonValue;
23525:    Func Put33( const Value : TJsonValue) : TJsonValue;
23526:    Func Put34( const Value : TJson) : TJsonValue;
23527:    Func Putbool( const Value :Bool) : TJsonValue;  ----> alias
23528:    Func Putint( const Value : Integer) : TJsonValue;
23529:    Func Putext( const Value : Extended) : TJsonValue;
23530:    Func Putstr( const Value :Str) : TJsonValue;
23531:    Func Putarr( const Value : TJsonArray2) : TJsonValue;
23532:    Func Putobj( const Value : TJsonObject2) : TJsonValue;
23533:    Func Putval( const Value : TJsonValue) : TJsonValue;
23534:    Func Putjson( const Value : TJson) : TJsonValue;
23535:    Func Put35( const Name :Str; const Value : TJsonEmpty) : TJsonValue;
23536:    Func Put36( const Name :Str; const Value : TJsonNull) : TJsonValue;
23537:    Func Put37( const Name :Str; const Value :Bool) : TJsonValue;
23538:    Func Put38( const Name :Str; const Value : Integer) : TJsonValue;
23539:    Func Put39( const Name :Str; const Value : Extended) : TJsonValue;
23540:    Func Put40( const Name :Str; const Value :Str) : TJsonValue;
23541:    Func Put41( const Name :Str; const Value : TJsonArray2) : TJsonValue;
23542:    Func Put42( const Name :Str; const Value : TJsonObject2) : TJsonValue;
23543:    Func Put43( const Name :Str; const Value : TJsonValue) : TJsonValue;
23544:    Func Put44( const Name :Str; const Value : TJson) : TJsonValue;
23545:    Func Put45( const Value : TJsonPair) : TJsonValue;
23546:    RegisterProperty('StructType', 'TJsonStructType', iptr);
23547:    'JsonObject', 'TJsonObject2', iptr);
23548:    'JsonArray', 'TJsonArray2', iptr);
23549:    'Count', 'Integer', iptr);
23550:    'Values', 'TJsonValue String', iptr);
23551:    SetDefaultPropery('Values');
23552:  end;
23553: end;
23554:
23555: (*----------------------------------------------------------------------------*)
23556: Proc SIRegister_TJsonBase(CL: TPSPascalCompiler);
23557: begin
23558:   //with RegClassS(CL,'TObject', 'TJsonBase') do
23559:   with CL.AddClassN(CL.FindClass('TObject'),'TJsonBase') do  begin
23560:     Constructor Create( AOwner : TJsonBase);
23561:     Proc Free');
23562:     Proc Parse( JsonString :Str);
23563:     Func Stringify :Str');
23564:     Proc Assign( Source : TJsonBase);
23565:     Func Encode( const S :Str) :Str');
23566:     Func Decode( const S :Str) :Str');
23567:     Proc Split( const S :Str; const Delimiter: Char; Strings:TStrings);
23568:     Func IsJsonObject( const S :Str) :Bool');
23569:     Func IsJsonArray( const S :Str) :Bool');
23570:     Func IsJsonString( const S :Str) :Bool');
23571:     Func IsJsonNumber( const S :Str) :Bool');
23572:     Func IsJsonBoolean( const S :Str) :Bool');
23573:     Func IsJsonNull( const S :Str) :Bool');
23574:     Func AnalyzeJsonValueType( const S :Str) : TJsonValueType');
23575:     RegisterProperty('Owner', 'TJsonBase', iptr);
23576:   end;
23577: end;
23578:
23579: (*----------------------------------------------------------------------------*)
23580: Proc SIRegister_Jsons(CL: TPSPascalCompiler);
23581: begin
23582:   TJsonValueType','( jvNone, jvNull, jvString, jvNumber, jvBoolean, jvObject,jvArray );
23583:   TJsonStructType','( jsNone, jsArray, jsObject );
23584:   //CL.AddTypeS('TJsonNull', '( null );
23585:   //CL.AddTypeS('TJsonEmpty', '( empty );
23586:   SIRegister_TJsonBase(CL);
23587:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJsonObject2');
23588:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJsonArray2');
23589:   SIRegister_TJsonValue(CL);
23590:   SIRegister_TJsonArray2(CL);
23591:   SIRegister_TJsonPair(CL);
23592:   SIRegister_TJsonObject2(CL);
23593:   SIRegister_TJson(CL);
23594: end;
23595:    var jOb:= ajt.JsonObject;
23596:     for cnt:= 2 to job.count-2 do begin
23597:        Clabel:= job.items[cnt].name;
23598:        writeln('iterate '+clabel)
23599:        JsArr:= job.values[Clabel].asArray;
23600:        for cnt2:= 0 to jsarr.count-1 do
23601:          jsobj:= jsarr.items[cnt2].asobject;
23602:          for cnt3:= 0 to jsobj.count do
23603:            writeln(jsobj['elements'].asarray[0].asobject.items[cnt3].name)
23604:     end;
23605:
23606: println('elements status:
       '+ajt['rows'].asarray[0].asObject['elements'].asarray[0].asobject['status'].asstring);
23607:
23608: Proc SIRegister_Bricks(CL: TPSPascalCompiler);
23609: begin
23610:  CL.AddConstantN('kError101','String').SetString( 'Runtime error: Negative parameter(s);
23611:  kError102','String').SetString( 'Runtime error: Parameter(s) out of range');
```

```
23612:  kError103','String').SetString( 'Runtime error: min > max');
23613:  kError104','String').SetString( 'Runtime error: max = 0');
23614:  kError105','String').SetString( 'Runtime error: Denominator is zero');
23615:  CL.AddTypeS('TVectorE', 'array of extended');
23616:  complexreal', 'record re : real; im : real; end');
23617:  TFR', 'record M : extended; phi : extended; F : complexreal; end');
23618:  //CL.AddTypeS('complexreal', 'record re : real; im : real; end');
23619:  SIRegister_TBlock(CL);
23620:  SIRegister_TP2(CL);
23621:  SIRegister_TPT0(CL);
23622:  SIRegister_TPT1(CL);
23623:  SIRegister_TPT2(CL);
23624:  SIRegister_TInt2(CL);
23625:  SIRegister_TIT1(CL);
23626:  SIRegister_TDT1(CL);
23627:  SIRegister_TIT2(CL);
23628:  SIRegister_TPAdd(CL);
23629:  SIRegister_TPSub(CL);
23630:  SIRegister_TPMul(CL);
23631:  SIRegister_TPDiv(CL);
23632: end;
23633:
23634: Proc SIRegister_lifeblocks(CL: TPSPascalCompiler);
23635: begin
23636:   SIRegister_TASIA(CL);
23637:   SIRegister_TMiMe(CL);
23638:   SIRegister_TNoCoDI(CL);
23639: end;
23640:
23641: ****************************************************************
23642: Release Notes maXbox 4.7.4.64 June 2020 mX47
23643: ****************************************************************
23644: 1254 unit uPSI_MaskEdit.pas    FCL
23645: 1255 unit uPSI_SimpleRSSTypes; BlueHippo
23646: 1256 unit uPSI_SimpleRSS;      BlueHippo
23647: 1257 unit uPSI_psULib.pas      Prometheus
23648: 1258 unit uPSI_psUFinancial;   Prometheus
23649: 1259 uPSI_PsAPI_2.pas          mX4
23650: 1260 uPSI_PersistSettings_2    mX4
23651: 1261 uPSI_rfc1213util2.pas     IP
23652: 1262 uPSI_JTools.pas           JCL
23653: 1263 unit uPSI_neuralbit.pas   CAI
23654: 1264 unit uPSI_neuralab.pas    CAI
23655: 1265 unit uPSI_winsvc2.pas     TEK
23656: 1266 unit uPSI_wmiserv2.pas    TEK
23657: 1267 uPSI_neuralcache.pas      CAI
23658: 1268 uPSI_neuralbyteprediction CAI
23659: 1269 unit uPSI_USolarSystem;   glscene.org
23660: 1270 uPSI_USearchAnagrams.pas  DFF
23661: 1271 uPSI_JsonsUtilsEx.pas     Randolph
23662: 1272 unit uPSI_Jsons.pas       Randolph
23663: 1273 unit uPSI_HashUnit;       DFF
23664: 1274 uPSI_U_Invertedtext.pas   DFF
23665: 1275 unit uPSI_Bricks;         Dendron
23666: 1276 unit uPSI_lifeblocks.pas  Dendron
23667:
23668: Totals of Func Calls: 32633
23669: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
23670: CRC32: 3EB27A87:  28.2 MB (29,608,248) bytes
23671: Totals of Func Calls: 32614
23672:
23673: Functions_max hex in the box maXbox
23674: functionslist.txt
23675: 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
23676:
23677: Proc SIRegister_JclStringLists(CL: TPSPascalCompiler);
23678: begin
23679:  CL.AddClassN(CL.FindClass('TOBJECT'),'EJclStringListError');
23680:  CL.AddInterface(CL.FindInterface('IUNKNOWN'),IJclStringList, 'IJclStringList');
23681:  CL.AddTypeS('TJclStringListObjectsMode', '( omNone, omObjects, omVariants, omInterfaces );
23682:  SIRegister_IJclStringList(CL);
23683:  SIRegister_TJclInterfacedStringList(CL);
23684:  SIRegister_TJclStringList(CL);
23685:  Func JclStringList : IJclStringList;
23686:  Func JclStringListStrings( AStrings : TStrings) : IJclStringList;
23687:  Func JclStringListStrings1( const A : array of string) : IJclStringList;
23688:  Func JclStringList1( const A : array of const) : IJclStringList;
23689:  Func JclStringList2( const AText :Str) : IJclStringList;
23690: end;
23691:
23692: (*----------------------------------------------------------------------------*)
23693: Proc SIRegister_cInternetUtils(CL: TPSPascalCompiler);
23694: begin
23695:  ('ctHTML','String').SetString( 'text/html');
23696:  ('ctText','String').SetString( 'text/plain');
23697:  ('ctXML','String').SetString( 'text/xml');
23698:  ('ctJPG','String').SetString( 'image/jpeg');
23699:  ('ctGIF','String').SetString( 'image/gif');
```

```
23700:  ('ctBMP','String').SetString( 'image/bmp');
23701:  ('ctPNG','String').SetString( 'image/png');
23702:  ('ctTIFF','String').SetString( 'image/tiff');
23703:  ('ctMPG','String').SetString( 'video/mpeg');
23704:  ('ctAVI','String').SetString( 'video/avi');
23705:  ('ctQT','String').SetString( 'video/quicktime');
23706:  ('ctBinary','String').SetString( 'application/binary');
23707:  ('ctPDF','String').SetString( 'application/pdf');
23708:  ('ctPostscript','String').SetString( 'application/postscript');
23709:  ('ctBasicAudio','String').SetString( 'audio/basic');
23710:  ('ctMP3','String').SetString( 'audio/mpeg');
23711:  ('ctRA','String').SetString( 'audio/x-realaudio');
23712:  ('ctURLEncoded','String').SetString( 'application/x-www-form-urlencoded');
23713:  ('ctZIP','String').SetString( 'application/zip');
23714:  ('ctJavaScript','String').SetString( 'application/javascript');
23715:  ('ctPascal','String').SetString( 'text/x-source-pascal');
23716:  ('ctCPP','String').SetString( 'text/x-source-cpp');
23717:  ('ctINI','String').SetString( 'text/x-windows-ini');
23718:  ('ctBAT','String').SetString( 'text/x-windows-bat');
23719:  Func flcMIMEContentTypeFromExtention( const Extention :Str):Str');
23720:  ('protoHTTP','String').SetString( 'http');
23721:  ('protoNNTP','String').SetString( 'news');
23722:  ('protoFTP','String').SetString( 'ftp');
23723:  ('protoGopher','String').SetString( 'gopher');
23724:  ('protoEMail','String').SetString( 'mailto');
23725:  ('protoHTTPS','String').SetString( 'https');
23726:  ('protoIRC','String').SetString( 'irc');
23727:  ('protoFile','String').SetString( 'file');
23728:  ('protoTelnet','String').SetString( 'telnet');
23729:  Proc flcDecodeURL( const URL :Str; var Protocol, Host, Path :Str);
23730:  Func flcEncodeURL( const Protocol, Host, Path :Str) :Str');
23731:  Proc flcDecodeHost(const Address:str;var Host,Port:str;const DefaultPort:str);
23732:  Func flcEncodeDotLineTerminated( const S : Ansistr) : Ansistr');
23733:  Func flcEncodeEmptyLineTerminated( const S : Ansistr) : Ansistr');
23734:  Func flcDecodeDotLineTerminated( const S : Ansistr) : Ansistr');
23735:  Func flcDecodeEmptyLineTerminated( const S : Ansistr) : Ansistr');
23736:  //CL.AddDelphiFunction('Proc StreamDotLineTerminated0(const Source,Destination: AStream; const
        ProgressCallback:TCopyProgressProcedure);
23737:  //CL.AddDelphiFunction('Proc StreamDotLineTerminated1(const Source:str;const Destination:AStream;const
        ProgressCallback:TCopyProgressProcedure);
23738:  Func flchtmlCharRef( const CharVal : LongWord; const UseHex :Bool):Str');
23739:  Func flchtmlSafeAsciiText( const S :Str) :Str');
23740:  Proc flchtmlSafeWideText( var S : WideString);
23741:  Func flchtmlSafeQuotedText( const S :Str) :Str');
23742:  Func flcEncodeHeaderField( const Name, Body :Str) :Str');
23743:  Func flcDecodeHeaderField( const S :Str; var Name, Body :Str) :Bool');
23744:  Proc flcDecodeEMailAddress( const S :Str; var User, Domain :Str);
23745:  Proc flcDecodeEMailField( const S :Str; var EMailAddress, Name :Str);
23746:  Func flcDateFieldBody :Str');
23747:  Func flcDateField :Str');
23748:  Func flcMessageIDFieldBody( const ID :Str; const Host :Str) :Str');
23749:   SIRegister_AHeaderField(CL);
23750:   //CL.AddTypeS('AHeaderFieldClass', 'class of AHeaderField');
23751:   CL.AddTypeS('AHeaderFieldArray', 'array of AHeaderField');
23752:   SIRegister_THeaderCls(CL);
23753:   //CL.AddTypeS('THeaderClass', 'class of THeader');
23754:   CL.AddTypeS('THeaderArray', 'array of THeaderCls');
23755:   SIRegister_THeaderField(CL);
23756:   SIRegister_TInvalidField(CL);
23757:   SIRegister_TDateFieldClass(CL);
23758:   SIRegister_TEMailField(CL);
23759:  CL.AddDelphiFunction('Proc SelfTestInternetUtils');
23760:  maxform1.JumptoOutput1Click(self)
23761: end;
23762:
23763: Proc SIRegister_cWindows(CL: TPSPascalCompiler);
23764: Func InvMod(const A, N: Integer): Integer;
23765: Func ExpMod(A, Z: Integer; const N: Integer): Integer;
23766: Func CopyFrom(const S:Str; const Index: Integer):Str;
23767: Func CopyEx(const S:Str; const Start, Count: Integer):Str;
23768: Func loadForm2(vx, vy: smallint; acolor: TColor; aname:Str): TForm;
23769: PlaySound('INDIGO_WAV', hInstance, SND_RESOURCE or SND_ASYNC);
23770: Proc flcRaiseLastOSError');
23771:  Func flcStrEqualNoCase( const A,B:Str; const AsciiCaseSensitive :Bool):Bool');
23772:  Func flcStrMatchNoAsciiCase( const S, M :Str; const Index : Integer) :Bool');
23773:  Func flcStrMatchLeft( const S, M :Str; const AsciiCaseSensitive :Bool):Bool');
23774:  Func flcStrMatchRight( const S,M:Str; const AsciiCaseSensitive :Bool) :Bool');
23775:  Func flcPosChar( const F : Char; const S :Str; const Index : Integer) : Integer');
23776:  Func flcPosStr(const F,S:str;const Index:Integer;const AsciiCaseSensitive:Bool):nteger;
23777:  Func flcStrMatch( const S, M :Str; const Index : Integer) :Bool');
23778:  Func flcStrPMatch( const A, B : PChar; const Len : Integer) :Bool');
23779:  Func flcflcCopyFrom( const S :Str; const Index : Integer) :Str');
23780:  Func flcCopyRange( const S :Str; const StartIndex, StopIndex : Integer) :Str');
23781:  Func flcCopyLeft( const S :Str; const Count : Integer) :Str');
23782:  Func flcCopyRight( const S :Str; const Count : Integer) :Str');
23783:  Func flcStrIsNumeric( const S :Str) :Bool');
23784:  Func flcStrMatchChar( const S :Str; const M : TSYsCharSet) :Bool');
23785: Func flcStrSplitAtChar(const S:Str;const C:Char;var Left,Right:Str const Optionl:Bool):Bool;
23786:  Func flcStrReplaceChar( const Find, Replace : Char; const S :Str) :Str');
```

```
23787: Func flcStrInclSuffix(const S:Str;const Suffix:Str;const AsciiCaseSensitive:Bool):str;
23788:  Func flcStrRemoveCharDelimited(var S:str;const FirstDelim,SecondDelim:Char) :Str');
23789:  Func flcStrSplit( const S, D :Str) : TStringArray');
23790:  Proc flcTrimInPlace( var S :Str; const C : TSYsCharSet);
23791:  Func flcTrimIn( S :Str; const C : TSYsCharSet) :Str');
23792:  Func flcStrAfterCharSet( const S :Str; const D : TSYsCharSet) :Str');
23793:  Func flcStrAfterChar( const S :Str; const D : Char) :Str');
23794:  Proc StrSplit(Delimiter: Char; Str:Str; ListOfStrings: TStrings) ;
23795:  (*----------------------------------------------------------------------------*)
23796: Proc SIRegister_cWindows(CL: TPSPascalCompiler);
23797: begin
23798:  CL.AddTypeS('TWindowsVersionflc', '( Win16_31, Win32_95, Win32_95R2, Win32_98, W'
23799:   +'in32_98SE, Win32_ME, Win32_Future, WinNT_31, WinNT_35, WinNT_351, WinNT_40'
23800:   +', WinNT5_2000, WinNT5_XP, WinNT5_2003, WinNT5_Future, WinNT_Future, Win_Future );
23801:  CL.AddTypeS('TWindowsVersions', 'set of TWindowsVersionflc');
23802:  Func flcGetWindowsVersion : TWindowsVersionflc');
23803:  Func flcIsWinPlatform95 :Bool');
23804:  Func flcIsWinPlatformNT :Bool');
23805:  Func flcGetWindowsProductID :Str');
23806:  Func flcGetWindowsTemporaryPath :Str');
23807:  Func flcGetWindowsPath :Str');
23808:  Func flcGetWindowsSystemPath :Str');
23809:  Func flcGetProgramFilesPath :Str');
23810:  Func flcGetCommonFilesPath :Str');
23811:  Func flcGetApplicationPath :Str');
23812:  Func flcGetUserName :Str');
23813:  Func flcGetLocalComputerName :Str');
23814:  Func flcGetLocalHostName :Str');
23815: CL.AddTypeS('TVersionInfoflc',(viFileVersion,viFileDescription,viLegalCopyrght,viComments,viCompanyName,
       viInternalNme,viLegalTrademarks,viOriginalFilename,viProductNme,viProductVersion;
23816:  Func flcGetAppVersionInfo( const VersionInfo : TVersionInfoflc) :Str');
23817:  Func flcWinExecute(const ExeName,Params:str;const ShowWin:Word;const Wait:Bool):Bool);
23818:  Func flcGetEnvironmentStrings : TStringArray');
23819:  Func flcContentTypeFromExtention( Extention :Str) :Str');
23820:  Func flcIsApplicationAutoRun( const Name :Str) :Bool');
23821:  Proc flcSetApplicationAutoRun( const Name :Str; const AutoRun :Bool);
23822:  Func flcGetWinPortNames : TStringArray');
23823:  Func flcGetKeyPressed( const VKeyCode : Integer) :Bool');
23824:  Func flcGetHardDiskSerialNumber( const DriveLetter : Char) :Str');
23825:  Func flcReboot :Bool');
23826: //CL.AddDelphiFunction('Func ConvertThreadToFiber( lpParameter : Pointer) : Pointer');
23827: //Func CreateFiber(dwStackSize:DWORD;lpStartAddress:TFNFiberStartRoutine;lpParameter:Ptr):Ptr');
23828:   CL.AddTypeS(TIEProxy', '( iepHTTP, iepHTTPS, iepFTP, iepGOPHER, iepSOCKS );
23829:  CL.AddDelphiFunction('Func GetIEProxy( const Protocol : TIEProxy) :Str');
23830:   CL.AddTypeS(TWindowHandleMessageEvent','Func(const Msg:Card;const wParam,lParam:Int;var
       Handled:Bool):Int;
23831:   CL.AddClassN(CL.FindClass('TOBJECT'),'TWindowHandle');
23832:   TWindowHandleEvent', 'Proc ( const Sender : TWindowHandle);
23833:   (TWindowHandleErrorEvent','Procedure(const Sender:TWindowHandle;const E:Exception);
23834:   SIRegister_TWindowHandle(CL);
23835:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWindowHandle');
23836:   SIRegister_TfndWindowHandle(CL);
23837:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTimerHandle');
23838:   TTimerEventflc', 'Proc ( const Sender : TTimerHandle);
23839:   SIRegister_TTimerHandle(CL);
23840:   SIRegister_TfndTimerHandle(CL);
23841:  Proc flcRaiseLastOSError');
23842:  Func flcStrEqualNoCase(const A,B:Str; const AsciiCaseSensitive :Bool):Bool');
23843:  Func flcStrMatchNoAsciiCase( const S, M :Str; const Index : Integer) :Bool');
23844:  Func flcStrMatchLeft( const S, M :Str; const AsciiCaseSensitive :Bool):Bool');
23845:  Func flcStrMatchRight(const S,M :Str; const AsciiCaseSensitive :Bool):Bool');
23846:  Func flcPosChar( const F : Char; const S :Str; const Index : Integer): Integer');
23847:  Func flcPosStr(const F,S:str;const Index:Integer;const AsciiCaseSensitive:Bool):Integer;
23848:  Func flcStrMatch( const S, M :Str; const Index : Integer):Bool');
23849:  Func flcStrPMatch( const A, B : PChar; const Len : Integer):Bool');
23850:  Func flcflcCopyFrom( const S :Str; const Index : Integer):Str');
23851:  Func flcCopyRange( const S :Str; const StartIndex, StopIndex :Integer):str');
23852:  Func flcCopyLeft( const S :Str; const Count : Integer):Str');
23853:  Func flcCopyRight( const S :Str; const Count : Integer):Str');
23854:  Func flcStrIsNumeric( const S :Str) :Bool');
23855:  Func flcStrMatchChar( const S :Str; const M : TSYsCharSet) :Bool');
23856: Func flcStrSplitAtChar(const S Str;const C:Char;var Left,Right:Str;const Optional:Bool):Bool
23857:  Func flcStrReplaceChar( const Find, Replace : Char; const S :Str) :Str');
23858:  Func flcStrInclSuffix(const S:Str;const Suffix:Str;const AsciiCaseSensitive:Bool):str;
23859:  Func flcStrRemoveCharDelimited(var S:str;const FirstDelim,SecondDelim:Char) :Str');
23860:  Func flcStrSplit( const S, D :Str) : TStringArray');
23861:  Proc flcTrimInPlace( var S :Str; const C : TSYsCharSet);
23862:  Func flcTrimIn( S :Str; const C : TSYsCharSet) :Str');
23863:  Func flcStrAfterCharSet( const S :Str; const D : TSYsCharSet) :Str');
23864:  Proc StrSplit(Delimiter: Char; Str:Str; ListOfStrings: TStrings) ;
23865:  Func StrSplitF(Delimiter: Char; Str:Str): TStringlist;
23866:  Func Int32toBytes(const Cint: Integer): TBytes;
23867:  Proc FGPlayASound(const AResName:Str);
23868:  //CL.AddDelphiFunction('Proc FGPlayASound(const AResName:Str);
23869:  Proc FGPlayASound2(const AResName:Str; atype:Str);
23870:  Proc PlayResWav(const AResName:Str; atype:Str);
23871:  Proc PlayResWav2(const AResName:Str; atype:Str);  //sync tick1.wav
23872:  procedure PlaySoundStreamSync(const AName: TMemoryStream; atype: string);
23873:  procedure PlaySoundStreamAsync(const AName: TMemoryStream; atype: string);
```

```
23874:  procedure PlaySoundStreamAsyncNoStop(const AName: TMemoryStream; atype: string);
23875:  Func queryPerformanceCounter2(var mse: int64):Bool;
23876:  Func AddQuantumToDateTime(const dt: TDateTime): TDateTime;
23877:  Func AddQuantumToDateTime(const dt: TDateTime): TDateTime;
23878:  var overlay: Int64 absolute Result;
23879:    begin
23880:      Result:= dt;
23881:      overlay:= overlay+1;
23882:    end;
23883:  Func GetRTFtext(Const RichEdit: TRichEdit):Str;
23884:  Func StrRemoveCharSet(const S:Str; const C: CharSet):Str;
23885:  Func GetTextBetween(const Start, Stop:Str;var S, Between:Str):Boolean;
23886: end;
23887:
23888: Proc SIRegister_flcSysUtils(CL: TPSPascalCompiler);
23889: begin
23890:  SIRegister_TfreqObj(CL);
23891:  CL.AddTypeS('MSArray','array[0..1] of tmemorystream;
23892:  Func GetLastOSErrorCode : NativeInt');
23893:  Func GetLastOSErrorMessage :Str');
23894:  Proc TForm1msPlaySound( MS : MSArray; streaminuse : integer; aloop:boolean);
23895:  Proc BytesSetLengthAndZero( var V : TBytes; const NewLength : NativeInt);
23896:  Proc BytesInit( var V : TBytes; const R : Byte);
23897:  Proc BytesInit1( var V : TBytes; const S :Str);
23898:  Func BytesAppend( var V : TBytes; const R : Byte) : NativeInt');
23899:  Func BytesAppend1(var V : TBytes; const R : TBytes) : NativeInt');
23900:  Func BytesAppend2(var V : TBytes; const R : array of Byte) : NativeInt');
23901:  Func BytesAppend3(var V : TBytes; const R :Str) : NativeInt');
23902:  Func BytesCompare(const A, B : TBytes) : Integer');
23903:  Func BytesEqual( const A, B : TBytes) :Bool');
23904:  Func StringRefCount2(const S: Ansistr): Integer;
23905: Func  hton16(const A:Word): Word;
23906: Func  ntoh16(const A:Word): Word;
23907: Func  hton32(const A:LongWord): LongWord;
23908: Func  ntoh32(const A:LongWord): LongWord;
23909: Func  hton64(const A:Int64): Int64;
23910: Func  ntoh64(const A:Int64): Int64;
23911: CL.AddTypeS(TPoint3D2', 'record x : single; y : single; z : single; end;
23912: Func SphereTPoint3D(Phi, Lambda: Double): TPoint3D2;
23913: Func RotateAroundXTPoint3D(const P: TPoint3D2; Alfa: Double): TPoint3D2;
23914: Func RotateAroundYTPoint3D(const P: TPoint3D2; Beta: Double): TPoint3D2;
23915: Func LinkerTimeStamp(const FileName:Str): TDateTime; overload;
23916: Func ByteCharDigitToInt(const A: Char): Integer;
23917: Func WideCharDigitToInt(const A: WideChar): Integer;
23918: Func CharDigitToInt(const A: Char): Integer;
23919: Func IntToByteCharDigit(const A: Integer): Char;
23920: Func IntToWideCharDigit(const A: Integer): WideChar;
23921: Func IntToCharDigit(const A: Integer): Char;
23922: Func flcMinInt(const A, B: Int64): Int64;
23923: Func flcMaxInt(const A, B: Int64): Int64;
23924: end;
23925:
23926: Proc SIRegister_RotImg(CL: TPSPascalCompiler);
23927: begin
23928:  SIRegister_TRotateImage(CL);
23929:  Func CreateRotatedBitmap(Bitmap: TBitmap; const Angle:Extended; bgColor: TColor): TBitmap;
23930: end;
23931:
23932: *----------------------------------------------------------------------------*)
23933: Proc SIRegister_HSLUtils(CL: TPSPascalCompiler);
23934: begin
23935:  Func ARGB( a, r, g, b : Byte) :Card');
23936:  Func GetAValue2( argb : DWORD) : Byte');
23937:  Func GetRValue2( argb : DWORD) : Byte');
23938:  Func GetGValue2( argb : DWORD) : Byte');
23939:  Func GetBValue2( argb : DWORD) : Byte');
23940:  Func GetABGRAValue(abgr: DWORD) : Byte');
23941:  Func GetABGRRValue(abgr: DWORD) : Byte');
23942:  Func GetABGRGValue(abgr: DWORD) : Byte');
23943:  Func GetABGRBValue(abgr: DWORD) : Byte');
23944:  Func HSLtoRGB2( H, S, L : double) : TColor');
23945:  Func AHSLtoARGB( A, H, S, L : double) :Card');
23946:  Func HSLRangeToRGB( H, S, L : integer) : TColor');
23947:  Proc ARGBtoAHSL( ARGB : TColor; var A, H, S, L : double);
23948:  Proc ABGRtoAHSL( ABGR : TColor; var A, H, S, L : double);
23949:  Proc RGBtoHSLRange( RGB : TColor; var H, S, L : integer);
23950: end;
23951:
23952: Proc SIRegister_GraphicsMathLibrary(CL: TPSPascalCompiler);
23953: begin
23954:  gmlsizeUndefined','LongInt').SetInt( 1);
23955:  gmlsize2D','LongInt').SetInt( 3);
23956:  gmlsize3D','LongInt').SetInt( 4);
23957:  CL.AddClassN(CL.FindClass('TOBJECT'),'EVectorError');
23958:  CL.AddClassN(CL.FindClass('TOBJECT'),'EMatrixError');
23959:  gmlTAxis', '( axisX, axisY, axisZ );
23960:  gmlTCoordinate', '( coordCartesian, coordSpherical, coordCylindrical );
23961:  gmlTDimension', '( dimen2D, dimen3D );
23962:  gmlTIndex', 'Integer');
```

```
23963:  gmlTrotation', '( rotateClockwise, rotateCounterClockwise );
23964:  gmlTVector','record size: gmlTIndex; vector: ARRAY[1..4] OF DOUBLE; x,y,z,h:double; end;
23965:  gmlTMatrix', 'record size: gmlTIndex; matrix: ARRAY[1..4] of Array[1..4] OF DOUBLE; end;
23966:  Func gmlVector2D( const xValue, yValue : DOUBLE) : gmlTVector');
23967:  Func gmlVector3D( const xValue, yValue, zValue : DOUBLE) : gmlTVector');
23968:  Func gmlAddVectors(const u, v : gmlTVector) : gmlTVector');
23969:  Func gmlTransform( const u : gmlTVector; const a : gmlTMatrix) : gmlTVector');
23970:  Func gmlMatrix2D(const m11,m12 m13,m21,m22,m23,m31,m32,m33 : DOUBLE) : gmlTMatrix');
23971:  Func gmlMatrix3D(const m11,m12,m13,m14,m21,m22,m23,m24,m31,m32,m33,m34,m41,m42,m43,m44:DOUBLE):gmlTMatrix;
23972:  Func gmlMultiplyMatrices( const a, b : gmlTMatrix) : gmlTMatrix');
23973:  Func gmlInvertMatrix( const a, b : gmlTMatrix; var determinant : DOUBLE): gmlTMatrix');
23974:  Func gmlRotateMatrix(const dimension:gmlTDimension;const xyz:gmlTAxis;const angle:DOUBLE;const rotation
23975:                        :gmlTrotation):gmlTMatrix;
23976:  Func gmlScaleMatrix( const s : gmlTVector) : gmlTMatrix');
23977:  Func gmlTranslateMatrix( const t : gmlTVector) : gmlTMatrix');
23978:  Func gmlViewTransformMatrix(const coordinate:gmlTCoordinate;const azimuth,elevation,distance:DOUBLE;const
       ScreenX,ScreenY,ScreenDistance:DOUBLE):gmlTMatrix');
23979:  Func gmlFromCartesian(const ToCoordinate:gmlTCoordinate;const u gmlTVector): gmlTVector;
23980:  Func gmlToCartesian(const FromCoordinate: gmlTCoordinate;const u:gmlTVector): gmlTVector;
23981:  Func gmlToDegrees( const angle : DOUBLE) : DOUBLE');
23982:  Func gmlToRadians( const angle : DOUBLE) : DOUBLE');
23983:  Func gmlDefuzz( const x : DOUBLE) : DOUBLE');
23984:  Func gmlGetFuzz : DOUBLE');
23985:  Proc gmlSetFuzz( const x : DOUBLE);
23986: end;
23987:
23988: Proc SIRegister_umodels(CL: TPSPascalCompiler);
23989: begin
23990:  CL.AddTypeS('TRegType', '( REG_LIN, REG_MULT, REG_POL, REG_FRAC, REG_EXPO, RE'
23991:   +'G_IEXPO, REG_EXLIN, REG_LOGIS, REG_POWER, REG_GAMMA, REG_MICH, REG_MINT, R'
23992:   +'EG_HILL, REG_PK, REG_EVAL );
23993:   //CL.AddTypeS('TModel', 'record [..] end');
23994:  Func FirstParam( Model : TModel) : Integer');
23995:  Func LastParam( Model : TModel) : Integer');
23996:  Func FuncName( Model : TModel) :Str');
23997:  Func ParamName( Model : TModel; I : Integer) :Str');
23998:  Func RegFunc( Model : TModel; X : Float; B: TVector): Float');
23999:  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);
24000:  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);
24001: end;
24002:
24003: (*----------------------------------------------------------------------------*)
24004: Proc SIRegister_TStatisticClass(CL: TPSPascalCompiler);
24005: begin
24006:   //with RegClassS(CL,'TOBJECT', 'TStatisticClass') do
24007:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TStatisticClass') do begin
24008:    Proc Assign( const S : TStatisticClass);
24009:    Func Duplicate : TStatisticClass');
24010:    Proc Clear');
24011:    Func IsEqual( const S : TStatisticClASS):Bool');
24012:    Proc Add( const V : MFloat);
24013:    Proc Add1( const V : array of MFloat);
24014:    Proc Add2( const V : TStatisticClass);
24015:    Proc AddNegated( const V : TStatisticClass);
24016:    Proc Negate');
24017:    Count', 'Integer', iptr);
24018:    Min', 'MFloat', iptr);
24019:    Max', 'MFloat', iptr);
24020:    Sum', 'MFloat', iptr);
24021:    SumOfSquares', 'MFloat', iptr);
24022:    SumOfCubes', 'MFloat', iptr);
24023:    SumOfQuads', 'MFloat', iptr);
24024:    Func Range : MFloat');
24025:    Func Mean : MFloat');
24026:    Func PopulationVariance : MFloat');
24027:    Func PopulationStdDev : MFloat');
24028:    Func Variance : MFloat');
24029:    Func StdDev : MFloat');
24030:    Func M1 : MFloat');
24031:    Func M2 : MFloat');
24032:    Func M3 : MFloat');
24033:    Func M4 : MFloat');
24034:    Func Skew : MFloat');
24035:    Func Kurtosis : MFloat');
24036:    Func GetAsString :Str');
24037:   end;
24038: end;
24039:
24040: (*----------------------------------------------------------------------------*)
24041: Proc SIRegister_flcStatistics(CL: TPSPascalCompiler);
24042: begin
24043:   CL.AddClassN(CL.FindClass('TOBJECT'),'EStatistics');
24044:   CL.FindClass('TOBJECT'),'EStatisticsInvalidArgument');
24045:   CL.FindClass('TOBJECT'),'EStatisticsOverflow');
24046:   //CL.AddTypeS('MFloat', 'double');
24047:   Proc SIRegister_flcStatistics(CL: TPSPascalCompiler);  begin
24048:   TOBJECT'),'EStatistics');
```

```
24049:   TOBJECT'),'EStatisticsInvalidArgument');
24050:   TOBJECT'),'EStatisticsOverflow');
24051:   //CL.AddTypeS('MFloat', 'double');        //check
24052:   CL.AddTypeS('MFloat', 'extended');        //checked
24053:  Func flcBinomialCoeff( N, R : Integer) : MFloat');
24054:  Func flcerf( x : MFloat) : MFloat');
24055:  Func flcerfc( const x : MFloat) : MFloat');
24056:  Func flcCummNormal( const u, s, X : MFloat) : MFloat');
24057:  Func flcCummNormal01( const X : MFloat) : MFloat');
24058:  Func flcInvCummNormal01( Y0 : MFloat) : MFloat');
24059:  Func flcInvCummNormal( const u, s, Y0 : MFloat) : MFloat');
24060:  Func flcCummChiSquare( const Chi, Df : MFloat) : MFloat');
24061:  Func flcCumF( const f, Df1, Df2 : MFloat) : MFloat');
24062:  Func flcCummPoisson( const X : Integer; const u : MFloat) : MFloat');
24063:   SIRegister_TStatisticClass(CL);
24064:   CL.AddClassN(CL.FindClass('TOBJECT'),'EStatistic');
24065:   CL.FindClass('TOBJECT'),'EStatisticNoSample');
24066:   CL.FindClass('TOBJECT'),'EStatisticDivisionByZero');
24067:  Proc TestStatisticClass');
24068: end;
24069:
24070: Proc Include(var aset: Charset; achar: Char);  ---> cFundamentUtils
24071: Proc Exclude(var aset: Charset; achar: Char);
24072: Proc SIRegister_StdFuncs(CL: TPSPascalCompiler);
24073:  begin
24074:   CL.AddClassN(CL.FindClass('TOBJECT'),'EParserError');
24075:   CL.AddTypeS('TCharSet', 'set of Char');
24076:  Proc SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
24077:  CL.AddTypeS('CharSet', 'set of Char');       //!!!
24078:  CL.AddTypeS('AnsiCharSet', 'TCharSet');
24079:  CL.AddTypeS('ByteSet', 'set of Byte');
24080:  CL.AddTypeS('AnsiChar', 'Char');
24081:
24082: Proc SIRegister_flcMaths(CL: TPSPascalCompiler);
24083: begin
24084:  flcPi','Extended).setExtended(3.14159265358979323846+0.26433832795028841971e-20+0.69399375105820974944e-
       40+0.59230781640628620899e-60+0.86280348253421170679e-80+
       100+0.09384460955058223172e-120+0.53594081284811174502e-140+
       160+0.64462294895493038196e-180);
24085:  'flcPi2','Extended').setExtended( 6.28318530717958647692528676655900 6);
24086:  'flcPi3','Extended').setExtended( 9.42477796076937971538793014983850 9);
24087:  'flcPi4','Extended').setExtended( 12.56637061435917295385057353311801);
24088:  'flcPiOn2','Extended').setExtended( 1.57079632679489661923132169163975 1);
24089:  'flcPiOn3','Extended').setExtended( 1.04719755119659774615421446109316 8);
24090:  'flcPiOn4','Extended').setExtended( 0.78539816339744830961566084581987 6);
24091:  'flcPiSq','Extended').setExtended( 9.86960440108935861883449099987615 1);
24092:  'flcPiE','Extended').setExtended( 22.45915771836104547342715220454374);
24093:  'flcLnPi','Extended').setExtended( 1.14472988584940017414342735135305 9);
24094:  'flcLogPi','Extended').setExtended( 0.49714987269413385435126828829089 9);
24095:  'flcSqrtPi','Extended').setExtended( 1.77245385090551602729816748334114 5);
24096:  'flcSqrt2Pi','Extended').setExtended( 2.50662827463100050241576528481104 5);
24097:  'flcLnSqrt2Pi','Extended').setExtended( 0.91893853320467274178032973640561 8);
24098:  'flcDegPerRad','Extended').setExtended( 57.29577951308232087679815481410517);
24099:  'flcDegPerGrad','Extended').setExtended( 0.9);
24100:  'flcDegPerCycle','Extended').setExtended( 360.0);
24101:  'flcGradPerCycle','Extended').setExtended( 400.0);
24102:  'flcGradPerDeg','Extended').setExtended( 1.1111111111111111111111111111111 1);
24103:  'flcGradPerRad','Extended').setExtended( 63.66197723675813430755350534900 6);
24104:  'flcRadPerDeg','Extended').setExtended( 0.01745329251994329576923690768488 6);
24105:  'flcRadPerGrad','Extended').setExtended( 0.01570796326794896619231321691639 8);
24106:  'flcRadPerCycle','Extended').setExtended( 6.28318530717958647692528676655 9);
24107:  'flcCyclePerDeg','Extended').setExtended( 0.00277777777777777777777777777777 78);
24108:  'flcCyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
24109:  'flcCyclePerGrad','Extended').setExtended( 0.0025);
24110:  'flcE','Extended').setExtended( 2.71828182845904523536028747135266 3);
24111:  'flcE2','Extended').setExtended( 7.38905609893065022723042746057500 8);
24112:  'flcExpM2','Extended').setExtended( 0.13533528323661269189399949497248 4);
24113:  'flcLn2','Extended').setExtended( 0.69314718055994530941723212145817 7);
24114:  'flcLn10','Extended').setExtended( 2.30258509299404568401799145468436 4);
24115:  'flcLogE','Extended').setExtended( 0.43429448190325182765112891891660 5);
24116:  'flcLog2','Extended').setExtended( 0.30102999566398119521373889472449 3);
24117:  'flcLog3','Extended').setExtended( 0.47712125471966243729502790325511 5);
24118:  'flcSqrt2','Extended').setExtended( 1.41421356237309504880168872420969 8);
24119:  'flcSqrt3','Extended').setExtended( 1.73205080756887729352744634150587 2);
24120:  'flcSqrt5','Extended').setExtended( 2.23606797749978969640917366873127 6);
24121:  'flcSqrt7','Extended').setExtended( 2.64575131106459059050161575363926 0);
24122:  //CL.AddTypeS('MFloat', 'Double');
24123:  //CL.AddTypeS('MFloatArray', 'DoubleArray');
24124:  //CL.AddTypeS('MFloat', 'Extended');      / check t<pe
24125:  CL.AddTypeS('ExtendedArray', 'array of MFloat');
24126:  //ExtendedArray = array of Extended;
24127:  CL.AddTypeS('MFloatArray', 'ExtendedArray');
24128:  CL.AddTypeS('Int64Array', 'array of Int64');
24129:  flcfx', 'Func (const x: Float): Float;
24130:  MFloatArray2', 'array of Mfloat');
24131:  //Int64Array  = array of Int64;
24132:  // CL.AddTypeS('PMFloat', '^MFloat // will not work');
24133:  Proc SetFPUPrecisionSingle');
24134:  Proc SetFPUPrecisionDouble');
```

```
24135:  Proc SetFPUPrecisionExtended');
24136:  Proc SetFPURoundingNearest');
24137:  Proc SetFPURoundingDown');
24138:  Proc SetFPURoundingUp');
24139:  Proc SetFPURoundingTruncate');
24140:  Func flcPolyEval(const X:MFloat;const Coef:array of MFloat;const N:Integer): MFloat;
24141:  Func flcSign( const R : Integer) : Integer;
24142:  Func flcSign1( const R : Int64) : Integer;
24143:  Func flcSign2( const R : Single) : Integer;
24144:  Func flcSign3( const R : Double) : Integer;
24145:  Func flcSign4( const R : Extended) : Integer;
24146:  Func flcFloatMod( const A, B : MFloat) : MFloat');
24147:  Func flcATan360( const X, Y : MFloat) : MFloat');
24148:  Func flcInverseTangentDeg( const X, Y : MFloat) : MFloat');
24149:  Func flcInverseTangentRad( const X, Y : MFloat) : MFloat');
24150:  Func flcInverseSinDeg( const Y, R : MFloat) : MFloat');
24151:  Func flcInverseSinRad( const Y, R : MFloat) : MFloat');
24152:  Func flcInverseCosDeg( const X, R : MFloat) : MFloat');
24153:  Func flcInverseCosRad( const X, R : MFloat) : MFloat');
24154:  Func flcDMSToReal( const Degs, Mins, Secs : MFloat) : MFloat');
24155:  Proc flcRealToDMS( const X : MFloat; var Degs, Mins, Secs : MFloat);
24156:  Proc flcPolarToRectangular( const R, Theta : MFloat; var X, Y : MFloat);
24157:  Proc flcRectangularToPolar( const X, Y : MFloat; var R, Theta : MFloat);
24158:  Func flcDistance( const X1, Y1, X2, Y2 : MFloat) : MFloat');
24159:  Func flcIsPrime( const N : Int64) :Bool');
24160:  Func flcIsPrimeFactor( const N, F : Int64) :Bool');
24161:  Func flcPrimeFactors( const N : Int64) : Int64Array');
24162:  Func flcGCD( const N1, N2 : Integer) : Integer;
24163:  Func flcGCD1( const N1, N2 : Int64) : Int64;
24164:  Func flcIsRelativePrime( const X, Y : Int64) :Bool');
24165:  Func flcInvMod( const A, N : Integer) : Integer;
24166:  Func flcInvMod1( const A, N : Int64) : Int64;
24167:  Func flcExpMod( A, Z : Integer; const N : Integer) : Integer;
24168:  Func flcExpMod1( A, Z : Int64; const N : Int64) : Int64;
24169:  Func flcJacobi( const A, N : Integer) : Integer');
24170:  CL.AddConstantN('FactorialMaxN','LongInt').SetInt( 1754);
24171:  //CL.AddConstantN('FactorialMaxN','LongInt').SetInt( 170);
24172: Func flcFactorial( const N : Integer) : MFloat');
24173: Func flcCombinations( const N, C : Integer) : MFloat');
24174: Func flcPermutations( const N, P : Integer) : MFloat');
24175: Func flcFibonacci( const N : Integer) : Int64');
24176: Func flcGammaLn( X : Extended) : Extended');
24177: Proc flcFourierTransform(const AngleNumerator:MFloat;const RealIn,ImagIn:array of MFloat;var RealOut,
       ImagOut: MFloatArray);
24178:  Proc flcFFT( const RealIn,ImagIn:array of MFloat; var RealOut, ImagOut : MFloatArray);
24179:  Proc flcInverseFFT(const RealIn,ImagIn:array of MFloat;var RealOut,ImagOut:MFloatArray);
24180: Proc flcCalcFrequency(const FrequencyIndex:Int;const RealIn,ImagIn:array of MFloat;var RealOut,
       ImagOut:MFloat);
24181:  Func flcSecantSolver( const f : flcfx; const y, Guess1, Guess2 : MFloat) : MFloat');
24182:  Func flcNewtonSolver( const f, df : flcfx; const y, Guess : MFloat) : MFloat');
24183:  Func flcFirstDerivative( const f : flcfx; const x : MFloat) : MFloat');
24184:  Func flcSecondDerivative( const f : flcfx; const x : MFloat) : MFloat');
24185:  Func flcThirdDerivative( const f : flcfx; const x : MFloat) : MFloat');
24186:  Func flcFourthDerivative( const f : flcfx; const x : MFloat) : MFloat');
24187:  Func flcSimpsonIntegration( const f : flcfx; const a, b : MFloat; N : Integer): MFloat');
24188:  Proc TestMathClass');
24189: end;
24190:
24191: Proc SIRegister_flcCharSet(CL: TPSPascalCompiler);
24192: begin
24193: //type ByteSet = set of Byte;
24194: //ByteCharSet = Charset; /
24195:    CL.AddTypeS('ByteCharSet', 'Charset');
24196:    // CL.AddTypeS('ByteCharSet', 'charset');
24197:  Func flcAsAnsiCharSet( const C : array of Char) : ByteCharSet');
24198:  Func flcAsByteSet( const C : array of Byte) : ByteSet');
24199:  Func flcAsByteCharSet( const C : array of Char) : ByteCharSet');
24200:  Proc flcComplementChar( var C : ByteCharSet; const Ch : Char);
24201:  Proc flcClearCharSet( var C : ByteCharSet);
24202:  Proc flcFillCharSet( var C : ByteCharSet);
24203:  Proc flcComplementCharSet( var C : ByteCharSet);
24204:  Proc flcAssignCharSet( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24205:  Proc flcUnion( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24206:  Proc flcDifference( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24207:  Proc flcIntersection( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24208:  Proc flcXORCharSet( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24209:  Func flcIsSubSet( const A, B : ByteCharSet) :Bool');
24210:  Func flcIsEqual( const A, B : ByteCharSet) :Bool');
24211:  Func flcIsEmpty( const C : ByteCharSet) :Bool');
24212:  Func flcIsComplete( const C : ByteCharSet) :Bool');
24213:  Func flcCharCount( const C : ByteCharSet) : Integer');
24214:  Proc flcConvertCaseInsensitive( var C : ByteCharSet);
24215:  Func flcCaseInsensitiveCharSet( const C : ByteCharSet) : ByteCharSet');
24216:  Func flcCharSetToStrB( const C : ByteCharSet) : RawByteString');
24217:  Func flcStrToCharSetB( const S :Str) : ByteCharSet');
24218:  Func flcCharSetToCharClassStr( const C : ByteCharSet) : Ansistr');
24219:  Proc TestCharset');
24220: end;
24221:
```

```
24222: Proc SIRegister_flcTimers(CL: TPSPascalCompiler);
24223: begin
24224:  TOBJECT'),'ETimerError');
24225:  Func GetHighResolutionTick : Word64');
24226:  Func GetHighResolutionFrequency : Word64');
24227:  Func HighResolutionTickDelta( const T1, T2 : Word64) : Int64');
24228:  Func HighResolutionTickDeltaU( const T1, T2 : Word64) : Word64');
24229:  'MicroTickFrequency','LongInt').SetInt( 1000000);
24230:  Func GetMicroTick : Word64');
24231:  Func MicroTickDelta( const T1, T2 : Word64) : Int64');
24232:  Func MicroTickDeltaU( const T1, T2 : Word64) : Word64');
24233: 'MilliTickFrequency','LongInt').SetInt( 1000);
24234:  Func GetMilliTick : Word64');
24235:  Func MilliTickDelta( const T1, T2 : Word64) : Int64');
24236:  Func MilliTickDeltaU( const T1, T2 : Word64) : Word64');
24237:  Func DateTimeToMicroDateTime( const DT : TDateTime) : Word64');
24238:  Func MicroDateTimeToDateTime( const DT : Word64) : TDateTime');
24239:  Func GetMicroDateTimeNow : Word64');
24240:  Func GetNowUT : TDateTime');
24241:  Func GetMicroDateTimeNowUT : Word64');
24242:  Func GetNowUTC( const ReInit :Bool) : TDateTime');
24243:  Func GetMicroDateTimeNowUTC( const ReInit :Bool) : Word64');
24244:  Proc TestTimerClass;
24245: end;
24246:
24247: Proc SIRegister_TBlaiseLexer(CL: TPSPascalCompiler);
24248: begin
24249: //with RegClassS(CL,'TOBJECT', 'TBlaiseLexer') do
24250:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TBlaiseLexer') do begin
24251:   TokenCount', 'Integer', iptrw);
24252:   IdenCount', 'Integer', iptrw);
24253:   SymbolCount', 'Integer', iptrw);
24254:   IgnoreCount', 'Integer', iptrw);
24255:   ReservedCount', 'Integer', iptrw);
24256:   Constructor Create');
24257:   Proc Reset');
24258:   Proc SetData( const Data : pchar; const Size : Integer);
24259:   Proc SetText( const S :Str; const StartPosition : Integer);
24260:   Func EOF :Bool');
24261:   Proc GetToken');
24262:   TokenLen', 'Integer', iptr);
24263:   TokenType', 'Integer', iptr);
24264:   Func TokenText :Str');
24265:   TokenPos', 'Integer', iptr);
24266:   LastTokenEnd', 'Integer', iptr);
24267:   Position', 'Integer', iptrw);
24268:   LineNr', 'Integer', iptr);
24269:   Column', 'Integer', iptr);
24270:   CurrentLine', 'String', iptr);
24271:   Func MatchType( const TokenType : Integer; const Skip :Bool) :Bool');
24272:   Func MatchTokenText(const Token:str;const Skip:Bool;const CaseSensitive:Bool):Bool;
24273:   Func TokenAsInteger : Int64');
24274:  end;
24275: end;
24276:
24277: Proc SIRegister_TRationalClass(CL: TPSPascalCompiler);
24278: begin
24279: //with RegClassS(CL,'TOBJECT', 'TRationalClass') do
24280:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TRationalClass') do begin
24281:   Constructor Create;
24282:   Constructor Create1( const Numerator : Int64; const Denominator: Int64);
24283:   Constructor Create2( const R : Extended);
24284:   Numerator', 'Int64', iptrw);
24285:   Denominator', 'Int64', iptrw);
24286:   AsString', 'String', iptrw);
24287:   AsStringB', 'RawByteString', iptrw);
24288:   AsStringU', 'UnicodeString', iptrw);
24289:   AsFloat', 'MFloat', iptrw);
24290:   Func Duplicate : TRationalClass');
24291:   Proc Assign( const R : TRationalClass);
24292:   Proc Assign1( const R : MFloat);
24293:   Proc Assign2( const Numerator : Int64; const Denominator : Int64);
24294:   Proc AssignZero');
24295:   Proc AssignOne');
24296:   Func IsEqual( const R : TRationalClass) :Bool;
24297:   Func IsEqual1(const Numerator:Int64;const Denominator:Int64):Bool;
24298:   Func IsEqual2( const R : Extended) :Bool;
24299:   Func IsZero :Bool');
24300:   Func IsOne :Bool');
24301:   Proc Add( const R : TRationalClass);
24302:   Proc Add1( const V : Extended);
24303:   Proc Add2( const V : Int64);
24304:   Proc Subtract( const R : TRationalClass);
24305:   Proc Subtract1( const V : Extended);
24306:   Proc Subtract2( const V : Int64);
24307:   Proc Negate');
24308:   Proc Abs');
24309:   Func Sgn : Integer');
24310:   Proc Multiply( const R : TRationalClass);
```

```
24311:   Proc Multiply1( const V : Extended);
24312:   Proc Multiply2( const V : Int64);
24313:   Proc Divide( const R : TRationalClass);
24314:   Proc Divide1( const V : Extended);
24315:   Proc Divide2( const V : Int64);
24316:   Proc Reciprocal');
24317:   Proc Sqrt');
24318:   Proc Sqr');
24319:   Proc Power( const R : TRationalClass);
24320:   Proc Power1( const V : Int64);
24321:   Proc Power2( const V : Extended);
24322:   Proc Exp');
24323:   Proc Ln');
24324:   Proc Sin');
24325:   Proc Cos');
24326:  end;
24327: end;
24328:
24329: Proc SIRegister_TComplexClass(CL: TPSPascalCompiler);
24330: begin
24331:  //with RegClassS(CL,'TOBJECT', 'TComplexClass') do
24332:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TComplexClass') do begin
24333:    Constructor Create( const ARealPart : MFloat; const AImaginaryPart : MFloat);
24334:    RealPart', 'MFloat', iptrw);
24335:    ImaginaryPart', 'MFloat', iptrw);
24336:    AsString', 'String', iptrw);
24337:    AsStringB', 'RawByteString', iptrw);
24338:    AsStringU', 'UnicodeString', iptrw);
24339:    Proc Assign( const C : TComplex);
24340:    Proc Assign1( const V : MFloat);
24341:    Proc AssignZero');
24342:    Proc AssignI');
24343:    Proc AssignMinI');
24344:    Func Duplicate : TComplexClass');
24345:    Func IsEqual2( const C : TComplexClass) :Bool;
24346:    Func IsEqual3( const R, I : MFloat) :Bool;
24347:    Func IsReal :Bool');
24348:    Func IsZero :Bool');
24349:    Func IsI :Bool');
24350:    Proc Add4( const C : TComplex);
24351:    Proc Add5( const V : MFloat);
24352:    Proc Subtract6( const C : TComplexClass);
24353:    Proc Subtract7( const V : MFloat);
24354:    Proc Multiply8( const C : TComplexClass);
24355:    Proc Multiply9( const V : MFloat);
24356:    Proc MultiplyI');
24357:    Proc MultiplyMinI');
24358:    Proc Divide10( const C : TComplexClass);
24359:    Proc Divide11( const V : MFloat);
24360:    Proc Negate');
24361:    Func Modulo : MFloat');
24362:    Func Denom : MFloat');
24363:    Proc Conjugate');
24364:    Proc Inverse');
24365:    Proc Sqrt');
24366:    Proc Exp');
24367:    Proc Ln');
24368:    Proc Sin');
24369:    Proc Cos');
24370:    Proc Tan');
24371:    Proc Power( const C : TComplexClass);
24372:  end;
24373: end;
24374:
24375: Proc SIRegister_TMatrixClass(CL: TPSPascalCompiler);
24376: begin
24377: //with RegClassS(CL,'TOBJECT', 'TMatrixClass') do
24378:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TMatrixClass') do begin
24379:   Constructor CreateSize( const RowCount, ColCount : Integer);
24380:   Constructor CreateSquare( const N : Integer; const Identity :Bool);
24381:   //RegisterMethod('Constructor CreateDiagonal( const D : TVector);
24382:   ColCount', 'Integer', iptrw);
24383:   RowCount', 'Integer', iptrw);
24384:   Proc SetSize( const RowCount, ColCount : Integer);
24385:   Proc Clear');
24386:   Item', 'MFloat Integer Integer', iptrw);
24387:   SetDefaultPropery('Item');
24388:   Proc AssignZero');
24389:   Proc AssignIdentity');
24390:   Proc Assign( const Value : MFloat);
24391:   Proc Assign1( const M : TMatrix);
24392:   //RegisterMethod('Proc Assign2( const V : TVector);
24393:   Func Duplicate : TMatrixClass;
24394:   Func DuplicateRange( const R1, C1, R2, C2 : Integer) : TMatrixClass;
24395:   //RegisterMethod('Func DuplicateRow( const Row : Integer) : TVector');
24396:   //RegisterMethod('Func DuplicateCol( const Col : Integer) : TVector');
24397:   //RegisterMethod('Func DuplicateDiagonal : TVector');
24398:   Func IsEqual( const M : TMatrixClass) :Bool;
24399:   //RegisterMethod('Func IsEqual6( const V : TVector) :Bool;
```

```
24400:   Func IsSquare :Bool');
24401:   Func IsZero :Bool');
24402:   Func IsIdentity :Bool');
24403:   AsString', 'String', iptr);
24404:   AsStringB', 'RawByteString', iptr);
24405:   AsStringU', 'UnicodeString', iptr);
24406:   Func Trace : MFloat');
24407:   //RegisterMethod('Proc SetRow7(const Row: Integer; const V : TVector);
24408:   Proc SetRow( const Row : Integer; const Values : array of MFloat);
24409:   //RegisterMethod('Proc SetCol( const Col: Integer; const V : TVector);
24410:   Func Transpose : TMatrixClass');
24411:   Proc TransposeInPlace');
24412:   Proc Add( const M : TMatrixClass);
24413:   Proc Subtract( const M : TMatrixClass);
24414:   Proc Negate');
24415:   Proc MultiplyRow( const Row : Integer; const Value : MFloat);
24416:   Proc Multiply( const Value : MFloat);
24417:   Func Multiply1( const M : TMatrixClass) : TMatrixClass;
24418:   Proc MultiplyInPlace( const M : TMatrixClass);
24419:   Func IsOrtogonal :Bool');
24420:   Func IsIdempotent :Bool');
24421:   Func Normalise( const M : TMatrixClass) : MFloat');
24422:   Func SolveMatrix( var M : TMatrixClass) : MFloat');
24423:   Func Determinant : MFloat');
24424:   Func Inverse : TMatrixClass');
24425:   Proc InverseInPlace');
24426:   //RegisterMethod('Func SolveLinearSystem( const V : TVector) : TVector');
24427:  end;
24428: end;
24429:
24430: Proc SIRegister_TStringBuilder(CL: TPSPascalCompiler);
24431: begin
24432: //with RegClassS(CL,'TOBJECT', 'TStringBuilder') do
24433:  with CL.AddClassN(CL.FindClass('TOBJECT'),'TStringBuilder') do begin
24434:   Constructor Create( const S :Str);
24435:   Constructor Create2( const Capacity : Integer);
24436:   RegisterProperty('Length', 'Integer', iptr);
24437:   RegisterProperty('AsString', 'String', iptrw);
24438:   Proc Clear');
24439:   Proc Assign( const S : TStringBuilder);
24440:   Proc Append( const S :Str);
24441:   Proc AppendCRLF');
24442:   Proc AppendLn( const S :Str);
24443:   Proc Append1( const S :Str; const Count : Integer);
24444:   Proc AppendCh( const C : Char);
24445:   Proc AppendCh1( const C : Char; const Count : Integer);
24446:   Proc Append2( const S : TStringBuilder);
24447:   Proc Pack');
24448:  end;
24449:  Proc TestStringBuilderClass
24450: end;
24451:
24452: * === compile-time registration functions === *)
24453: (*----------------------------------------------------------------------------*)
24454: Proc SIRegister_flcASCII(CL: TPSPascalCompiler);
24455: begin
24456: 'AsciiNULL','Char').SetString( ByteChar ( #0 ));
24457: 'AsciiSOH','Char').SetString( ByteChar ( #1 ));
24458: 'AsciiSTX','Char').SetString( ByteChar ( #2 ));
24459: 'AsciiETX','Char').SetString( ByteChar ( #3 ));
24460: 'AsciiEOT','Char').SetString( ByteChar ( #4 ));
24461: 'AsciiENQ','Char').SetString( ByteChar ( #5 ));
24462: 'AsciiACK','Char').SetString( ByteChar ( #6 ));
24463: 'AsciiBEL','Char').SetString( ByteChar ( #7 ));
24464: 'AsciiBS','Char').SetString( ByteChar ( #8 ));
24465: 'AsciiHT','Char').SetString( ByteChar ( #9 ));
24466: 'AsciiLF','Char').SetString( ByteChar ( #10 ));
24467: 'AsciiVT','Char').SetString( ByteChar ( #11 ));
24468: 'AsciiFF','Char').SetString( ByteChar ( #12 ));
24469: 'AsciiCR','Char').SetString( ByteChar ( #13 ));
24470: 'AsciiSO','Char').SetString( ByteChar ( #14 ));
24471: 'AsciiSI','Char').SetString( ByteChar ( #15 ));
24472: 'AsciiDLE','Char').SetString( ByteChar ( #16 ));
24473: 'AsciiDC1','Char').SetString( ByteChar ( #17 ));
24474: 'AsciiDC2','Char').SetString( ByteChar ( #18 ));
24475: 'AsciiDC3','Char').SetString( ByteChar ( #19 ));
24476: 'AsciiDC4','Char').SetString( ByteChar ( #20 ));
24477: 'AsciiNAK','Char').SetString( ByteChar ( #21 ));
24478: 'AsciiSYN','Char').SetString( ByteChar ( #22 ));
24479: 'AsciiETB','Char').SetString( ByteChar ( #23 ));
24480: 'AsciiCAN','Char').SetString( ByteChar ( #24 ));
24481: 'AsciiEM','Char').SetString( ByteChar ( #25 ));
24482: 'AsciiEOF','Char').SetString( ByteChar ( #26 ));
24483: 'AsciiESC','Char').SetString( ByteChar ( #27 ));
24484: 'AsciiFS','Char').SetString( ByteChar ( #28 ));
24485: 'AsciiGS','Char').SetString( ByteChar ( #29 ));
24486: 'AsciiRS','Char').SetString( ByteChar ( #30 ));
24487: 'AsciiUS','Char').SetString( ByteChar ( #31 ));
24488: 'AsciiSP','Char').SetString( ByteChar ( #32 ));
```

```
24489: 'AsciiDEL','Char').SetString( ByteChar ( #127 ));
24490: 'AsciiXON','char').SetString( AsciiDC1);
24491: 'AsciiXOFF','char').SetString( AsciiDC3);
24492: 'AsciiDecimalPoint','Char').SetString( ByteChar ( #46 ));
24493: 'AsciiComma','Char').SetString( ByteChar ( #44 ));
24494: 'AsciiBackSlash','Char').SetString( ByteChar ( #92 ));
24495: 'AsciiForwardSlash','Char').SetString( ByteChar ( #47 ));
24496: 'AsciiPercent','Char').SetString( ByteChar ( #37 ));
24497: 'AsciiAmpersand','Char').SetString( ByteChar ( #38 ));
24498: 'AsciiPlus','Char').SetString( ByteChar ( #43 ));
24499: 'AsciiMinus','Char').SetString( ByteChar ( #45 ));
24500: 'AsciiEqualSign','Char').SetString( ByteChar ( #61 ));
24501: 'AsciiSingleQuote','Char').SetString( ByteChar ( #39 ));
24502: 'AsciiDoubleQuote','Char').SetString( ByteChar ( #34 ));
24503: 'AsciiDigit0','Char').SetString( ByteChar ( #48 ));
24504: 'AsciiDigit9','Char').SetString( ByteChar ( #57 ));
24505: 'AsciiUpperA','Char').SetString( ByteChar ( #65 ));
24506: 'AsciiUpperZ','Char').SetString( ByteChar ( #90 ));
24507: 'AsciiLowerA','Char').SetString( ByteChar ( #97 ));
24508: 'AsciiLowerZ','Char').SetString( ByteChar ( #122 ));
24509: 'WideNULL','Char').SetString( WideChar ( #0 ));
24510: 'WideSOH','Char').SetString( WideChar ( #1 ));
24511: 'WideSTX','Char').SetString( WideChar ( #2 ));
24512: 'WideETX','Char').SetString( WideChar ( #3 ));
24513: 'WideEOT','Char').SetString( WideChar ( #4 ));
24514: 'WideENQ','Char').SetString( WideChar ( #5 ));
24515: 'WideACK','Char').SetString( WideChar ( #6 ));
24516: 'WideBEL','Char').SetString( WideChar ( #7 ));
24517: 'WideBS','Char').SetString( WideChar ( #8 ));
24518: 'WideHT','Char').SetString( WideChar ( #9 ));
24519: 'WideLF','Char').SetString( WideChar ( #10 ));
24520: 'WideVT','Char').SetString( WideChar ( #11 ));
24521: 'WideFF','Char').SetString( WideChar ( #12 ));
24522: 'WideCR','Char').SetString( WideChar ( #13 ));
24523: 'WideSO','Char').SetString( WideChar ( #14 ));
24524: 'WideSI','Char').SetString( WideChar ( #15 ));
24525: 'WideDLE','Char').SetString( WideChar ( #16 ));
24526: 'WideDC1','Char').SetString( WideChar ( #17 ));
24527: 'WideDC2','Char').SetString( WideChar ( #18 ));
24528: 'WideDC3','Char').SetString( WideChar ( #19 ));
24529: 'WideDC4','Char').SetString( WideChar ( #20 ));
24530: 'WideNAK','Char').SetString( WideChar ( #21 ));
24531: 'WideSYN','Char').SetString( WideChar ( #22 ));
24532: 'WideETB','Char').SetString( WideChar ( #23 ));
24533: 'WideCAN','Char').SetString( WideChar ( #24 ));
24534: 'WideEM','Char').SetString( WideChar ( #25 ));
24535: 'WideEOF','Char').SetString( WideChar ( #26 ));
24536: 'WideESC','Char').SetString( WideChar ( #27 ));
24537: 'WideFS','Char').SetString( WideChar ( #28 ));
24538: 'WideGS','Char').SetString( WideChar ( #29 ));
24539: 'WideRS','Char').SetString( WideChar ( #30 ));
24540: 'WideUS','Char').SetString( WideChar ( #31 ));
24541: 'WideSP','Char').SetString( WideChar ( #32 ));
24542: 'WideDEL','Char').SetString( WideChar ( #127 ));
24543: 'WideXON','char').SetString( WideDC1);
24544: 'WideXOFF','char').SetString( WideDC3);
24545: 'WideCRLF','String').SetString( WideString ( #13#10 ));
24546:  Func IsAsciiCharB( const C : Char) :Bool');
24547:  Func IsAsciiCharW( const C : WideChar) :Bool');
24548:  Func IsAsciiChar( const C : Char) :Bool');
24549:  //Func IsAsciiBufB( const Buf: Pointer; const Len: NativeInt):Bool');
24550:  //Func IsAsciiBufW( const Buf: Pointer; const Len  NativeInt):Bool');
24551:  Func IsAsciiStringB( const S :Str) :Bool');
24552:  Func IsAsciiStringU( const S : UnicodeString) :Bool');
24553:  Func IsAsciiString( const S :Str) :Bool');
24554:  Func AsciiHexCharValueB( const C : Char) : Integer');
24555:  Func AsciiHexCharValueW( const C : WideChar) : Integer');
24556:  Func AsciiIsHexCharB( const C : Char) :Bool');
24557:  Func AsciiIsHexCharW( const C : WideChar) :Bool');
24558:  Func AsciiDecimalCharValueB( const C : Char) : Integer');
24559:  Func AsciiDecimalCharValueW( const C : WideChar) : Integer');
24560:  Func AsciiIsDecimalCharB( const C : Char) :Bool');
24561:  Func AsciiIsDecimalCharW( const C : WideChar) :Bool');
24562:  Func AsciiOctalCharValueB( const C : Char) : Integer');
24563:  Func AsciiOctalCharValueW( const C : WideChar) : Integer');
24564:  Func AsciiIsOctalCharB( const C : Char) :Bool');
24565:  Func AsciiIsOctalCharW( const C : WideChar) :Bool');
24566:  Func CharCompareNoAsciiCaseB( const A, B : Char) : Integer');
24567:  Func CharCompareNoAsciiCaseW( const A, B : WideChar) : Integer');
24568:  Func CharCompareNoAsciiCase( const A, B : Char) : Integer');
24569:  Func CharEqualNoAsciiCaseB( const A, B : Char) :Bool');
24570:  Func CharEqualNoAsciiCaseW( const A, B : WideChar) :Bool');
24571:  Func CharEqualNoAsciiCase( const A, B : Char) :Bool');
24572:  //Func StrPCompareNoAsciiCaseB( const A, B: PByteChar; const Len: NativeInt):Integer');
24573:  //Func StrPCompareNoAsciiCaseW( const A, B: PWideChar; const Len: NativeInt):Integer');
24574:  Func StrPCompareNoAsciiCase( const A, B : PChar; const Len : NativeInt) : Integer');
24575:  Func StrCompareNoAsciiCaseA( const A, B : Ansistr) : Integer');
24576:  Func StrCompareNoAsciiCaseB( const A, B :Str) : Integer');
24577:  Func StrCompareNoAsciiCaseU( const A, B : UnicodeString) : Integer');
```

```
24578:  Func StrCompareNoAsciiCase( const A, B :Str) : Integer');
24579:  Func AsciiLowCaseB( const C : Char) : Char');
24580:  Func AsciiLowCaseW( const C : WideChar) : WideChar');
24581:  Func AsciiLowCase( const C : Char) : Char');
24582:  Func AsciiUpCaseB( const C : Char) : Char ');
24583:  Func AsciiUpCaseW( const C : WideChar) : WideChar');
24584:  Func AsciiUpCase( const C : Char) : Char');
24585:  Proc AsciiConvertUpperB( var S :Str);
24586:  Proc AsciiConvertUpperU( var S : UnicodeString);
24587:  Proc AsciiConvertUpper( var S :Str);
24588:  Proc AsciiConvertLowerB( var S :Str);
24589:  Proc AsciiConvertLowerU( var S : UnicodeString);
24590:  Proc AsciiConvertLower( var S :Str);
24591:  Func AsciiUpperCaseB( const A :Str) :Str');
24592:  Func AsciiUpperCaseU( const A : UnicodeString) : UnicodeString');
24593:  Func AsciiUpperCase( const A :Str) :Str');
24594:  Func AsciiLowerCaseB( const A :Str) :Str');
24595:  Func AsciiLowerCaseU( const A : UnicodeString) : UnicodeString');
24596:  Func AsciiLowerCase( const A :Str) :Str');
24597:  Proc AsciiConvertFirstUpB( var S : RawByteString);
24598:  Proc AsciiConvertFirstUpU( var S : UnicodeString);
24599:  Proc AsciiConvertFirstUp( var S :Str);
24600:  Func AsciiFirstUpB( const S : RawByteString) : RawByteString');
24601:  Func AsciiFirstUpU( const S : UnicodeString) : UnicodeString');
24602:  Func AsciiFirstUp( const S :Str) :Str');
24603:  Proc AsciiConvertArrayUpper( var S : TStringArray);
24604:  Proc AsciiConvertArrayLower( var S : TStringArray);
24605:  Proc TestAscii');
24606: end;
24607:
24608: Proc SIRegister_flcStringPatternMatcher(CL: TPSPascalCompiler);
24609: begin
24610:  TMatchPatternGreed', '( mpgLazy, mpgGreedy, mpgGreedyNoBacktrack);
24611:  //CL.AddDelphiFunction('Func StrZMatchPatternA(M, S: PAnsiChar; const G : TMatchPatternGreed): Integer');
24612:  //CL.AddDelphiFunction('Func StrZMatchPatternW(M, S: PWideChar; const G : TMatchPatternGreed): Integer');
24613:  Func StrEqualPatternA(const M,S: RawByteString; const G : TMatchPatternGreed) : Bool;
24614:  Func StrEqualPatternU(const M,S: UnicodeString; const G : TMatchPatternGreed) : Bool;
24615:  Func StrEqualPattern( const M, S :Str; const G : TMatchPatternGreed) :Bool');
24616:  Func StrPosPatternA(const F,S:RawByteString; var Len:Int;const StartIndex:Int;const
        G:TMatchPatternGreed):Int;
24617:  Func StrPosPatternU(const F,S:UnicodeString;var Len:Int;const StartIndex:Int const
        G:TMatchPatternGreed):Int;
24618:  Func MatchFileMaskB(const Mask,Key:RawByteString;const AsciiCaseSensitive:Bool):Bool);
24619:   TMatchQuantifier','( mqOnce, mqAny, mqLeastOnce, mqOptional );
24620:   TMatchQuantSeqOptions', '(moDeterministic, moNonGreedy );
24621:  //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');
24622:  Proc TestPatternmatcher');
24623:  Func CharMatch(const A, B: Char; const AsciiCaseSensitive:Bool):Bool;
24624: end;
24625:
24626: https://github.com/fundamentalslib/fundamentals5/blob/master/Source/Utils/flcStrings.pas
24627: https://github.com/fundamentalslib/fundamentals5/blob/master/Source/Utils/flcUtils.pas
24628: https://raw.githubusercontent.com/fundamentalslib/fundamentals5/master/Source/Utils/flcUnicodeChar.pas
24629: 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).
24630: 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.
24631: 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.
24632: 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:
24633:
24634: Func HexToBits(C: Char): Byte;
24635: begin
24636:   case C of
24637:     '0'..'9': Result:= Byte(Ord(C) - Ord('0'));
24638:     'a'..'f': Result:= Byte(10 + (Ord(C) - Ord('a')));
24639:     'A'..'F': Result:= Byte(10 + (Ord(C) - Ord('A')));
24640:   else
24641:     raise Exception.Create('Invalid encoding detected');
24642:   end;
24643: end;
24644:
24645: https://stackoverflow.com/questions/14107717/decode-utf-8-encoded-cyrillic-with-delphi-2007
24646: (* === compile-time registration functions === *)
24647: (*----------------------------------------------------------------------------*)
24648: Proc SIRegister_flcUnicodeChar(CL: TPSPascalCompiler);
24649: begin
24650:  WideSingleQuote','String').SetString( WideChar ( '''' ));
24651:  WideDoubleQuote','String').SetString( WideChar ( '"' ));
24652:  WideNoBreakSpace','Char').SetString( WideChar ( #$00A0 ));
24653:  'WideLineSeparator','Char').SetString( WideChar ( #$2028 ));
```

```
24654:  'WideParagraphSeparator','Char').SetString( WideChar ( #$2029 ));
24655:  WideBOM_MSB_First','Char').SetString( WideChar ( #$FFFE ));
24656:  WideBOM_LSB_First','Char').SetString( WideChar ( #$FEFF ));
24657:  WideObjectReplacement','Char).SetString( WideChar ( #$FFFC ));
24658:  WideCharReplacement','Char).SetString( WideChar ( #$FFFD ));
24659:  WideInvalid','Char).SetString( WideChar ( #$FFFF ));
24660:  WideCopyrightSign','Char).SetString( WideChar ( #$00A9 ));
24661:  WideRegisteredSign','Char).SetString( WideChar ( #$00AE ));
24662:  WideHighSurrogateFirst','Char).SetString( WideChar ( #$D800 ));
24663:  WideHighSurrogateLast','Char).SetString( WideChar ( #$DB7F ));
24664:  WideLowSurrogateFirst','Char).SetString( WideChar ( #$DC00 ));
24665:  WideLowSurrogateLast','Char).SetString( WideChar ( #$DFFF ));
24666:  WidePrivateHighSurrogateFirst',Char').SetString( WideChar ( #$DB80 ));
24667:  WidePrivateHighSurrogateLast',Char').SetString( WideChar ( #$DBFF ));
24668:  flcWideNULL','Char).SetString( WideChar ( #0 ));
24669:  flcWideBS','Char).SetString( WideChar ( #8 ));
24670:  flcWideHT','Char).SetString( WideChar ( #9 ));
24671:  'flcWideLF','Char').SetString( WideChar ( #10 ));
24672:  'flcWideVT','Char').SetString( WideChar ( #11 ));
24673:  'flcWideFF','Char').SetString( WideChar ( #12 ));
24674:  'flcWideCR','Char').SetString( WideChar ( #13 ));
24675:  'flcWideEOF','Char').SetString( WideChar ( #26 ));
24676:  'flcWideSP','Char').SetString( WideChar ( #32 ));
24677:  'UCS4_STRING_TERMINATOR','LongWord').SetUInt( $9C);
24678:  'UCS4_LF','LongWord').SetUInt( $0A);
24679:  'UCS4_CR','LongWord').SetUInt( $0D);
24680: Func UnicodeIsAsciiChar( const Ch : WideChar) :Bool');
24681: CL.AddTypeS('TUnicodeSpaceWidth','(uswNone,uswZero,uswAdjustableEmOneOver'
24682:  +'Four,uswEmOneOverTwo,uswEmOneOverThree,uswEmOneOverFour,uswEmFourOverEighteen,uswEmOneOverFive,
       uswEmOneOverSix,uswEmOne,uswWidthDigit,uswWidthPeriod,uswWidthIdeographic);
24683: Func UnicodeSpaceWidth( const Ch : WideChar) : TUnicodeSpaceWidth');
24684: Func UnicodeIsNoBreakSpace( const Ch : WideChar) :Bool');
24685: Func UnicodeIsSpaceFixedWidth( const Ch : WideChar) :Bool');
24686: Func UnicodeIsSpaceAdjustableWidth( const Ch : WideChar) :Bool');
24687: Func UnicodeIsSpace( const Ch : WideChar) :Bool');
24688: Func UnicodeIsLineBreak( const Ch : WideChar) :Bool');
24689: Func UnicodeIsWhiteSpace( const Ch : WideChar) :Bool');
24690: Func UnicodeIsControl( const Ch : WideChar) :Bool');
24691: Func UnicodeIsControlOrWhiteSpace( const Ch : WideChar) :Bool');
24692: Func UnicodeIsIgnorable( const Ch : UCS4Char) :Bool');
24693: Func UnicodeIsDash( const Ch : WideChar) :Bool');
24694: Func UnicodeIsHyphen( const Ch : WideChar) :Bool');
24695: Func UnicodeIsDashOrHyphen( const Ch : WideChar) :Bool');
24696: Func UnicodeIsFullStop0( const Ch : UCS4Char) :Bool;
24697: Func UnicodeIsFullStop1( const Ch : WideChar) :Bool;
24698: Func UnicodeIsComma( const Ch : WideChar) :Bool');
24699: Func UnicodeIsExclamationMark( const Ch : WideChar) :Bool');
24700: Func UnicodeIsQuestionMark( const Ch : WideChar) :Bool');
24701: Func UnicodeIsLeftParenthesis( const Ch : WideChar) :Bool');
24702: Func UnicodeIsLeftBracket( const Ch : WideChar) :Bool');
24703: Func UnicodeGetRightParenthesis( const LeftParenthesis : WideChar) : WideChar');
24704: Func UnicodeGetRightBracket( const LeftBracket : WideChar) : WideChar');
24705: Func UnicodeIsSingularQuotationMark( const Ch : WideChar) :Bool');
24706: Func UnicodeIsOpeningQuotationMark( const Ch : WideChar) :Bool');
24707: Func UnicodeIsClosingQuotationMark( const Ch : WideChar) :Bool');
24708: Func UnicodeGetClosingQuotationMark( const OpeningQuote : WideChar) : WideChar');
24709: Func UnicodeGetOpeningQuotationMark( const ClosingQuote : WideChar) : WideChar');
24710: Func UnicodeIsPunctuation( const Ch : WideChar) :Bool');
24711: Func UnicodeIsDecimalDigit2( const Ch : UCS4Char) :Bool;
24712: Func UnicodeIsDecimalDigit3( const Ch : WideChar) :Bool;
24713: Func UnicodeIsAsciiDecimalDigit( const Ch : WideChar) :Bool');
24714: Func UnicodeDecimalDigitValue4( const Ch : UCS4Char) : Integer;
24715: Func UnicodeDecimalDigitValue5( const Ch : WideChar) : Integer;
24716: Func UnicodeFractionCharacterValue( const Ch : WideChar; var A, B :Integer):Bool');
24717: Func UnicodeRomanNumeralValue( const Ch : WideChar) : Integer');
24718: Func UnicodeIsAsciiHexDigit( const Ch : WideChar) :Bool');
24719: Func UnicodeIsHexDigit6( const Ch : UCS4Char) :Bool;
24720: Func UnicodeIsHexDigit7( const Ch : WideChar) :Bool;
24721: Func UnicodeHexDigitValue8( const Ch : UCS4Char) : Integer;
24722: Func UnicodeHexDigitValue9( const Ch : WideChar) : Integer;
24723: Func UnicodeIsUpperCase( const Ch : WideChar) :Bool');
24724: Func UnicodeIsLowerCase( const Ch : WideChar) :Bool');
24725: Func UnicodeIsTitleCase( const Ch : WideChar) :Bool');
24726: Func UnicodeIsLetter( const Ch : WideChar) :Bool');
24727: Func UnicodeIsAlphabetic( const Ch : WideChar) :Bool');
24728: Func UnicodeUpCase( const Ch : WideChar) : WideChar');
24729: Func UnicodeLowCase( const Ch : WideChar) : WideChar');
24730: Func UnicodeCharIsEqualNoCase( const A, B : WideChar) :Bool');
24731: Func UnicodeGetCombiningClass( const Ch : WideChar) : Byte');
24732: Func UnicodeLocateFoldingUpperCase( const Ch : WideChar) : UnicodeString');
24733: Func UnicodeLocateFoldingTitleCase( const Ch : WideChar) : UnicodeString');
24734: Func UnicodeUpCaseFoldingU( const Ch : WideChar) : UnicodeString');
24735: Func UnicodeLowCaseFoldingU( const Ch : WideChar) : UnicodeString');
24736: Func UnicodeTitleCaseFoldingU( const Ch : WideChar) : UnicodeString');
24737: Func UnicodeGetCharacterDecompositionU10( const Ch : WideChar) : UnicodeString;
24738: Func UnicodeGetCharacterDecompositionU11( const Ch : UCS4Char) : UnicodeString;
24739: Func UnicodeUpperCaseFoldingU( const S : UnicodeString) : UnicodeString');
24740: Func UnicodeLowerCaseFoldingU( const S : UnicodeString) : UnicodeString');
24741: Proc TestUnicodeChar');
```

```
24742: Func HexToBits(C: Char): Byte;
24743: Func Utf8ToStr(const Source :Str) :Str;); end;
24744: unit PJResFile_Routines;
24745: Func IsIntResource( const ResID : PChar) :Bool');
24746: Func IsEqualResID2( const R1, R2 : PChar) :Bool');
24747: Func ResIDToStr( const ResID : PChar) :Str');
24748:
24749: 15 CLF_Fundamentals Testroutines 47520
24750: ------------------------------------
24751:    1 TestMathClass;
24752:    2 TestStatisticClass;
24753:    3 TestBitClass;
24754:    4 TestCharset;
24755:    5 TestTimerClass
24756:    6 TestRationalClass
24757:    7 TestComplexClass
24758:    8 TestMatrixClass;
24759:    9 TestStringBuilderClass
24760:    10 TestASCII;
24761:    11 TestASCIIRoutines;
24762:    12 TestPatternmatcher;
24763:    13 TestUnicodeChar;
24764:    14 unit uPSI_AfUtils;
24765:    15 unit uPSI_PsAPI;
24766:    16 TestVectorClass;
24767:    17 TestVectorClassExtended;
24768:    18 TestKMeans;
24769:    19 TestKMeans2;
24770:    20 TestTNNetVolume;
24771:    21 TestConvolutionAPI();
24772:    22 TestDataParallelism(NN: TNNet);
24773:    //TestTNNetVolume;
24774:
24775: ****************************************************************
24776: Release Notes maXbox 4.7.5.20 Jan 2021 mX47
24777: ****************************************************************
24778: Add 25 Units + 4 Tutorials
24779: 1277 unit uPSI_SystemsDiagram.pas  Dendron
24780: 1278 unit uPSI_qsFoundation.pas    Dendron
24781: 1279 uPSI_JclStringLists2          JCL
24782: 1280 uPSI_cInternetUtils2          FLC
24783: 1281 uPSI_cWindows.pas            FLC
24784: 1282 uPSI_flcSysUtils.pas  +TBytes utils
24785: 1283 unit uPSI_RotImg.pas          DA
24786: 1284 uPSI_SimpleImageLoader.pas    LAZ
24787: 1285 uPSI_HSLUtils.pas             LAZ
24788: 1286 uPSI_GraphicsMathLibrary.pas  EF
24789: 1287 unit uPSI_umodels.pas         DMath
24790: 1288 uPSI_flcStatistics.pas        FLC5
24791: 1289 uPSI_flcMaths.pas             FLC5
24792: 1290 uPSI_flcCharSet.pas
24793: 1291 uPSI_flcBits32.pas
24794: 1292 uPSI_flcTimers.pas
24795: 1293 uPSI_cBlaiseParserLexer.pas
24796: 1294 uPSI_flcRational.pas
24797: 1295 uPSI_flcComplex.pas
24798: 1296 unit uPSI_flcMatrix (uPSI_flcVectors.pas)
24799: 1297 unit uPSI_flcStringBuilder.pas
24800: 1298 unit PJResFile_Routines;
24801: 1299 uPSI_flcASCII.pas
24802: 1300 uPSI_flcStringPatternMatcher;
24803: 1301 unit uPSI_flcUnicodeChar.pas
24804: Totals of Func Calls: 33282
24805: SHA1: of 4.7.5.20 D82EAD01C58738887661428F94B207DB1D8FAEB5
24806: CRC32: 203C82F0 29.5 MB (31,012,768 bytes
24807: newunits: 01 RotImg.pas + uModel : TModel lib dmath.dll
24808:           02 SimpleImageLoader.pas
24809:           03 systemsdiagram.pas + fpc switch
24810:           04 qsfoundation.pas   NO FPC Vector operator
24811:           05 prediction.pas  SimulationEngine missing
24812:           06 HSLUtils.pas - color model
24813:           07 cInternetUtils.pas  - header information
24814:           08 cWindows.pas  - cstrings routines as flcSysUtils
24815:           09 flcSysUtils.pas  //2 functions with cwindows possible+ freqObj +TBytes utils
24816:           10 GraphicsMathLibrary.PAS  gml-prefix
24817:           11 flcBits32.pas  //$IFDEF DEBUG} {$IFDEF TEST} Proc Test;
24818:           12 flcFloats.pas  No Floats instead: uPSI_cBlaiseParserLexer.pas
24819:           13 flcDecimal.pas  + TestClass
24820:           14 flcCharSet.pas //test include wrapper + -
24821:           15 flcComplex.pas     -Class
24822:           16 flcMaths.pas    //{$IFDEF MATHS_TEST}  Proc Test
24823:           17 flcMatrix.pas    - less TVectors
24824:           18 flcRational.pas    -Class
24825:           19 flcStatistics.pas  -Class
24826:           20 flcStringBuilder.pas - less Unicode
24827:           21 flcVectors.pas   No Vectors cause compatibility
24828:           22 flcTimers.pas     {$DEFINE TIMERS_TEST}
24829:
24830: ****************************************************************
```

```
24831: Release Notes maXbox 4.7.5.90 October 2021 mX475
24832: ****************************************************************
24833:
24834: Add 26 Units + 10 Tutorials
24835: 1403 unit uPSI_SemaphorGrids;
24836: 1404 unit uXmlDates2;
24837: 1405 unit uPSI_JclTimeZones;
24838: 1406 unit uPSI_XmlDocRssParser.pas
24839: 1407 unit uPSI_RssParser.pas
24840: 1408 uPSI_SimpleParserRSS.pas
24841: 1409 unit uPSI_SimpleRSSUtils;
24842: 1410 unit uPSI_RssModel;      _BlueHippo
24843: 1411 unit uPSI_StrUtil;       _FIBPlus
24844: 1412 unit uPSI_TAChartUtils; _TEE
24845: 1413 unit uPSI_PythonEngine.pas _P4D_Beta
24846: 1414 unit uPSI_VclPythonGUIInputOutput;
24847: 1415 unit uPSI_VarPyth;
24848: 1416 unit JclUsesUtils;
24849: 1417 unit uPSI_cParameters;
24850: 1418 unit uPSI_WDCCMisc; (uPSI_cFileTemplates);
24851: 1419 uPSI_WDCCOleVariantEnum.pas
24852: 1420 unit uPSI_WDCCWinInet.pas  _WDCC
24853: 1421 uPSI_PythonVersions.pas
24854: 1422 unit uPSI_PythonAction.pas
24855: 1423 uPSI_SingleList.pas
24856: 1424 unit uPSI_AdMeter.pas; Async Professional
24857: 1425 unit uPSI_neuralplanbuilder; CAI
24858: 1426 unit uPSI_neuralvolume.pas; CAI
24859: 1427 unit uPSI_neuralvolumev.pas; CAI
24860: 1428 unit uPSI_DoubleList4;  CapJack
24861: 1429 unit uPSI_ByteListClass;  CapJack
24862: 1430 unit uPSI_flcVectors4; FLC5
24863: 1431 unit uPSI_flcMatrix4;
24864: 1432 uPSI_CurlHttpCodes.pas
24865: 1433 unit uPSI_NeuralNetwork.pas; CAI
24866: 1434 unit unit uPSI_neuralfit; CAI
24867: 1435 unit uPSI_neuraldatasets; CAI
24868: 1436 unit uPSI_neuraldatasetsv.pas CAI
24869: 1437 uPSI_flcFloats.pas FLC5
24870:
24871: //Async Professional
24872:
24873: SIRegister_TFastStringStream(CL);
24874: Proc LoadJPEGResource(image1: TImage; aJpgImage:Str);
24875: Func CreateDOSProcessRedirected3(const CommandLine, InputFile, OutputFile, ErrMsg:Str):Bool;
24876: //Type TSysCharSet = set of Char;
24877: Func SysCharSetToStr(const C: TSysCharSet): Ansistr;
24878: Func StrToSysCharSet(const S: Ansistr): TSysCharSet;
24879: Proc MaskFPUExceptions( ExceptionsMasked :Bool; MatchPythonPrecision :Bool);
24880: Func GetOleVariantEnum( Collection : OLEVariant) : IGetOleVariantEnum');
24881: Func GetOleVariantArrEnum( Collection : OLEVariant) : IGetOleVariantEnum');
24882:
24883:
24884: Proc SIRegister_TSimpleParserRSS(CL: TPSPascalCompiler);
24885: begin
24886:   //with RegClassS(CL,'TSimpleParserBase', 'TSimpleParserRSS') do
24887:   with CL.AddClassN(CL.FindClass('TSimpleParserBase'),'TSimpleParserRSS') do begin
24888:     RegisterMethod('Proc Parse');
24889:     RegisterMethod('Proc Generate');
24890:   end;
24891: end;
24892:
24893:  SERIES_MARK_FORMATS: array [TSeriesMarksStyle] of String = (
24894:     '', '',
24895:     '%0:.9g', // smsValue
24896:     '%1:.2f%%', // smsPercent
24897:     '%2:s', // smsLabel
24898:     '%2:s %1:.2f%%', // smsLabelPercent
24899:     '%2:s %0:.9g', // smsLabelValue
24900:     '%2:s', // smsLegend: not sure what it means, left for Delphi compatibility
24901:     '%1:.2f%% of %3:g', // smsPercentTotal
24902:     '%1:.2f%% of %3:g', // smsLabelPercentTotal
24903:     '%4:.9g' // smsXValue
24904:   );
24905:
24906: Proc SIRegister_uXmlDates(CL: TPSPascalCompiler);
24907: begin
24908:   CL.AddTypeS('TXMLDateZeroOptions', '( dzoBlank, dzoZero, dzoNow );
24909:  Func GetTimeZoneOffset :Str');
24910:  Func GetTimestampWithTimeZone( const ADateTime : TDateTime) :Str');
24911:  Func GetXmlDate( const ADateTime : TDateTime; const DateZeroOption : TXMLDateZeroOptions) :Str');
24912:  Func ConvertToDelphiDateFromXml( const ADateTime :Str) : TDateTime');
24913: end;
24914:
24915: Proc SIRegister_JclTimeZones(CL: TPSPascalCompiler);
24916: begin
24917:   CL.AddTypeS('TJclTZIValueInfo', 'record Bias : Longint; StandardBias : Intege'
24918:    +'r; DaylightBias : Integer; StandardDate : TSystemTime; DaylightDate : TSystemTime; end');
24919:   //CL.AddTypeS('PJclTimeZoneRegInfo', '^TJclTimeZoneRegInfo // will not work');
```

```
24920:   TJclTimeZoneRegInfo', 'record DisplayDesc :Str; StandardName'
24921:    + :Str; DaylightName :Str; SortIndex : Integer; MapID :Str; TZI: TJclTZIValueInfo; end');
24922:   TJclTimeZoneCallBackFunc', 'Func ( const TimeZoneRec : TJclTimeZoneRegInfo) :Bool');
24923:   SIRegister_TJclTimeZoneInfo(CL);
24924:   SIRegister_TJclTimeZones(CL);
24925:   CL.AddClassN(CL.FindClass('TOBJECT'),'EDaylightSavingsNotSupported');
24926:   CL.AddClassN(CL.FindClass('TOBJECT'),'EAutoAdjustNotEnabled');
24927:  Func EnumTimeZones( CallBackFunc : TJclTimeZoneCallBackFunc) :Bool');
24928:  Func IsAutoAdjustEnabled :Bool');
24929:  Func CurrentTimeZoneSupportsDaylightSavings :Bool');
24930:  Func DateCurrentTimeZoneClocksChangeToStandard : TDateTime');
24931:  Func DateCurrentTimeZoneClocksChangeToDaylightSavings : TDateTime');
24932:  Func GetCurrentTimeZoneDescription :Str');
24933:  Func GetCurrentTimeZoneDaylightSavingsPeriod :Str');
24934:  Func GetCurrentTimeZoneGMTOffset :Str');
24935:  Func GetCurrentTimeZoneUTCBias : Integer');
24936:  Func GetWMIScheduledJobUTCTime( Time : TDateTime) :Str');
24937:  Func UTCNow : TDateTime');
24938: end;
24939:
24940: type IRSSParser = interface
24941:     ['{263159EC-F94D-4BA6-9D34-5D93277D4DDC}']
24942:     Func ParseRSSFeed(XML:Str): TRSSFeed;
24943:   end;
24944:
24945: Proc SIRegister_TXmlDocRssParser(CL: TPSPascalCompiler);
24946: begin
24947:   //with RegClassS(CL,'TInterfacedObject', 'TXmlDocRssParser') do
24948:   with CL.AddClassN(CL.FindClass('TInterfacedObject'),'TXmlDocRssParser') do begin
24949:     RegisterMethod('Func ParseRSSDate( DateStr :Str) : TDateTime');
24950:     RegisterMethod('Func ParseRSSFeed( XML :Str) : TRSSFeed');
24951:   end;
24952: end;
24953:
24954: (*----------------------------------------------------------------------------*)
24955: Proc SIRegister_RssParser(CL: TPSPascalCompiler);
24956: begin
24957:  Func ParseRSSDate( DateStr :Str) : TDateTime');
24958:  Func ParseRSSFeed( XML :Str) : TRSSFeed');
24959: end;
24960:
24961: unit SimpleRSSUtils;
24962: type TXMLTypeRSS = (xtRDFrss, xtRSSrss, xtAtomrss, xtiTunesrss);
24963: type TContentTypeRSS = (ctTextrss, ctHTMLrss, ctXHTMLrss);
24964: type TEncodingTypeRSS = (etBase64rss,etEscapedrss,etXMLrss);
24965:
24966: Func StringToLanguage(Language:Str): TLanguages;
24967: Func LanguageToString(Language: TLanguages):str;
24968: Func DecodeStringRSS(Encoding: TEncodingTypeRSS; Data:Str):Str;
24969: Proc GetSkipHours(RootNode: IXMLNode; var aSimpleRSS: TSimpleRSS);
24970: Proc GetSkipDays(RootNode: IXMLNode; var aSimpleRSS: TSimpleRSS);
24971: Proc SIRegister_SimpleRSSUtils(CL: TPSPascalCompiler);
24972:  Func StringToLanguage( Language :Str) : TLanguages');
24973:  Func LanguageToString( Language : TLanguages) :Str');
24974:  Func DecodeStringRSS( Encoding : TEncodingTypeRSS; Data :Str) :Str');
24975:  Proc GetSkipHours( RootNode : IXMLNode; var aSimpleRSS : TSimpleRSS);
24976:  Proc GetSkipDays( RootNode : IXMLNode; var aSimpleRSS : TSimpleRSS);
24977:  Func GetPageByURL(URL:Str):Str;
24978: end;
24979:
24980: (*----------------------------------------------------------------------------*)
24981: Proc SIRegister_StrUtil(CL: TPSPascalCompiler);
24982: begin
24983:   CL.AddTypeS('_TPosition', 'record X : Longint; Y : Longint; end');
24984:   CL.AddTypeS('TFastStringStream', 'TStringStream');
24985:   SIRegister_TFastStringStream(CL);
24986:   CL.AddTypeS('_TCharSet', 'set of AnsiChar');
24987:  Func _Position( AX, AY : Integer) : _TPosition');
24988:  Func _MakeStr( C : Char; N : Integer) :Str');
24989:  Func _StrAsFloat( S :Str) : double');
24990:  Func _ToClientDateFmt( D :Str; caseFmt : byte) :Str');
24991:  Func _ExtractWord( Num : integer; const Str :Str; const WordDelims : TCharSet):Str');
24992:  Func _WordCount( const S :Str; const WordDelims : TCharSet) : Integer');
24993:  Func _AnsiUpperCaseA( const s : Ansistr) : Ansistr');
24994:  Func _CompareStrWA( const S1 :Str; const S2 : Ansistr) : Integer');
24995:  Func _AnsiCompareTextAA( const S1, S2 : Ansistr) : Integer');
24996:  Func _WildStringCompare( FirstString, SecondString :Str) :Bool');
24997:  Func _MaskCompare( const aString, Mask :Str) :Bool');
24998:  Func _SQLMaskCompare( const aString, Mask :Str) :Bool;
24999:  Func _SQLMaskCompareAW( const aString : Ansistr; const Mask :Str) :Bool');
25000:  Func _SQLMaskCompareAA( const aString, Mask : Ansistr) :Bool');
25001:  Func _SQLMaskCompare1( const aString, Mask : Widestring) :Bool;
25002:  Func _TrimCLRF( const s :Str) :Str');
25003:  Func _ReplaceStr( const S, Srch, Replace :Str) :Str');
25004:  Func _ReplaceStrInSubstr( const Source String; const Srch, Replace:str;BeginPos, EndPos:integer):str');
25005:  Func _ReplaceCIStr( const S, Srch, Replace:Str; Ansi :Bool):str');
25006:  Func _ReplaceStrCIInSubstr(const Source:str; const Srch,Replace:str; BeginPos,EndPos:integer):str');
25007:  Func _FastUpperCase( const S :Str) :Str');
25008:  Proc _DoUpperCase( var S :Str; FirstPos : integer; LastPos : integer);
```

```
25009:  Proc _DoAnsiUpperCase( var S : Ansistr);
25010:  Proc _DoAnsiUpperCase1( var S :Str);
25011:  Proc _DoWideUpperCase( var S : WideString);
25012:  Proc _DoUtf8Decode( const s : AnsiChar; StrLen : integer; var ws: WideString);
25013:  Proc _DoUtf8Decode1( const s : Ansistr; var ws : WideString);
25014:  Proc _DoUtf8Decode2( const s : variant; var ws : WideString);
25015:  Func _EquelStrings( const s, s1 :Str; CaseSensitive :Bool) :Bool');
25016:  Func _iifStr( Condition :Bool; const Str1, Str2 :Str) :Str');
25017:  Func _iifVariant( Condition :Bool; const Var1, Var2 : Variant) : Variant');
25018:  Func _StrOnMask( const StrIn, MaskIn, MaskOut :Str) :Str');
25019:  Func _StrIsInteger( const Str :Str) :Bool');
25020:  Func _FormatIdentifierValue( Dialect : Integer; const Value :Str) :Str');
25021:  Func _FormatIdentifier( Dialect : Integer; const Value :Str) :Str');
25022:  Func _NormalizeName( Dialect : Integer; const Name :Str) :Str');
25023:  Func _EasyFormatIdentifier( Dialect : Integer; const Value :Str; DoEasy :Bool) :Str');
25024:  Func _EquelNames( CI :Bool; const Value, Value1 :Str) :Bool');
25025:  Func _NeedQuote( const Name :Str) :Bool');
25026:  Func _EasyNeedQuote( const Name :Str) :Bool');
25027:  Func _PosCh( aCh : Char; const s :Str) : integer');
25028:  Func _PosCh1( aCh : Char; const s :Str; StartPos : integer) : integer;
25029:  Func _PosCh2( aCh : AnsiChar; const s : Ansistr; StartPos : integer) : integer;
25030:  Func _PosCI( const Substr, Str :Str) : integer');
25031:  Func _PosExt( const Substr, Str :Str; BegSub, EndSub : TCharSet) : integer');
25032:  Func _PosExtCI( const Substr, Str :Str; BegSub, EndSub : TCharSet; AnsiUpper :Bool) : integer');
25033:  Func _PosInSubstr( const substr, Str :Str; BeginPos, EndPos : integer) : integer');
25034:  Func _PosInRight( const substr, Str :Str; BeginPos : integer) : integer');
25035:  Func _PosInSubstrCI( const SearchStr, Str :Str; BeginPos, EndPos : integer) : integer');
25036:  Func _PosInSubstrExt(const SearchStr:str;Str:str;BeginPos,EndPos:integer; BegSub, EndSub:TCharSet):int');
25037:  Func _PosInSubstrCIExt(const SearchStr,Str:Str;BeginPos, EndPos : integer; BegSub,EndSub :TCharSet):int');
25038:  Func _FirstPos( Search : array of Char; const InString :Str) : integer;
25039:  Func _FirstPos1( Search : array of AnsiChar; const InString : Ansistr) : integer;
25040:  Func _LastChar( const Str :Str) : Char');
25041:  Func _QuotedStr( const S :Str) :Str');
25042:  Func _CutQuote( const S :Str) :Str');
25043:  Func _StringInArray( const s :Str; const a : array of String) :Bool');
25044:  Func _IsBlank( const Str :Str) :Bool');
25045:  Func _IsBeginPartStr11( const PartStr, TargetStr : Ansistr) :Bool;
25046:  Func _IsBeginPartStrWA( const PartStr :Str; const TargetStr : Ansistr) :Bool');
25047:  Func _IsBeginPartStrVarA( const PartStr : Variant; const TargetStr : Ansistr) :Bool');
25048:  Func _IsBeginPartStr12( const PartStr, TargetStr : Widestring) :Bool;
25049:  Proc _DoLowerCase13( var Str :Str);
25050:  Proc _DoLowerCase14( var Str : Ansistr);
25051:  Proc _DoTrim( var Str :Str);
25052:  Func _VarTrimRight( const str : Variant) : Variant');
25053:  Proc _DoTrimRight15( var Str :Str);
25054:  Proc _DoTrimRight16( var Str : Widestring);
25055:  Proc _DoTrimRight17( var Str : Ansistr);
25056:  Proc _DoCopy18( const Source :Str; var Dest :Str; Index, Count : Integer);
25057:  Proc _DoCopy19( const Source : Ansistr; var Dest : Ansistr; Index, Count : Integer);
25058:  Func _FastTrim( const S :Str) :Str');
25059:  Func _FastCopy20( const S :Str; Index : integer; Count : Integer) :Str;
25060:  Func _FastCopy21( const S : Ansistr; Index : integer; Count : Integer) : Ansistr;
25061:  Proc _SLDifference( ASL, BSL : TStringList; ResultSL : TStrings);
25062:  Func _EmptyStrings( SL : TStrings) :Bool');
25063:  Proc _DeleteEmptyStr( Src : TStrings);
25064:  Func _NonAnsiSortCompareStrings( SL : TStringList; Index1, Index2 : Integer) : Integer');
25065:  Func _FindInDiapazon(SL:TStrList;const S:str;const StartIndex,EndIndex:int;AnsiCompare:bool;var
       Index:Int):bool);
25066:  Func _NonAnsiIndexOf( SL : TStringList; const S :Str) : integer');
25067:  Proc _GetNameAndValue( const s :Str; var Name, Value : Ansistr);
25068:  Func _StrToDateFmt( const ADate, Fmt :Str) : TDateTime');
25069:  Func _DateToSQLStr( const ADate : TDateTime) :Str');
25070:  Func _ValueFromStr( const Str :Str) :Str');
25071:  Func _WideUpperCase( const S : WideString) : WideString');
25072:  Func _Q_StrLen( P : AnsiChar) :Card');
25073:  Func _IsOldParamName( const ParamName :Str) :Bool');
25074:  Func _IsNewParamName( const ParamName :Str) :Bool');
25075:  Func _IsMasParamName( const ParamName :Str) :Bool');
25076:  Func _GUIDAsString( const AGUID : TGUID) : Ansistr');
25077:  Proc _GUIDAsStringToPChar( const AGUID : TGUID; Dest : AnsiChar);
25078:  Func _StringAsGUID( const AStr : Ansistr) : TGUID');
25079:  Func _CompareStrAndGuid( GUID : TGUID; const Str : Ansistr) : integer');
25080:  Func _CompareStrAndGuidP( GUID : TGUID; const Str : Ansistr) : integer');
25081:  Func _IsEqualGUIDs( const guid1, guid2 : TGUID) :Bool');
25082:  Func _IsNumericStr( const Str :Str) :Bool');
25083:  Func _IsEmptyStr( const Str :Str) :Bool');
25084:  Func _CompareStrTrimmed( const Str1, Str2 : PChar; Len : integer) : integer');
25085:  Func _CharInSet( C : AnsiChar; const CharSet : TSysCharSet) :Bool');
25086:  Func DecodeBase58(const Input:Str): TByteArray;
25087:  Func StreamToByteArray(Stream: TStream): TBytes;
25088:  Func InterpolateRGB(AColor1, AColor2: Integer; ACoeff: Double): Integer;
25089:  Proc SetPropDefaults(AObject: TPersistent; APropNames: array of String);
25090:  //Hashed := HashSHA256(HashSHA256(Copy(Decoded, 0, 21)));
25091:  Proc ValidateBitcoinAddress(const Address:Str; DHashSHA256: TByteArray);
25092: end;
25093:
25094: //type TPyObject = PPyObject;  this is all a part of prerelease to get a decent bufferpointer layout!
25095: //type PPyObject = TPyObject;
25096: The engine has a property, PythonEngine.InitScript:
```

```
25097: PythonEngine.InitScript.Add('import sys');
25098: PythonEngine.InitScript.Add('sys.path.insert(0, 'path/to/script');
25099:
25100: (*----------------------------------------------------------------------------*)
25101: Proc SIRegister_PythonEngine(CL: TPSPascalCompiler);
25102: begin
25103:   CL.AddTypeS('TPythonVersionProp', 'record DllName:Str; RegVersion:Str; APIVersion: Integer; end');
25104:  'PYT_METHOD_BUFFER_INCREASE','LongInt').SetInt( 10);
25105:  'PYT_MEMBER_BUFFER_INCREASE','LongInt').SetInt( 10);
25106:  'PYT_GETSET_BUFFER_INCREASE','LongInt').SetInt( 10);
25107:  'METH_VARARGS','LongWord').SetUInt( $0001);
25108:  'METH_KEYWORDS','LongWord').SetUInt( $0002);
25109:  'CO_OPTIMIZED','LongWord').SetUInt( $0001);
25110:  'CO_NEWLOCALS','LongWord').SetUInt( $0002);
25111:  'CO_VARARGS','LongWord').SetUInt( $0004);
25112:  'CO_VARKEYWORDS','LongWord').SetUInt( $0008);
25113:  Py_LT','LongInt').SetInt( 0);
25114:  Py_LE','LongInt').SetInt( 1);
25115:  Py_EQ','LongInt').SetInt( 2);
25116:  Py_NE','LongInt').SetInt( 3);
25117:  Py_GT','LongInt').SetInt( 4);
25118:  Py_GE','LongInt').SetInt( 5);
25119:   CL.AddTypeS('TRichComparisonOpcode', '( pyLT, pyLE, pyEQ, pyNE, pyGT, pyGE );
25120:   C_Long', 'Integer');  C_ULong', 'Cardinal');
25121:   C_Long', 'NativeInt'); C_ULong', 'NativeUInt');
25122:   TPFlag', '( tpfHeapType, tpfBaseType, tpfReady, tpfReadying, tpf'
25123:    +'HaveGC, tpVectorCall, tpMethodDescriptor, tpHaveVersionTag, tpValidVersion'
25124:    +'Tag, tpIsAbstract, tpLongSubclass, tpListSubClass, tpTupleSubclass, tpByte'
25125:    +'sSubclass, tpBaseExcSubclass, tpTypeSubclass );
25126:   CL.AddTypeS('TPFlags', 'set of TPFlag');
25127:   CL.AddConstantN('TPFLAGS_DEFAULT','LongInt').Value.ts32 := ord(tpfBaseType) or ord(tpHaveVersionTag);
25128:  'single_input','LongInt').SetInt( 256);
25129:  'file_input','LongInt').SetInt( 257);
25130:  'eval_input','LongInt').SetInt( 258);
25131:  'PyUnicode_WCHAR_KIND','LongInt').SetInt( 0);
25132:  'PyUnicode_1BYTE_KIND','LongInt').SetInt( 1);
25133:  'PyUnicode_2BYTE_KIND','LongInt').SetInt( 2);
25134:  'PyUnicode_4BYTE_KIND','LongInt').SetInt( 4);
25135:   T_SHORT','LongInt').SetInt( 0);
25136:  'T_INT','LongInt').SetInt( 1);
25137:  'T_LONG','LongInt').SetInt( 2);
25138:  'T_FLOAT','LongInt').SetInt( 3);
25139:  'T_DOUBLE','LongInt').SetInt( 4);
25140:  'T_STRING','LongInt').SetInt( 5);
25141:  'T_OBJECT','LongInt').SetInt( 6);
25142:  'T_CHAR','LongInt').SetInt( 7);
25143:  'T_BYTE','LongInt').SetInt( 8);
25144:  'T_UBYTE','LongInt').SetInt( 9);
25145:  'T_USHORT','LongInt').SetInt( 10);
25146:  'T_UINT','LongInt').SetInt( 11);
25147:  'T_ULONG','LongInt').SetInt( 12);
25148:  'T_STRING_INPLACE','LongInt').SetInt( 13);
25149:  'T_OBJECT_EX','LongInt').SetInt( 16);
25150:  'PY_READONLY','LongInt').SetInt( 1);
25151:  //CL.AddConstantN('RO','').SetString( READONLY);
25152:  CL.AddConstantN('READ_RESTRICTED','LongInt').SetInt( 2);
25153:  CL.AddConstantN('PY_WRITE_RESTRICTED','LongInt').SetInt( 4);
25154:   CL.AddTypeS('TPyMemberType', '( mtShort, mtInt, mtLong, mtFloat, mtDouble, mt'
25155:    +'String, mtObject, mtChar, mtByte, mtUByte, mtUShort, mtUInt, mtULong, mtStringInplace, mtObjectEx );
25156:   CL.AddTypeS('TPyMemberFlag', '( mfDefault, mfReadOnly, mfReadRestricted, mfWriteRestricted, mfRestricted
      );
25157:    CL.AddTypeS('TBasicService', '( pbsGetAttr, pbsSetAttr, pbsRepr, pbsCompa'
25158:    +'re, pbsHash, pbsStr, pbsGetAttrO, pbsSetAttrO, pbsCall, pbsTraverse, pbsClear, pbs'
25159:    +'RichCompare, pbsIter, pbsIterNext );
25160:   CL.AddTypeS('TBasicServices', 'set of TBasicService);
25161:   CL.AddTypeS('TNumberService', '( pnsAdd, pnsSubtract, pnsMultiply, pnsRem'
25162:    +'ainder, pnsDivmod, pnsPower, pnsNegative, pnsPositive, pnsAbsolute, pnsInvert, pn'
25163:    +'sLShift, pnsRShift, pnsAnd, pnsXor, pnsOr, pnsInt, pnsFloat, pnsFloorDivide, pnsTr'
25164:    +'ueDivide, pnsMatrixMultiply, pnsBool );
25165:    CL.AddTypeS('TNumberServices', 'set of TNumberService');
25166:   CL.AddTypeS('TInplaceNumberService', '( pnsInplaceAdd, pnsInplaceSubtra'
25167:    +'ct, pnsInplaceMultiply, pnsInplaceRemainder, pnsInplacePower, pnsInplaceLShift'
25168:    +', pnsInplaceRShift, pnsInplaceAnd, pnsInplaceXor, pnsInplaceOr, pnsInplaceFloor'
25169:    +'Divide, pnsInplaceTrueDivide, pnsInplaceMatrixMultiply );
25170:    CL.AddTypeS('TInplaceNumberServices', 'set of TInplaceNumberService');
25171:  CL.AddConstantN('PY_CR','Char').SetString( #13);
25172:  'PY_LF','Char').SetString( #10);
25173:  'PY_TAB','Char').SetString( #09);
25174:   //TypeS('PP_frozen', '^P_frozen // will not work');
25175:   //TypeS('P_frozen', '^_frozen // will not work');
25176:   //TypeS('PPyObject', '^PyObject // will not work');
25177:   //TypeS('PPPyObject', '^PPyObject // will not work');
25178:   //TypeS('PPPPyObject', '^PPPyObject // will not work');
25179:   //TypeS('PPyTypeObject', '^PyTypeObject // will not work');
25180:   //TypeS('PPySliceObject', '^PySliceObject // will not work');
25181:   //TypeS('binaryfunc', 'integer// will not work');
25182:   CL.AddTypeS('Py_complex', 'record real : double; imag : double; end');
25183:   CL.AddTypeS('PyMethodDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25184:    +'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_method : PPyMethodDef; end');
```

```
25185:   CL.AddTypeS('PPyMemberDescrObject', '^PyMemberDescrObject // will not work');
25186:   CL.AddTypeS('PyMemberDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25187:    +'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_member : PPyMemberDef; end');
25188:   //CL.AddTypeS('PPyGetSetDescrObject', '^PyGetSetDescrObject // will not work');
25189:   CL.AddTypeS('PyGetSetDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25190:    +'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_getset : PPyGetSetDef; end');
25191:   //CL.AddTypeS('PPyWrapperDescrObject', '^PyWrapperDescrObject // will not work');
25192:   CL.AddTypeS('PyWrapperDescrObject', 'record ob_refcnt : NativeInt; ob_type : '
25193:    +'PPyTypeObject; d_type: PPyTypeObject;d_name: PPyObject; d_base: pwrapperbase; d_wrapped:Pointer;
      end');
25194:   //CL.AddTypeS('PPyModuleDef_Base', '^PyModuleDef_Base // will not work');
25195:   CL.AddTypeS('PyModuleDef_Base', 'record ob_refcnt : NativeInt; ob_type : PPyT'
25196:    +'ypeObject; m_index : NativeInt; m_copy : PPyObject; end');
25197:   //CL.AddTypeS('PPyModuleDef', '^PyModuleDef // will not work');
25198:   CL.AddTypeS('PyModuleDef', 'record m_base : PyModuleDef_Base; m_name : PAnsiC'
25199:    +'har; m_doc : PAnsiChar; m_size : NativeInt; m_methods : PPyMethodDef; m_re'
25200:    +'load : inquiry; m_traverse : traverseproc; m_clear : inquiry; m_free : inquiry; end');
25201:   CL.AddTypeS('PyTypeObject', 'record ob_refcnt : NativeInt; ob_type : PPyTypeO'
25202:    +'bject; ob_size : NativeInt; tp_name : PAnsiChar; tp_basicsize : NativeInt;'
25203:    +' tp_itemsize : NativeInt; tp_dealloc : pydestructor; tp_vectorcall_offset '
25204:    +': NativeInt; tp_getattr : getattrfunc; tp_setattr : setattrfunc; tp_as_asy'
25205:    +'nc : Pointer; tp_repr : reprfunc; tp_as_number : PPyNumberMethods; tp_as_s'
25206:    +'equence : PPySequenceMethods; tp_as_mapping : PPyMappingMethods; tp_hash :'
25207:    +' hashfunc; tp_call : ternaryfunc; tp_str : reprfunc; tp_getattro : getattr'
25208:    +'ofunc; tp_setattro : setattrofunc; tp_as_buffer : Pointer; tp_flags : C_UL'
25209:    +'ong; tp_doc : PAnsiChar; tp_traverse : traverseproc; tp_clear : inquiry; t'
25210:    +'p_richcompare : richcmpfunc; tp_weaklistoffset : NativeInt; tp_iter : geti'
25211:    +'terfunc; tp_iternext : iternextfunc; tp_methods : PPyMethodDef; tp_members'
25212:    +' : PPyMemberDef; tp_getset : PPyGetSetDef; tp_base : PPyTypeObject; tp_dic'
25213:    +'t : PPyObject; tp_descr_get : descrgetfunc; tp_descr_set : descrsetfunc; t'
25214:    +'p_dictoffset : NativeInt; tp_init : initproc; tp_alloc : allocfunc; tp_new'
25215:    +' : newfunc; tp_free : pydestructor; tp_is_gc : inquiry; tp_bases : PPyObje'
25216:    +'ct; tp_mro : PPyObject; tp_cache : PPyObject; tp_subclasses : PPyObject; t'
25217:    +'p_weaklist : PPyObject; tp_del : PyDestructor; tp_version_tag :Card; '
25218:    +'tp_finalize : PyDestructor; tp_vectorcall : Pointer; tp_xxx1 : NativeInt; '
25219:    +'tp_xxx2 : NativeInt; tp_xxx3 : NativeInt; tp_xxx4 : NativeInt; tp_xxx5 : N'
25220:    +'ativeInt; tp_xxx6 : NativeInt; tp_xxx7 : NativeInt; tp_xxx8 : NativeInt; t'
25221:    +'p_xxx9 : NativeInt; tp_xxx10 : NativeInt; end');
25222:   CL.AddTypeS('PPyInterpreterState', '__Pointer');
25223:   CL.AddTypeS('PPyThreadState', '__Pointer');
25224:   //CL.AddTypeS('PNode', '^node // will not work');
25225:   CL.AddTypeS('node', 'record n_type : smallint; n_str : PAnsiChar; n_lineno : '
25226:    +'integer; n_col_offset : integer; n_nchildren : integer; n_child : PNode; end');
25227:   //CL.AddTypeS('PPyCompilerFlags', '^PyCompilerFlags // will not work');
25228:   CL.AddTypeS('PyCompilerFlags', 'record flags : integer; cf_feature_version:integer; end');
25229:  CL.AddConstantN('_PyDateTime_DATE_DATASIZE','LongInt').SetInt( 4);
25230:  ('_PyDateTime_TIME_DATASIZE','LongInt').SetInt( 6);
25231:  ('_PyDateTime_DATETIME_DATASIZE','LongInt').SetInt( 10);
25232:   CL.AddTypeS('PyGILState_STATE', '( PyGILState_LOCKED, PyGILState_UNLOCKED );
25233:   CL.AddClassN(CL.FindClass('TOBJECT'),'EDLLLoadError');
25234:   SIRegister_EDLLImportError(CL);
25235:   SIRegister_EPythonError(CL);
25236:    TOBJECT'),'EPyExecError');
25237:   'TOBJECT'),'EPyException');
25238:   'TOBJECT'),'EPyStandardError');
25239:   'TOBJECT'),'EPyArithmeticError');
25240:   'TOBJECT'),'EPyLookupError');
25241:   'TOBJECT'),'EPyAssertionError');
25242:   'TOBJECT'),'EPyAttributeError');
25243:   'TOBJECT'),'EPyEOFError');
25244:   'TOBJECT'),'EPyFloatingPointError');
25245:   'TOBJECT'),'EPyEnvironmentError');
25246:   'TOBJECT'),'EPyIOError');
25247:   'TOBJECT'),'EPyOSError');
25248:   'TOBJECT'),'EPyImportError');
25249:   'TOBJECT'),'EPyIndexError');
25250:   'TOBJECT'),'EPyKeyError');
25251:   'TOBJECT'),'EPyKeyboardInterrupt');
25252:   'TOBJECT'),'EPyMemoryError');
25253:   'TOBJECT'),'EPyNameError');
25254:   'TOBJECT'),'EPyOverflowError');
25255:   'TOBJECT'),'EPyRuntimeError');
25256:   'TOBJECT'),'EPyNotImplementedError');
25257:   PySyntaxError(CL);
25258:   (TOBJECT'),'EPyIndentationError');
25259:   (TOBJECT'),'EPyTabError');
25260:   (TOBJECT'),'EPySystemError');
25261:   (TOBJECT'),'EPySystemExit');
25262:   (TOBJECT'),'EPyTypeError');
25263:   (TOBJECT'),'EPyUnboundLocalError');
25264:   (TOBJECT'),'EPyValueError');
25265:   (TOBJECT'),'EPyUnicodeError');
25266:   (TOBJECT'),'UnicodeEncodeError');
25267:   (TOBJECT'),'UnicodeDecodeError');
25268:   (TOBJECT'),'UnicodeTranslateError');
25269:   (TOBJECT'),'EPyZeroDivisionError');
25270:   (TOBJECT'),'EPyStopIteration');
25271:   (TOBJECT'),'EPyWarning');
25272:   (TOBJECT'),'EPyUserWarning');
```

```
25273:   (TOBJECT'),'EPyDeprecationWarning');
25274:   (TOBJECT'),'PendingDeprecationWarning');
25275:   (TOBJECT'),'FutureWarning');
25276:   (TOBJECT'),'EPySyntaxWarning');
25277:   (TOBJECT'),'EPyRuntimeWarning');
25278:   (TOBJECT'),'EPyReferenceError');
25279:   (TOBJECT'),'EPyWindowsError');
25280:   CL.AddConstantN('kMaxLines','LongInt').SetInt( 1000);
25281:   CL.AddConstantN('kMaxLineLength','LongInt').SetInt( 256);
25282:   CL.AddTypeS('TSendDataEvent', 'Proc ( Sender : TObject; const Data : Ansistr);
25283:   TReceiveDataEvent', 'Proc ( Sender : TObject; var Data : Ansistr);
25284:   TSendUniDataEvent', 'Proc ( Sender : TObject; const Data : UnicodeString);
25285:   TReceiveUniDataEvent', 'Proc ( Sender : TObject; var Data : UnicodeString);
25286:   IOChar', 'WideChar');
25287:   IOString', 'UnicodeString');
25288:   TIOStringList', 'TStringList');
25289:   SIRegister_TPythonInputOutput(CL);
25290:   SIRegister_TDynamicDll(CL);
25291:   SIRegister_TPythonInterface(CL);
25292:   CL.AddTypeS('TDatetimeConversionMode', '( dcmToTuple, dcmToDatetime );
25293:  //CL.AddConstantN('DEFAULT_DATETIME_CONVERSION_MODE','').SetString( dcmToTuple);
25294:   CL.AddClassN(CL.FindClass('TOBJECT'),'TEngineClient');
25295:   CL.AddTypeS('TPathInitializationEvent', 'Proc ( Sender : TObject; var Path :Str);
25296:   //CL.AddTypeS('TSysPathInitEvent', 'Proc ( Sender : TObject; PathList : PPyObject);
25297:   CL.AddTypeS('TPythonFlag','(pfDebug,pfInteractive,pfNoSite,pfOptimize,pfVerbose,pfFrozenFlag
25298:                              ,pfIgnoreEnvironmentFlag);
25299:   CL.AddTypeS('TPythonFlags', 'set of TPythonFlag');
25300:   SIRegister_TTracebackItem(CL);
25301:   SIRegister_TPythonTraceback(CL);
25302:   SIRegister_TPythonEngine(CL);
25303:   SIRegister_TEngineClient(CL);
25304:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMethodsContainer');
25305:   CL.AddClassN(CL.FindClass('TOBJECT'),'TEventDefs');
25306:   SIRegister_TEventDef(CL);
25307:   SIRegister_TEventDefs(CL);
25308:   SIRegister_TMethodsContainer(CL);
25309:   SIRegister_TMembersContainer(CL);
25310:   SIRegister_TGetSetContainer(CL);
25311:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPythonModule');
25312:   CL.AddClassN(CL.FindClass('TOBJECT'),'TErrors');
25313:   CL.AddTypeS('TErrorType', '( etString, etClass );
25314:   SIRegister_TParentClassError(CL);
25315:   SIRegister_TError(CL);
25316:   SIRegister_TErrors(CL);
25317:   SIRegister_TPythonModule(CL);
25318:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPythonType');
25319:   SIRegister_TPyObject(CL);
25320:   //CL.AddTypeS('TPyObjectClass', 'class of TPyObject');
25321:   CL.AddTypeS('TMappingService', '( msLength, msSubscript, msAssSubscript );
25322:   CL.AddTypeS('TMappingServices', 'set of TMappingService');
25323:   //CL.AddTypeS('TJclEmailReadOptions', 'set of TJclEmailReadOption');
25324:   SIRegister_TTypeServices(CL);
25325:   SIRegister_TPythonType(CL);
25326:   CL.AddTypeS('TGetDataEvent', 'Proc ( Sender : TObject; var Data : Variant);
25327:   CL.AddTypeS('TSetDataEvent', 'Proc ( Sender : TObject; Data : Variant);
25328:   //CL.AddTypeS('TExtGetDataEvent', 'Proc ( Sender : TObject; var Data : PPyObject);
25329:   //CL.AddTypeS('TExtSetDataEvent', 'Proc ( Sender : TObject; Data : PPyObject);
25330:   SIRegister_TPythonDelphiVar(CL);
25331:   SIRegister_TPyVar(CL);
25332:   CL.AddTypeS('TThreadExecMode', '( emNewState, emNewInterpreter );
25333:   SIRegister_TPythonThread(CL);
25334:  {('Func pyio_write( self, args : PPyObject) : PPyObject');
25335:  C'Func pyio_read( self, args : PPyObject) : PPyObject');
25336:  C'Func pyio_SetDelayWrites( self, args : PPyObject) : PPyObject');
25337:  C'Func pyio_SetMaxLines( self, args : PPyObject) : PPyObject');
25338:  C'Func pyio_GetTypesStats( self, args : PPyObject) : PPyObject');  }
25339:  Func GetPythonEngine : TPythonEngine');
25340:  Func PythonOK :Bool');
25341:  //Func PythonToDelphi( obj : PPyObject) : TPyObject');
25342:  //Func IsDelphiObject( obj : PPyObject) :Bool');
25343: // Proc PyObjectDestructor( pSelf : PPyObject);
25344:  //Proc FreeSubtypeInst( ob : PPyObject);
25345:  //Func PyType_HasFeature( AType : PPyTypeObject; AFlag : Integer) :Bool');
25346:  Func SysVersionFromDLLName( const DLLFileName :Str) :Str');
25347:  Proc PythonVersionFromDLLName( LibName :Str; out MajorVersion, MinorVersion : integer);
25348:  Func IsPythonVersionRegistered(PythonVersion:str; out InstallPath:tring;out AllUserInstall:Bool) Bool');
25349:  Proc MaskFPUExceptions( ExceptionsMasked :Bool; MatchPythonPrecision :Bool);
25350:  Func CleanString13( const s : Ansistr; AppendLF :Bool) : Ansistr;
25351:  Func CleanString14( const s : UnicodeString; AppendLF :Bool) : UnicodeString;
25352:  Proc setgPythonEngine(mypyengine: TPythonEngine);
25353:  Func getgPythonEngine: TPythonEngine;
25354: end;
25355:
25356: Proc SIRegister_VclPythonGUIInputOutput(CL: TPSPascalCompiler);
25357: begin
25358:  CL.AddConstantN('WM_WriteOutput','LongInt').SetInt( WM_USER + 1);
25359:   SIRegister_TPythonGUIInputOutput(CL);
25360:  CL.AddDelphiFunction('Proc Register');
25361: end;
```

```
25362:
25363: (*----------------------------------------------------------------------------*)
25364: Proc SIRegister_TDynamicDll(CL: TPSPascalCompiler);
25365: begin
25366:   //with RegClassS(CL,'TComponent', 'TDynamicDll') do
25367:   with CL.AddClassN(CL.FindClass('TComponent'),'TDynamicDll') do begin
25368:     Constructor Create( AOwner : TComponent);
25369:     Proc Free;
25370:     Proc OpenDll( const aDllName :Str);
25371:     Func IsHandleValid :Bool');
25372:     Proc LoadDll');
25373:     Proc UnloadDll');
25374:     Proc Quit');
25375:     RAutoLoad', 'Boolean', iptrw);
25376:     RAutoUnload', 'Boolean', iptrw);
25377:     RDllName', 'string', iptrw);
25378:     RDllPath', 'string', iptrw);
25379:     RAPIVersion', 'Integer', iptrw);
25380:     RRegVersion', 'string', iptrw);
25381:     RFatalAbort', 'Boolean', iptrw);
25382:     RFatalMsgDlg', 'Boolean', iptrw);
25383:     RUseLastKnownVersion', 'Boolean', iptrw);
25384:     ROnAfterLoad', 'TNotifyEvent', iptrw);
25385:     ROnBeforeLoad', 'TNotifyEvent', iptrw);
25386:     ROnBeforeUnload', 'TNotifyEvent', iptrw);
25387:   end;
25388: end;
25389:
25390: (*----------------------------------------------------------------------------*)
25391: Proc SIRegister_TPythonEngine(CL: TPSPascalCompiler);
25392: begin
25393:   //with RegClassS(CL,'TPythonInterface', 'TPythonEngine') do
25394:   with CL.AddClassN(CL.FindClass('TPythonInterface'),'TPythonEngine') do begin
25395:     Constructor Create( AOwner : TComponent);
25396:     Proc Free;
25397:     Proc SetPythonHome( const PythonHome : UnicodeString);
25398:     Proc SetProgramName( const ProgramName : UnicodeString);
25399:     Func IsType( ob : PPyObject; obt : PPyTypeObject) :Bool');
25400:     Func Run_CommandAsString(const command:Ansistr; mode:Integer):str');
25401:     Func Run_CommandAsObject(const command : Ansistr; mode : Integer) : PPyObject');
25402:     Func Run_CommandAsObjectWithDict(const command:Ansistr;mode:Integer;locals,globals:PPyObject):
PPyObject');
25403:     Func EncodeString1( const str : UnicodeString) : Ansistr;;
25404:     Func EncodeString2( const str : Ansistr) : Ansistr;
25405:     Func EncodeWindowsFilePath( const str :Str) : Ansistr');
25406:     Proc Initialize2;
25407:     Proc DoOpenDll2(const aDllName :Str);
25408:     Proc SetProgramArgs2;
25409:     Proc ExecString( const command : Ansistr);
25410:     Proc ExecStr( const command : Ansistr);       //alias
25411:     Proc ExecString3( const command : Ansistr);   //alias
25412:     Proc ExecStrings4( strings : TStrings);
25413:     Proc ExecStrings( strings : TStrings);     //alias
25414:     Func EvalString5( const command : Ansistr) : PPyObject;
25415:     Func EvalStringAsStr( const command : Ansistr) :Str');
25416:     Func EvalStr( const command : Ansistr) :Str');
25417:     Func EvalStrings6( strings : TStrings) : PPyObject;
25418:     Proc ExecString7(const command:Ansistr; locals,globals:PPyObject);
25419:     Proc ExecStrings8( strings : TStrings; locals, globals : PPyObject);
25420:     Func EvalString9(const command:Ansistr; locals, globals : PPyObject): PPyObject;
25421:     Func EvalStrings10( strings : TStrings; locals, globals : PPyObject): PPyObject;
25422:     Func EvalStringsAsStr( strings : TStrings) :Str');
25423:     Func EvalPyFunction( pyfunc, pyargs : PPyObject) : Variant');
25424:     Func EvalFunction(pyfunc:PPyObject;const args:array of const): Variant');
25425:     Func EvalFunctionNoArgs( pyfunc : PPyObject) : Variant');
25426:     Func CheckEvalSyntax( const str : Ansistr) :Bool');
25427:     Func CheckExecSyntax( const str : Ansistr) :Bool');
25428:     Func CheckSyntax( const str : Ansistr; mode : Integer) :Bool');
25429:     Proc RaiseError');
25430:     Func PyObjectAsString( obj : PPyObject) :Str');
25431:     Proc DoRedirectIO');
25432:     Proc AddClient( client : TEngineClient);
25433:     Proc RemoveClient( client : TEngineClient);
25434:     Func FindClient( const aName :Str) : TEngineClient');
25435:     Func TypeByName( const aTypeName : Ansistr) : PPyTypeObject');
25436:     Func ModuleByName( const aModuleName : Ansistr) : PPyObject');
25437:     Func MethodsByName( const aMethodsContainer :Str) : PPyMethodDef');
25438:     Func VariantAsPyObject( const V : Variant) : PPyObject');
25439:     Func PyObjectAsVariant( obj : PPyObject) : Variant');
25440:     Func VarRecAsPyObject( const v : TVarRec) : PPyObject');
25441:     Func MakePyTuple( const objects : array of PPyObject) : PPyObject');
25442:     Func MakePyList( const objects : array of PPyObject) : PPyObject');
25443:     Func ArrayToPyTuple( const items : array of const) : PPyObject');
25444:     Func ArrayToPyList( const items : array of const) : PPyObject');
25445:     Func ArrayToPyDict( const items : array of const) : PPyObject');
25446:     Func StringsToPyList( strings : TStrings) : PPyObject');
25447:     Func StringsToPyTuple( strings : TStrings) : PPyObject');
25448:     Proc PyListToStrings( list : PPyObject; strings : TStrings);
25449:     Proc PyTupleToStrings( tuple : PPyObject; strings : TStrings);
```

```
25450:     Func ReturnNone : PPyObject');
25451:     Func ReturnTrue : PPyObject');
25452:     Func ReturnFalse : PPyObject');
25453:     Func FindModule( const ModuleName : Ansistr) : PPyObject');
25454:     Func FindFunction( const ModuleName, FuncName : Ansistr) : PPyObject');
25455:     Func SetToList( data : Pointer; size : Integer) : PPyObject');
25456:     Proc ListToSet( List : PPyObject; data : Pointer; size : Integer);
25457:     Proc CheckError( ACatchStopEx :Bool);
25458:     Func GetMainModule : PPyObject');
25459:     Func PyTimeStruct_Check( obj : PPyObject) :Bool');
25460:     Func PyDate_Check( obj : PPyObject) :Bool');
25461:     Func PyDate_CheckExact( obj : PPyObject) :Bool');
25462:     Func PyDateTime_Check( obj : PPyObject) :Bool');
25463:     Func PyDateTime_CheckExact( obj : PPyObject) :Bool');
25464:     Func PyTime_Check( obj : PPyObject) :Bool');
25465:     Func PyTime_CheckExact( obj : PPyObject) :Bool');
25466:     Func PyDelta_Check( obj : PPyObject) :Bool');
25467:     Func PyDelta_CheckExact( obj : PPyObject) :Bool');
25468:     Func PyTZInfo_Check( obj : PPyObject) :Bool');
25469:     Func PyTZInfo_CheckExact( obj : PPyObject) :Bool');
25470:     Func PyUnicodeFromString11( const AString : UnicodeString) : PPyObject;
25471:     Func PyUnicodeFromString12( const AString : Ansistr) : PPyObject;
25472:     Func PyUnicodeAsString( obj : PPyObject) : UnicodeString');
25473:     Func PyUnicodeAsUTF8String( obj : PPyObject) : RawByteString');
25474:     Func PyBytesAsAnsistr( obj : PPyObject) : Ansistr');
25475:     ClientCount', 'Integer', iptr);
25476:     Clients', 'TEngineClient Integer', iptr);
25477:     ExecModule', 'Ansistr', iptrw);
25478:     ThreadState', 'PPyThreadState', iptr);
25479:     Traceback', 'TPythonTraceback', iptr);
25480:     LocalVars', 'PPyObject', iptrw);
25481:     GlobalVars', 'PPyObject', iptrw);
25482:     IOPythonModule', 'TObject', iptr);
25483:     PythonHome', 'UnicodeString', iptrw);
25484:     ProgramName', 'UnicodeString', iptrw);
25485:     AutoFinalize', 'Boolean', iptrw);
25486:     VenvPythonExe', 'string', iptrw);
25487:     DatetimeConversionMode', 'TDatetimeConversionMode', iptrw);
25488:     InitScript', 'TStrings', iptrw);
25489:     InitThreads', 'Boolean', iptrw);
25490:     IO', 'TPythonInputOutput', iptrw);
25491:     PyFlags', 'TPythonFlags', iptrw);
25492:     RedirectIO', 'Boolean', iptrw);
25493:     UseWindowsConsole', 'Boolean', iptrw);
25494:     OnAfterInit', 'TNotifyEvent', iptrw);
25495:     OnPathInitialization', 'TPathInitializationEvent', iptrw);
25496:     OnSysPathInit', 'TSysPathInitEvent', iptrw);
25497:   end;
25498: end;
25499:
25500: Proc SIRegister_TPythonModule(CL: TPSPascalCompiler);
25501: begin
25502:   //with RegClassS(CL,'TMethodsContainer', 'TPythonModule') do
25503:   with CL.AddClassN(CL.FindClass('TMethodsContainer'),'TPythonModule') do  begin
25504:     Constructor Create( AOwner : TComponent);
25505:     Proc MakeModule');
25506:     Proc DefineDocString');
25507:     Proc Initialize');
25508:     Proc InitializeForNewInterpreter');
25509:     Proc AddClient( client : TEngineClient);
25510:     Func ErrorByName( const AName : Ansistr) : TError');
25511:     Proc RaiseError( const error, msg : Ansistr);
25512:     Proc RaiseErrorFmt(const error,format:Ansistr;const Args:array of const);
25513:     Proc RaiseErrorObj(const error, msg : Ansistr; obj : PPyObject);
25514:     Proc BuildErrors');
25515:     Proc SetVar( const varName : Ansistr; value : PPyObject);
25516:     Func GetVar( const varName : Ansistr) : PPyObject');
25517:     Proc DeleteVar( const varName : Ansistr);
25518:     Proc ClearVars');
25519:     Proc SetVarFromVariant(const varName:Ansistr;const value:Variant);
25520:     Func GetVarAsVariant( const varName : Ansistr) : Variant');
25521:     Module', 'PPyObject', iptr);
25522:     Clients', 'TEngineClient Integer', iptr);
25523:     ClientCount', 'Integer', iptr);
25524:     DocString', 'TStringList', iptrw);
25525:     ModuleName', 'Ansistr', iptrw);
25526:     Errors', 'TErrors', iptrw);
25527:     OnAfterInitialization', 'TNotifyEvent', iptrw);
25528:   end;
25529: end;
25530:
25531: Example:
25532: Proc initDemodll;
25533: begin   //mode: eval_input, file_input
25534:   try
25535:     gEngine:= TPythonEngine.Create(nil);
25536:     gEngine.AutoFinalize:= False;
25537:     gEngine.LoadDll;
25538:     gModule:= TPythonModule.Create(nil);
```

```
25539:     gModule.Engine:= gEngine;
25540:     gModule.ModuleName:= 'pydemodll';
25541:     //gModule.AddMethod( 'add', @Add, 'add(a,b) -> a+b' );
25542:    //  The engine has a property, PythonEngine.InitScript:
25543:     gEngine.InitScript.Add('import sys');
25544:     gEngine.InitScript.Add('sys.path.insert(0, "path/to/script");
25545:     //gModule.Initialize;
25546:     writeln('PythonOK: in init '+botostr(PythonOK));
25547:     //gEngine.EvalStringAsStr(pycmd);
25548:   except
25549:      gEngine.raiseError;
25550:      writeln(ExceptionToString(ExceptionType, ExceptionParam));
25551:      gEngine.Free;
25552:      gModule.Free;
25553:   end;
25554: end;
25555:
25556: const LB = CR+LF;
25557: const PyModule =
25558:          'def printData(data): '+#13#10+
25559:          '  return data+data+"/n"';
25560:
25561: const PYCMD = 'print("this is box")'+LB+
25562:               'import sys'+LB+
25563:               'f=open(r"C:\maXbox\maxbox3\maxbox3\maXbox3\examples\pytest2.txt","w")'+LB+
25564:               'f.write("Hello PyWorld_mX4, \n")'+LB+
25565:               'f.write("This data will be written on the file.")'+LB+
25566:               'f.close()';
25567:
25568: Proc PYLaz_P4D_Demo;
25569: //https://wiki.freepascal.org/Python4Delphi
25570: var eng : TPythonEngine;
25571:     Out1: TPythonGUIInputOutput;
25572: begin
25573:   eng:= TPythonEngine.Create(Nil);
25574:   Out1:= TPythonGUIInputOutput.create(nil)
25575:   Out1.output:= pyMemo; //debugout.output;  //memo2;
25576:   Out1.RawOutput:= False;
25577:   Out1.UnicodeIO:= False;
25578:   Out1.maxlines:= 20;
25579:   out1.displaystring('this string')
25580:   //eng.IO:= Out1;
25581:   Out1.writeline('draw the line');
25582:   eng.LoadDll;
25583:   eng.IO:= Out1;
25584:     if eng.IsHandleValid then begin
25585:       writeln('DLLhandle: '+botostr(eng.IsHandleValid))
25586:       WriteLn('evens: '+ eng.EvalStringAsStr('[x*2 for x in range(10)]'));
25587:       WriteLn('gauss: '+ eng.EvalStr('sum([x for x in range(101)])'));   //alias
25588:       writeln('syncheck '+
25589:           botostr(eng.CheckEvalSyntax('print("powers:",[x**2 for x in range(10)])')));
25590:       eng.ExecString('print("powers:",[x**2 for x in range(10)]);
25591:       eng.ExecStr(PYCMD);
25592:       writeln('ExecSynCheck2 '+
25593:             botostr(eng.CheckExecSyntax(filetostring(PYSCRIPT))));
25594:       eng.ExecString(filetostring(PYSCRIPT));
25595:     end
25596:   else writeln('invalid library handle! '+Getlasterrortext);
25597:   writeln('PythonOK '+botostr(PythonOK));
25598:   out1.free;
25599:   //pyImport(PyModule);
25600:   eng.free;
25601: end;
25602:
25603: //python template
25604: with TPythonEngine.Create(Nil) do begin
25605:   pythonhome:= 'C:\Users\max\AppData\Local\Programs\Python\Python36-32\';
25606:   try
25607:     loadDLL;
25608:     Println('Decimal: '+
25609:           EvalStr('__import__("decimal").Decimal(0.1)'));
25610:   except
25611:     raiseError;
25612:   finally
25613:     free;
25614:   end;
25615:
25616:  // This is a new method in VarPyth, but I'll put it here just in case....
25617: function VarPyToStrings(const AValue : Variant; const AStrings: TStrings): Integer;
25618: begin
25619:   Assert(Assigned(AStrings));
25620:   if VarIsPythonList(AValue) then
25621:     GetPythonEngine.PyListToStrings(
25622:       ExtractPythonObjectFrom(AValue), AStrings)
25623:   else
25624:     raise Exception.Create('Python List expected: ' + _type(AValue));
25625:   Result := AStrings.Count;
25626: end;
25627:
```

```
25628: procedure TForm16.Button1Click(Sender: TObject);
25629: begin
25630:   with NLTK1 do begin
25631:     nltk.download('punkt');
25632:     nltk.download('averaged_perceptron_tagger');
25633:     nltk.download('maxent_ne_chunker');
25634:     nltk.download('words');
25635:     nltk.download('treebank');
25636:
25637:     var tokens := nltk.word_tokenize(memo1.lines.Text);
25638:     VarPyToStrings(tokens, lbTokens.Items);
25639:
25640: Proc SIRegister_VarPyth(CL: TPSPascalCompiler);
25641: begin
25642:  CL.AddTypeS('TSequenceType', '( pystTuple, pystList );
25643:  Func VarPythonCreate0( AObject : TPyObject) : Variant;
25644:  Func VarPythonCreate1( const AValue : Variant) : Variant;
25645:  Func VarPythonCreate2( const AValues: array of const; ASequenceType : TSequenceType) : Variant;
25646:  Func VarPythonEval( const APythonExpression : Ansistr) : Variant');
25647:  Func VarPython : TVarType');
25648:  Func VarIsPython( const AValue : Variant) :Bool');
25649:  Func VarAsPython( const AValue : Variant) : Variant');
25650:  Func ExtractPythonObjectFrom( const AValue : Variant) : TPyObject');
25651:  Func VarIsSame( const A, B : Variant) :Bool');
25652:  Func VarIsSameType( const A, B : Variant) :Bool');
25653:  Func VarIsPythonSequence( const AValue : Variant) :Bool');
25654:  Func VarIsPythonMapping( const AValue : Variant) :Bool');
25655:  Func VarIsPythonNumber( const AValue : Variant) :Bool');
25656:  Func VarIsPythonString( const AValue : Variant) :Bool');
25657:  Func VarIsPythonInteger( const AValue : Variant) :Bool');
25658:  Func VarIsPythonFloat( const AValue : Variant) :Bool');
25659:  Func VarIsPythonTuple( const AValue : Variant) :Bool');
25660:  Func VarIsPythonList( const AValue : Variant) :Bool');
25661:  Func VarIsPythonDict( const AValue : Variant) :Bool');
25662:  Func VarIsPythonClass( const AValue : Variant) :Bool');
25663:  Func VarIsPythonMethod( const AValue : Variant) :Bool');
25664:  Func VarIsPythonFunction( const AValue : Variant) :Bool');
25665:  Func VarIsPythonModule( const AValue : Variant) :Bool');
25666:  Func VarIsPythonCallable( const AValue : Variant) :Bool');
25667:  Func VarIsPythonIterator( const AValue : Variant) :Bool');
25668:  Func VarIsPythonUnicode( const AValue : Variant) :Bool');
25669:  Func VarIsPythonDateTime( const AValue : Variant) :Bool');
25670:  Func VarIsPythonDate( const AValue : Variant) :Bool');
25671:  Func VarIsPythonTime( const AValue : Variant) :Bool');
25672:  Func VarIsPythonDateTimeDelta( const AValue : Variant) :Bool');
25673:  Func VarIsPythonTZInfo( const AValue : Variant) :Bool');
25674:  Func VarIsBool( const AValue : Variant) :Bool');
25675:  Func VarIsEnum( const AValue : Variant) :Bool');
25676:  Func VarIsInstanceOf( const AInstance, AClass : Variant) :Bool');
25677:  Func VarIsSubclassOf( const ADerived, AClass : Variant) :Bool');
25678:  Func VarIsSubtypeOf( const ADerived, AType : Variant) :Bool');
25679:  Func VarIsNone( const AValue : Variant) :Bool');
25680:  Func VarIsTrue( const AValue : Variant) :Bool');
25681:  Func VarModuleHasObject( const AModule : Variant; aObj: Ansistr):Bool');
25682:  Func NewPythonList( const ASize : Integer) : Variant');
25683:  Func NewPythonTuple( const ASize : Integer) : Variant');
25684:  Func NewPythonDict : Variant');
25685:  Func VarPythonAsString( AValue : Variant) :Str');
25686:  Func VarPythonToVariant( AValue : Variant) : Variant');
25687:  Func PyNone : Variant');
25688:  Func pyEllipsis : Variant');
25689:  Func pyMainModule : Variant');
25690:  Func pyBuiltinModule : Variant');
25691:  Func pySysModule : Variant');
25692:  Func ypDatetimeModule : Variant');
25693:  Func PyImport( const AModule : Ansistr) : Variant');
25694:  Func pylen( const AValue : Variant) : NativeInt');
25695:  Func py_type( const AValue : Variant) : Variant');
25696:  Func pyiter( const AValue : Variant) : Variant');
25697:   CL.AddTypeS(TVarPyEnumerator', 'record FIterator : Variant; FCurrent : Variant; end');
25698:   CL.AddTypeS(TVarPyEnumerateHelper', 'record FIterable : Variant; end');
25699:  Func VarPyIterate( const AValue : Variant) : TVarPyEnumerateHelper');
25700: end;
25701:
25702: Proc SIRegister_WDCCMisc(CL: TPSPascalCompiler);  //WMI Delphi Code Creator
25703: begin   //https://github.com/RRUZ/wmi-delphi-code-creator/tree/master/Units/Misc
25704:  TProcLog', 'Proc ( const Log :Str);
25705:  Proc wdc_CaptureConsoleOutput( const lpCommandLine :Str; OutPutList : TStrings);
25706:  Proc wdc_MsgWarning( const Msg :Str);
25707:  Proc wdc_MsgInformation( const Msg :Str);
25708:  Func wdc_MsgQuestion( const Msg :Str) :Bool');
25709:  Func wdc_GetFileVersion( const FileName :Str) :Str');
25710:  Func wdc_GetFileDescription( const FileName :Str) :Str');
25711:  Func wdc_GetTempDirectory :Str');
25712:  Func wdc_GetWindowsDirectory :Str');
25713:  Func wdc_GetSpecialFolder( const CSIDL : integer) :Str');
25714:  Func wdc_IsWow64 :Bool');
25715:  Func wdc_CopyDir( const fromDir, toDir :Str) :Bool');
25716:  Proc wdc_SetGridColumnWidths( DbGrid : TDBGrid);
```

```
25717:  Func wdc_Ping( const Address :Str; Retries, BufferSize : Word; Log : TStrings) :Bool');
25718:  Proc wdc_ScaleImage32( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double);
25719:  Proc wdc_ExtractIconFile( Icon : TIcon; const FileName :Str; IconType :Card);
25720:  Proc wdc_ExtractBitmapFile( Bmp : TBitmap; const FileName :Str; IconType :Card);
25721:  Proc wdc_ExtractBitmapFile32( Bmp : TBitmap; const FileName :Str; IconType :Card);
25722:  Proc wdc_CheckForUpdates( Silent :Bool);
25723: end;
25724:
25725: (*----------------------------------------------------------------------------*)
25726: Proc SIRegister_WDCCOleVariantEnum(CL: TPSPascalCompiler);
25727: begin
25728:   SIRegister_IOleVariantEnum(CL);
25729:   SIRegister_IGetOleVariantEnum(CL);
25730:   SIRegister_TOleVariantEnum(CL);
25731:   SIRegister_TOleVariantArrayEnum(CL);
25732:  Func GetOleVariantEnum( Collection : OLEVariant) : IGetOleVariantEnum');
25733:  Func GetOleVariantArrEnum( Collection : OLEVariant) : IGetOleVariantEnum');
25734: end;
25735:
25736: (*----------------------------------------------------------------------------*)
25737: Proc SIRegister_WDCCWinInet(CL: TPSPascalCompiler);
25738: begin
25739:  CL.AddConstantN('WM_UWININET_THREAD_FINISHED','LongInt').SetInt( WM_USER + 669);
25740:  CL.AddConstantN('WM_UWININET_THREAD_CANCELLED','LongInt').SetInt( WM_USER + 670);
25741:  CL.AddTypeS('TuWinInetProcCallBack', 'Proc ( BytesRead : Integer);
25742:   SIRegister_TWinINetGetThread(CL);
25743:  Func wdc_GetRemoteFileSize( const Url :Str) : Integer');
25744:  Proc wdc_WinInet_HttpGet( const Url :Str; Stream : TStream; CallBack : TuWinInetProcCallBack);
25745:  Func wdc_WinInet_HttpGet2( const Url :Str; CallBack : TuWinInetProcCallBack) :Str;
25746:  Func wdc_GetWinInetError( ErrorCode :Card) :Str');
25747: end;
25748:
25749: Proc SIRegister_PythonVersions(CL: TPSPascalCompiler);
25750: begin
25751:   SIRegister_TPythonVersion(CL);
25752:   CL.AddTypeS('TPythonVersions', 'array of TPythonVersion');
25753:  Func CompareVersions( A, B :Str) : Integer');
25754:  Func IsEXEx64( const EXEName :Str) :Bool');
25755:  Func Isx64( const FileName :Str) :Bool');
25756:  Func GetRegisteredPythonVersion( SysVersion :Str; out PythonVersion : TPythonVersion) :Bool');
25757:  Func GetRegisteredPythonVersions : TPythonVersions');
25758:  Func GetLatestRegisteredPythonVersion( out PythonVersion : TPythonVersion) :Bool');
25759:  Func PythonVersionFromPath(const Path:str;out PythonVersion:TPythonVersion;AcceptVirtualEnvs:Bool):Bool);
25760: end;
25761:
25762: Example: var pyv: TPythonVersion;  pyvs:TPythonVersions;
25763:    pyvs:= GetRegisteredPythonVersions;
25764:    writeln(objtostr(pyv));
25765:    for it:= 1 to length(pyvs)-1 do begin
25766:     writeln(pyvs[it].dllname)
25767:     writeln(pyvs[it].installpath)
25768:     writeln(pyvs[it].PythonExecutable )
25769:     writeln(pyvs[it].Version )
25770:    end;
25771:
25772: Proc SIRegister_TPythonAction(CL: TPSPascalCompiler);
25773: begin
25774:   //with RegClassS(CL,'TAction', 'TPythonAction') do
25775:   with CL.AddClassN(CL.FindClass('TAction'),'TPythonAction') do begin
25776:     'fRegisteredMethods', 'TList', iptrw);
25777:     'fPythonModule', 'TPythonModule', iptrw);
25778:     'fClearname', 'string', iptrw);
25779:     'fRegistername', 'string', iptrw);
25780:     'fUnregistername', 'string', iptrw);
25781:     Proc ClearMethods');
25782:     Constructor Create( AOwner : TComponent);
25783:     Func HandlesTarget( Target : TObject) :Bool');
25784:     Func Execute :Bool');
25785:     Proc UpdateTarget( Target : TObject);
25786:     Proc InitializeAction');
25787:     RegisterProperty('RegisteredMethods', 'TList', iptr);
25788:     RegisterProperty('PythonModule', 'TPythonModule', iptrw);
25789:   end;
25790: end;
25791:
25792: //V4.7.5.90
25793: procedure SIRegister_TSingleListClass(CL: TPSPascalCompiler);
25794: begin
25795:   TSingleListSortCompare
25796:    = function (Item1, Item2: Single): Integer;
25797:   TSingleDescriptor
25798:    = function (Index:Integer;Item : Single) : string;
25799:
25800:   //with RegClassS(CL,'TObject', 'TSingleListClass') do
25801:   with CL.AddClassN(CL.FindClass('TObject'),'TSingleListClass') do begin
25802:     Constructor Create');
25803:     Procedure Free;
25804:     Function Add( Item : Single) : Integer');
25805:     Procedure Clear');
```

```
25806:     Procedure SaveToStream( const S : TStream);
25807:     Procedure LoadFromStream( const S : TStream; const KeepCurrentSortType : Boolean);
25808:     Procedure SaveToFile( FileName : string);
25809:     Procedure LoadFromFile( FileName : string; const KeepCurrentSortType : Boolean);
25810:     Procedure Delete( Index : Integer);
25811:     Function ErrMsg( const Msg : string; Data : Integer) : string');
25812:     Procedure Exchange( Index1, Index2 : Integer);
25813:     Function Expand : TSingleListClass');
25814:     Function First : Single');
25815:     Function IndexOf( Value : Single) : Integer');
25816:     Procedure Insert( Index : Integer; Item : Single);
25817:     Function Last : Single');
25818:     Procedure Move( CurIndex, NewIndex : Integer);
25819:     Function Remove( Item : Single) : Integer');
25820:     Procedure Pack( NilValue : Single);
25821:     Procedure Sort( Compare : TSingleListSortCompare);
25822:     Procedure SortUp');
25823:     Procedure SortDown');
25824:     Procedure ShowList( StringList: TStrings; Descriptor: TSingleDescriptor; ClearIt:Boolean);
25825:     Function Minimum : Single');
25826:     Function Maximum : Single');
25827:     Function Range : Single');
25828:     Function Sum : Extended');
25829:     Function SumSqr : Extended');
25830:     Function Average : Extended');
25831:     Procedure CopyFrom( List : TSingleListClass; const KeepCurrentSortType : Boolean);
25832:     Procedure CopyTo( List : TSingleListClass; const KeepDestSortType : Boolean);
25833:     Procedure Push( Value : Single);
25834:     Function LifoPop( DefValue : Single) : Single');
25835:     Function FifoPop( DefValue : Single) : Single');
25836:     RegisterProperty('List', 'PSinglePtrList', iptr);
25837:     RegisterProperty('Capacity', 'Integer', iptrw);
25838:     RegisterProperty('Count', 'Integer', iptrw);
25839:     RegisterProperty('Items', 'Single Integer', iptrw);
25840:     SetDefaultPropery('Items');
25841:     RegisterProperty('SortType', 'TSingleSortOption', iptrw);
25842:   end;
25843: end;
25844:
25845: procedure SIRegister_DoubleList4(CL: TPSPascalCompiler);
25846: begin
25847:  CL.AddConstantN('SDoubleListVoidError','String').SetString( 'Invalid method call (empty list)!');
25848:  CL.AddConstantN('SDoubleListSortError','String').SetString( 'Invalid method call (sorted list)!');
25849:    //= array[0..MaxDoubleListSize - 1] of Double;
25850:   //  CL.AddTypeS('TDoublePtrList', 'array[0..2147417850] of Single');
25851:  CL.AddTypeS('TDoublePtrList', 'array of Single');
25852:     { TDoubleListSortCompare
25853:    = function (Item1, Item2: Double): Integer;
25854:     TDoubleDescriptor = function (Index:Integer;Item : Double) : string;  }
25855:      CL.AddTypeS('TDoubleListSortCompare', 'function (Item1,Item2: Double):Integer;
25856:    //TSingleDescriptor //=// function (Index:Integer;Item : Single) : string;
25857:     CL.AddTypeS('TDoubleDescriptor', 'function (Index:Integer;Item : Double) : string;
25858:     CL.AddTypeS('TDoubleSortOption', '( DoubleSortNone, DoubleSortUpWithDup, Doub'
25859:    +'leSortUpNoDup, DoubleSortDownWithDup, DoubleSortDownNoDup );
25860:   SIRegister_TDoubleList(CL);
25861:  CL.AddDelphiFunction('Function DefDescDouble( Index : Integer; Item : Double) : string');
25862: end;
25863:
25864: procedure SIRegister_ByteListClass(CL: TPSPascalCompiler);
25865: begin
25866:  CL.AddConstantN('SByteListVoidError','String').SetString( 'Invalid method call (empty list)!');
25867:  CL.AddConstantN('SByteListSortError','String').SetString( 'Invalid method call (sorted list)!');
25868:   CL.AddTypeS('TByteSortOption', '( ByteSortNone, ByteSortUpWithDup, ByteSortUp'
25869:    +'NoDup, ByteSortDownWithDup, ByteSortDownNoDup );
25870:     CL.AddTypeS('TBytePtrList', 'array of Byte');
25871:     TByteListSortCompare', 'function (Item1, Item2: Byte): Integer;
25872:     TByteDescriptor', 'function (Index:Integer;Item : Byte) : string;
25873:   SIRegister_TByteListClass(CL);
25874:  CL.AddDelphiFunction('Function DefDescByte( Index : Integer; Item : Byte): string');
25875: end;
25876:
25877: procedure SIRegister_neuralplanbuilder(CL: TPSPascalCompiler);
25878: begin
25879:  'NeuralMaxStates2','LongInt').SetInt( 400);
25880:  'NeuralMaxPlans2','LongInt').SetInt( 100);
25881:   CL.AddTypeS('TNeuralState2', 'array of byte');
25882:   CL.AddTypeS('TProcPred2', Function ( var ST : array of byte; Action : byte): boolean');
25883:   SIRegister_TActionStateList(CL);
25884:   Procedure TVisitedStatesCopy( var A, B : TActionStateList);
25885:   SIRegister_TPlan(CL);
25886:   SIRegister_TCompositePlan(CL);
25887: end;
25888:
25889: procedure SIRegister_neuralvolume(CL: TPSPascalCompiler);
25890: begin
25891:  'csMinAvxSize','LongInt').SetInt( 16);
25892:  'csEncodeRGB','LongInt').SetInt( 0);
25893:  'csEncodeHSV','LongInt').SetInt( 1);
25894:  'csEncodeHSL','LongInt').SetInt( 2);
```

```
25895:  'csEncodeLAB','LongInt').SetInt( 3);
25896:  'csEncodeGray','LongInt').SetInt( 4);
25897:   CL.AddTypeS('TNeuralFloat', 'Single');
25898:   //CL.AddTypeS('TNeuralFloatPtr', '^TNeuralFloat // will not work');
25899:   //CL.AddTypeS('TNeuralFloatArrPtr', '^TNeuralFloatArr // will not work');
25900:   CL.AddTypeS('TNeuralIntegerArray', 'array of integer');
25901:   //CL.AddTypeS('T', 'TNeuralFloat');
25902:   CL.AddTypeS('PtrInt', 'Integer');
25903:   SIRegister_TNNetList(CL);
25904:   SIRegister_TVolume(CL);
25905:   SIRegister_TNNetVolume(CL);
25906:   SIRegister_TNNetVolumePair(CL);
25907:   SIRegister_TMObject(CL);
25908:   SIRegister_TNNetVolumeList(CL);
25909:   SIRegister_TNNetVolumePairList(CL);
25910:   SIRegister_TNNetKMeans(CL);
25911:   SIRegister_TNNetStringList(CL);
25912:   SIRegister_TStringListInt(CL);
25913:   SIRegister_TNNetDictionary(CL);
25914:  Function CreateTokenizedStringList86( str : string; c : char) : TStringList;
25915:  Function CreateTokenizedStringList87( c : char) : TStringList;
25916:  Function HiperbolicTangent( x : TNeuralFloat) : TNeuralFloat');
25917:  Function HiperbolicTangentDerivative( x : TNeuralFloat) : TNeuralFloat');
25918:  Function RectifiedLinearUnit( x : TNeuralFloat) : TNeuralFloat');
25919:  Function RectifiedLinearUnitDerivative( x : TNeuralFloat) : TNeuralFloat');
25920:  Function RectifiedLinearUnitLeaky( x : TNeuralFloat) : TNeuralFloat');
25921:  Function RectifiedLinearUnitLeakyDerivative( x : TNeuralFloat) : TNeuralFloat');
25922:  Function ReLULeakyBound( x : TNeuralFloat) : TNeuralFloat');
25923:  Function ReLULeakyBoundDerivative( x : TNeuralFloat) : TNeuralFloat');
25924:  Function Sigmoid( x : TNeuralFloat) : TNeuralFloat');
25925:  Function SigmoidDerivative( x : TNeuralFloat) : TNeuralFloat');
25926:  Function Identity( x : TNeuralFloat) : TNeuralFloat');
25927:  Function IdentityDerivative( x : TNeuralFloat) : TNeuralFloat');
25928:  Function SoftmaxDerivative( x : TNeuralFloat) : TNeuralFloat');
25929:  Function DiffAct( x : TNeuralFloat) : TNeuralFloat');
25930:  Function DiffActDerivative( x : TNeuralFloat) : TNeuralFloat');
25931:  Function NeuronForceMinMax88( x, pMin, pMax : TNeuralFloat) : TNeuralFloat;
25932:  Function NeuronForceMinMax89( x, pMin, pMax : integer) : integer;
25933:  Function NeuronForceRange( x, range : TNeuralFloat) : TNeuralFloat');
25934:  Function NeuronForceMinRange( x, range : TNeuralFloat) : TNeuralFloat');
25935:  Procedure rgb2hsv( r, g, b : TNeuralFloat; var h, s, v : TNeuralFloat);
25936:  Procedure hsv2rgb( h, s, v : TNeuralFloat; var r, g, b : TNeuralFloat);
25937:  Function hue2rgb( p, q, t : TNeuralFloat) : TNeuralFloat');
25938:  Procedure rgb2hsl( r, g, b : TNeuralFloat; var h, s, l : TNeuralFloat);
25939:  Procedure hsl2rgb( h, s, l : TNeuralFloat; var r, g, b : TNeuralFloat);
25940:  Procedure lab2rgb( l, a, b : TNeuralFloat; var r, g, bb : TNeuralFloat);
25941:  Procedure rgb2lab( r, g, b : TNeuralFloat; var l, a, bb : TNeuralFloat);
25942:  Function RoundAsByte( x : TNeuralFloat) : byte');
25943:  Function CompareStringListIntegerAsc( List : TStringList; Index1, Index2 : Integer) : Integer');
25944:  Function CompareStringListIntegerDesc( List : TStringList; Index1, Index2 : Integer) : Integer');
25945:  Function CompareNNetVolumeListAsc( const Item1, Item2 : TNNetVolume) : Integer');
25946:  Function CompareNNetVolumeListDesc( const Item1, Item2 : TNNetVolume) : Integer');
25947:  Function NeuralFloatToStr( V : TNeuralFloat) : string');
25948:  Function NeuralStrToFloat( V : String) : TNeuralFloat');
25949:  Procedure TestTNNetVolume( );    //messagedialog!
25950:  Procedure TestKMeans2( );         //unattendend
25951: end;
25952:
25953:  (*----------------------------------------------------------------------------*)
25954: procedure SIRegister_neuralvolumev(CL: TPSPascalCompiler);
25955: begin
25956:  Procedure SaveHandleToBitmap( OutputFileName : string; hWnd : HWND);
25957:  Procedure LoadVolumeIntoTImage( V : TNNetVolume; Image : TImage; color_encoding : integer);
25958:  Procedure LoadRGBVolumeIntoTImage( V : TNNetVolume; Image : TImage);
25959:  Procedure LoadPictureIntoVolume( LocalPicture : TPicture; Vol : TNNetVolume);
25960:  Procedure LoadBitmapIntoVolume( LocalBitmap : TBitmap; Vol : TNNetVolume);
25961:  Procedure LoadImageFromFileIntoVolume( ImageFileName : string; V : TNNetVolume);
25962: end;
25963:
25964: function GetSmallestIdxInRange(StartPos, Len: integer): integer;
25965: function GetValueCount(Value: T): integer;
25966: GetParentProcessName2
25967: writeln('GetParentProcessName2: '+GetProcessName(GetParentProcessID(getprocessID)));
25968: Function DefDescByte( Index : Integer; Item : Byte) : string');
25969: function NumStringParts(SourceStr,Delimiter:String):Integer');
25970: function GetStringPart(SourceStr,Delimiter:String;Num:Integer):string');
25971: Function InverseCompareResult( const C : TCompareResult) : TCompareResult');
25972:
25973: function GetWindowProcessID(const Wnd: Windows.HWND): Windows.DWORD;
25974: var
25975:   GetWindowThreadProcessId: function(Wnd: Windows.HWND; // API fn prototype
25976:     lpdwProcessId: Windows.PDWORD): Windows.DWORD; stdcall;
25977: begin
25978:   // We load the API function explicitly to make the routine compatible with as
25979:   // many compilers as possible: the definition of GetWindowThreadProcessId in
25980:   // Delphi and FreePascal Windows units varies across compilers
25981:   GetWindowThreadProcessId := GetProcAddress(
25982:     GetModuleHandle('user32.dll'), 'GetWindowThreadProcessId' );
25983:   if Assigned(GetWindowThreadProcessId) and Windows.IsWindow(Wnd) then
```

```
25984:     GetWindowThreadProcessId(Wnd, @Result)
25985:   else
25986:     Result := Windows.DWORD(-1);
25987: end;
25988:
25989:
25990: Procedure TestVectorClassExtended  - type  TInt64Array = TInt64ArrayClass;
25991:
25992: procedure SIRegister_TVectorClass(CL: TPSPascalCompiler);
25993: begin
25994:   //with RegClassS(CL,'TVectorBaseArray', 'TVectorClass') do
25995:   //with CL.AddClassN(CL.FindClass('TVectorBaseArray'),'TVectorClass') do
25996:   with CL.AddClassN(CL.FindClass('TExtendedArray'),'TVectorClass') do begin
25997:     Function CreateInstance : AType');
25998:     Procedure Add( const V : MFloat);
25999:     Procedure Add1( const V : PMFloat; const Count : Integer);
26000:     Procedure Add2( const V : PInt64; const Count : Integer);
26001:     Procedure Add3( const V : MFloatArray);
26002:     Procedure Add4( const V : Int64Array);
26003:     Procedure Add5( const V : TVectorBaseArray);
26004:     Procedure Add6( const V : TInt64Array);
26005:     Procedure Add7( const V : TObject);
26006:     Procedure Subtract8( const V : MFloat);
26007:     Procedure Subtract9( const V : PMFloat; const Count : Integer);
26008:     Procedure Subtract10( const V : PInt64; const Count : Integer);
26009:     Procedure Subtract11( const V : MFloatArray);
26010:     Procedure Subtract12( const V : Int64Array);
26011:     Procedure Subtract13( const V : TVectorBaseArray);
26012:     Procedure Subtract14( const V : TInt64Array);
26013:     Procedure Subtract15( const V : TObject);
26014:     Procedure Multiply16( const V : MFloat);
26015:     Procedure Multiply17( const V : PMFloat; const Count : Integer);
26016:     Procedure Multiply18( const V : PInt64; const Count : Integer);
26017:     Procedure Multiply19( const V : MFloatArray);
26018:     Procedure Multiply20( const V : Int64Array);
26019:     Procedure Multiply21( const V : TVectorBaseArray);
26020:     Procedure Multiply22( const V : TInt64Array);
26021:     Procedure Multiply23( const V : TObject);
26022:     Function DotProduct24( const V : PMFloat; const Count : Integer): MFloat;
26023:     Function DotProduct25( const V : PInt64; const Count : Integer) : MFloat;
26024:     Function DotProduct26( const V : MFloatArray) : MFloat;
26025:     Function DotProduct27( const V : Int64Array) : MFloat;
26026:     Function DotProduct28( const V : TVectorBaseArray) : MFloat;
26027:     Function DotProduct29( const V : TInt64Array) : MFloat;
26028:     Function DotProduct30( const V : TObject) : MFloat;
26029:     Function Norm : MFloat');
26030:     Function Min : MFloat');
26031:     Function Max : MFloat');
26032:     Function Range( var Min, Max : MFloat) : MFloat');
26033:     Function IsZero( const CompareDelta : MFloat) : Boolean');
26034:     Function HasZero( const CompareDelta : MFloat) : Boolean');
26035:     Function HasNegative : Boolean');
26036:     Procedure Normalize');
26037:     Procedure Negate');
26038:     Procedure ValuesInvert');
26039:     Procedure ValuesSqr');
26040:     Procedure ValuesSqrt');
26041:     Function Sum : MFloat');
26042:     Function SumOfSquares : MFloat');
26043:     Procedure SumAndSquares( out Sum, SumOfSquares : MFloat);
26044:     Procedure SumAndCubes( out Sum, SumOfSquares, SumOfCubes : MFloat);
26045:     Procedure SumAndQuads( out Sum, SumOfSquares, SumOfCubes, SumOfQuads : MFloat);
26046:     Function WeightedSum( const Weights : TVector) : MFloat');
26047:     Function Mean : MFloat');
26048:     Function HarmonicMean : MFloat');
26049:     Function GeometricMean : MFloat');
26050:     Function Median : MFloat');
26051:     Function Mode : MFloat');
26052:     Function Variance : MFloat');
26053:     Function StdDev( var Mean : MFloat) : MFloat');
26054:     Function PopulationVariance : MFloat');
26055:     Function PopulationStdDev : MFloat');
26056:     Function M1 : MFloat');
26057:     Function M2 : MFloat');
26058:     Function M3 : MFloat');
26059:     Function M4 : MFloat');
26060:     Function Skew : MFloat');
26061:     Function Kurtosis : MFloat');
26062:     Function Product : MFloat');
26063:     Function Angle( const V : TVector) : MFloat');
26064:   end;
26065: end;
26066:
26067: (*----------------------------------------------------------------------------*)
26068: procedure SIRegister_TInt64ArrayClass(CL: TPSPascalCompiler);  BigDataClass
26069: begin
26070:   //with RegClassS(CL,'AInt64Array', 'TInt64ArrayClass') do
26071:   with CL.AddClassN(CL.FindClass('AInt64Array'),'TInt64ArrayClass') do begin
26072:     Constructor Create( const V : Int64Array);
```

```
26073:     Procedure ExchangeItems( const Idx1, Idx2 : Integer);
26074:     Function DuplicateRange( const LoIdx, HiIdx : Integer) : AArray');
26075:     Procedure Delete( const Idx : Integer; const ACount : Integer);
26076:     Procedure Insert( const Idx : Integer; const ACount : Integer);
26077:     Procedure Assign1( const V : Int64Array);
26078:     Procedure Assign2( const V : array of Int64);
26079:     Function AppendItem( const Value : Int64) : Integer');
26080:     ('Data', 'Int64Array', iptrw);
26081:     RegisterProperty('Count', 'Integer', iptrw);
26082:   end;
26083: end;
26084:
26085: procedure SIRegister_TExtendedArray(CL: TPSPascalCompiler);
26086: begin
26087:   //with RegClassS(CL,'AExtendedArray', 'TExtendedArray') do
26088:   with CL.AddClassN(CL.FindClass('AExtendedArray'),'TExtendedArray') do begin
26089:     Constructor Create( const V : ExtendedArray);
26090:     //RegisterMethod('Procedure Free;
26091:     Procedure Assign( const Source : TObject);
26092:     Procedure ExchangeItems( const Idx1, Idx2 : Integer);
26093:     Function DuplicateRange( const LoIdx, HiIdx : Integer) : AArray');
26094:     Procedure Delete( const Idx : Integer; const ACount : Integer);
26095:     Procedure Insert( const Idx : Integer; const ACount : Integer);
26096:     Procedure Assign2( const V : ExtendedArray);
26097:     //RegisterMethod('Procedure Assign( const V : array of Extended);
26098:     Function AppendItem( const Value : Extended) : Integer');
26099:     RegisterProperty('Data', 'ExtendedArray', iptrw);
26100:     RegisterProperty('Count', 'Integer', iptrw);
26101:   end;
26102: end;
26103:
26104: (*----------------------------------------------------------------------------*)
26105: procedure SIRegister_neuraldatasets(CL: TPSPascalCompiler);
26106: begin
26107:   TTinyImageChannel','array [0..31] of array[0..31] of byte; ');
26108:   TTinyImageChannel1D','array [0..32 * 32 - 1] of byte; ');
26109:   TMNistImage','array [0..27] of array[0..27] of byte; ');
26110:   //TTinyImageChannel1D = packed array [0..32 * 32 - 1] of byte;
26111:   //TMNistImage = packed array [0..27, 0..27] of byte;
26112:   TTinyImage', 'record bLabel : byte; R : TTinyImageChannel; G : T'
26113:    +'TinyImageChannel; B : TTinyImageChannel; end');
26114:   TCifar100Image', 'record bCoarseLabel : byte; bFineLabel : byte;'
26115:    +' R : TTinyImageChannel; G : TTinyImageChannel; B : TTinyImageChannel; end');
26116:   TTinySingleChannelImage', 'record bLabel : byte; Grey : TTinyImageChannel; end');
26117:   TTinySingleChannelImage1D', 'record bLabel : byte; Grey : TTinyImageChannel1D; end');
26118:   //CL.AddTypeS('TTinySingleChannelImagePtr', '^TTinySingleChannelImage // will not work');
26119:   //CL.AddTypeS('TTinySingleChannelImage1DPtr', '^TTinySingleChannelImage1D // will not work');
26120:   SIRegister_TFileNameList(CL);
26121:   SIRegister_TClassesAndElements(CL);
26122:  Procedure CreateVolumesFromImagesFromFolder(out ImgTrainingVolumes,ImgValidationVolumes,
        ImgTestVolumes:TNNetVolumeList;FolderName,pImageSubFolder:string;color_encoding:integer;TrainingProp,
        ValidationProp,TestProp:single;'+'NewSizeX:integer;NewSizeY:integer);
26123:  Procedure CreateFileNameListsFromImagesFromFolder( out TrainingFileNames, ValidationFileNames,
        TestFileNames: TFileNameList; FolderName, pImageSubFolder : string; TrainingProp, ValidationProp, TestProp
        : single);
26124:  //CL.AddDelphiFunction('Procedure LoadImageIntoVolume( M : TFPMemoryImage; Vol : TNNetVolume);
26125:  //CL.AddDelphiFunction('Procedure LoadVolumeIntoImage( Vol : TNNetVolume; M : TFPMemoryImage);
26126:  Function LoadImageFromFileIntoVolume( ImageFileName : string; V : TNNetVolume) : boolean');
26127:  Function SaveImageFromVolumeIntoFile( V : TNNetVolume; ImageFileName : string) : boolean');
26128:  Procedure ConfusionWriteCSVHeader( var CSVConfusion : TextFile; Labels : array of string);
26129:  Procedure ConfusionWriteCSV( var CSVConfusion : TextFile; Vol : TNNetVolume; Digits : integer);
26130:  Procedure LoadTinyImageIntoNNetVolume1( var TI : TTinyImage; Vol : TNNetVolume);
26131:  Procedure LoadTinyImageIntoNNetVolume2( var TI : TCifar100Image; Vol : TNNetVolume);
26132:  Procedure LoadTinyImageIntoNNetVolume3( var TI : TMNistImage; Vol : TNNetVolume);
26133:  Procedure LoadNNetVolumeIntoTinyImage4( Vol : TNNetVolume; var TI : TTinyImage);
26134:  Procedure LoadNNetVolumeIntoTinyImage5( Vol : TNNetVolume; var TI : TCifar100Image);
26135:  Procedure LoadTinySingleChannelIntoNNetVolume( var SC : TTinySingleChannelImage; Vol : TNNetVolume);
26136:  Procedure TinyImageCreateGrey( var TI : TTinyImage; var TIGrey : TTinySingleChannelImage);
26137:  Procedure TinyImageHE( var TI, TIHE : TTinySingleChannelImage);
26138:  Procedure TinyImageVE( var TI, TIVE : TTinySingleChannelImage);
26139:  Procedure TinyImageRemoveZeroGradient( var TI : TTinySingleChannelImage; distance : byte);
26140:  Procedure TinyImageHVE( var TI, TIHE : TTinySingleChannelImage);
26141:  Function TinyImageTo1D( var TI : TTinySingleChannelImage) : TTinySingleChannelImage1D');
26142:  Procedure CreateCifar10Volumes(out ImgTrainingVolumes,ImgValidationVolumes,
        ImgTestVolumes:TNNetVolumeList;color_encoding:byte);
26143:  Procedure CreateCifar100Volumes(out ImgTrainingVolumes,ImgValidationVolumes,
        ImgTestVolumes:TNNetVolumeList;color_encoding:byte;Verbose :bool;
26144:  Procedure CreateMNISTVolumes( out ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes :
        TNNetVolumeList; TrainFileName, TestFileName : string; Verbose : boolean; IsFashion : boolean);
26145:  Procedure loadCifar10Dataset6(ImgVolumes:TNNetVolumeList;idx:integer;base_pos:
        integer;color_encoding:byte););
26146:  Procedure
        loadCifar10Dataset7(ImgVolumes:TNNetVolumeList;fileName:string;base_pos:integer;color_encoding:byte);
26147:  Procedure loadCifar100Dataset(ImgVolumes TNNetVolumeList;fileName: string;
        color_encoding:byte;Verbose:bool);
26148:  Procedure
        loadMNISTDataset(ImgVolumes:TNNetVolumeList;fileName:str;Verbose:bool;IsFashion:bool;MaxLabel:int);
26149:  Function CheckCIFARFile( ) : boolean');
26150:  Function CheckCIFAR100File( ) : boolean');
```

```
26151:  Function CheckMNISTFile( fileName : string; IsFasion : boolean) : boolean');
26152:  Procedure TestBatch(NN TNNet;ImgVolumes:TNNetVolumeList;SampleSize:integer;out Rate,Loss,
        ErrorSum:TNeuralFloat)
26153:  Procedure TranslateCifar10VolumesToMachineAnimal( VolumeList : TNNetVolumeList);
26154:  Function SwapEndian( I : integer) : integer');
26155: end;
26156:
26157: (*----------------------------------------------------------------------------*)
26158: procedure SIRegister_neuralnetworkCAI(CL: TPSPascalCompiler);
26159: begin
26160:  CL.AddConstantN('csMaxInterleavedSize','integer').SetInt( 95);
26161:  CL.AddTypeS('TNeuralFloatDynArr', 'array of TNeuralFloat');
26162:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNet');
26163:   SIRegister_TNNetNeuron(CL);
26164:   SIRegister_TNNetNeuronList(CL);
26165:  CL.AddConstantN('csNNetMaxParameterIdx','LongInt').SetInt( 7);
26166:   SIRegister_TNNetLayer(CL);
26167:   //CL.AddTypeS('TNNetLayerClass', 'class of TNNetLayer');
26168:   SIRegister_TNNetLayerConcatedWeights(CL);
26169:   SIRegister_TNNetLayerList(CL);
26170:   SIRegister_TNNetInputBase(CL);
26171:   SIRegister_TNNetInput(CL);
26172:   SIRegister_TNNetIdentity(CL);
26173:   SIRegister_TNNetPad(CL);
26174:   SIRegister_TNNetIdentityWithoutL2(CL);
26175:   SIRegister_TNNetIdentityWithoutBackprop(CL);
26176:   //CL.AddTypeS('TNNetActivationFunctionClass', 'class of TNNetIdentity');
26177:   SIRegister_TNNetReLUBase(CL);
26178:   SIRegister_TNNetDigital(CL);
26179:   SIRegister_TNNetReLU(CL);
26180:   SIRegister_TNNetReLUL(CL);
26181:   SIRegister_TNNetSELU(CL);
26182:   SIRegister_TNNetSwish(CL);
26183:   SIRegister_TNNetReLUSqrt(CL);
26184:   SIRegister_TNNetPower(CL);
26185:   SIRegister_TNNetLeakyReLU(CL);
26186:   SIRegister_TNNetVeryLeakyReLU(CL);
26187:   SIRegister_TNNetSigmoid(CL);
26188:   SIRegister_TNNetHyperbolicTangent(CL);
26189:   SIRegister_TNNetMulLearning(CL);
26190:   SIRegister_TNNetMulByConstant(CL);
26191:   SIRegister_TNNetNegate(CL);
26192:   SIRegister_TNNetAddAndDiv(CL);
26193:   SIRegister_TNNetAddNoiseBase(CL);
26194:   SIRegister_TNNetDropout(CL);
26195:   SIRegister_TNNetRandomMulAdd(CL);
26196:   SIRegister_TNNetChannelRandomMulAdd(CL);
26197:   SIRegister_TNNetLayerMaxNormalization(CL);
26198:   SIRegister_TNNetLayerStdNormalization(CL);
26199:   SIRegister_TNNetMovingStdNormalization(CL);
26200:   SIRegister_TNNetChannelTransformBase(CL);
26201:   SIRegister_TNNetChannelShiftBase(CL);
26202:   SIRegister_TNNetChannelBias(CL);
26203:   SIRegister_TNNetChannelMul(CL);
26204:   SIRegister_TNNetChannelMulByLayer(CL);
26205:   SIRegister_TNNetCellMulByCell(CL);
26206:   SIRegister_TNNetCellBias(CL);
26207:   SIRegister_TNNetCellMul(CL);
26208:   SIRegister_TNNetChannelZeroCenter(CL);
26209:   SIRegister_TNNetChannelStdNormalization(CL);
26210:   SIRegister_TNNetLocalResponseNorm2D(CL);
26211:   SIRegister_TNNetInterleaveChannels(CL);
26212:   SIRegister_TNNetLocalResponseNormDepth(CL);
26213:   SIRegister_TNNetReshape(CL);
26214:   SIRegister_TNNetConcatBase(CL);
26215:   SIRegister_TNNetConcat(CL);
26216:   SIRegister_TNNetDeepConcat(CL);
26217:   SIRegister_TNNetSum(CL);
26218:   SIRegister_TNNetSplitChannels(CL);
26219:   SIRegister_TNNetSplitChannelEvery(CL);
26220:   SIRegister_TNNetFullConnect(CL);
26221:   //CL.AddTypeS('TNNetFullConnectClass', 'class of TNNetFullConnect');
26222:   SIRegister_TNNetFullConnectLinear(CL);
26223:   SIRegister_TNNetFullConnectSigmoid(CL);
26224:   SIRegister_TNNetFullConnectReLU(CL);
26225:   SIRegister_TNNetFullConnectDiff(CL);
26226:   SIRegister_TNNetSoftMax(CL);
26227:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetLayerFullConnect');
26228:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetLayerFullConnectReLU');
26229:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetLayerSoftMax');
26230:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetDense');
26231:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetDenseReLU');
26232:   SIRegister_TNNetConvolutionAbstract(CL);
26233:   SIRegister_TNNetDepthwiseConv(CL);
26234:   SIRegister_TNNetDepthwiseConvLinear(CL);
26235:   SIRegister_TNNetDepthwiseConvReLU(CL);
26236:   SIRegister_TNNetConvolutionBase(CL);
26237:   //CL.AddTypeS('TNNetConvolutionClass', 'class of TNNetConvolutionBase');
26238:   SIRegister_TNNetGroupedConvolutionLinear(CL);
```

```
26239:   SIRegister_TNNetGroupedConvolutionReLU(CL);
26240:   SIRegister_TNNetGroupedPointwiseConvLinear(CL);
26241:   //CL.AddTypeS('TNNetGroupedPointwiseConvClass', 'class of TNNetGroupedPointwiseConvLinear');
26242:   SIRegister_TNNetGroupedPointwiseConvReLU(CL);
26243:   SIRegister_TNNetConvolution(CL);
26244:   SIRegister_TNNetConvolutionSharedWeights(CL);
26245:   SIRegister_TNNetConvolutionLinear(CL);
26246:   SIRegister_TNNetConvolutionReLU(CL);
26247:   SIRegister_TNNetPointwiseConv(CL);
26248:   SIRegister_TNNetPointwiseConvLinear(CL);
26249:   SIRegister_TNNetPointwiseConvReLU(CL);
26250:   SIRegister_TNNetDeconvolution(CL);
26251:   SIRegister_TNNetDeconvolutionReLU(CL);
26252:   SIRegister_TNNetLocalConnect(CL);
26253:   SIRegister_TNNetLocalProduct(CL);
26254:   SIRegister_TNNetDeLocalConnect(CL);
26255:   SIRegister_TNNetLocalConnectReLU(CL);
26256:   SIRegister_TNNetDeLocalConnectReLU(CL);
26257:   SIRegister_TNNetPoolBase(CL);
26258:   SIRegister_TNNetMaxPool(CL);
26259:   SIRegister_TNNetMaxPoolPortable(CL);
26260:   SIRegister_TNNetMinPool(CL);
26261:   SIRegister_TNNetMaxChannel(CL);
26262:   SIRegister_TNNetMinChannel(CL);
26263:   SIRegister_TNNetAvgPool(CL);
26264:   SIRegister_TNNetAvgChannel(CL);
26265:   SIRegister_TNNetDeMaxPool(CL);
26266:   SIRegister_TNNetUpsample(CL);
26267:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNNetDeAvgPool');
26268:   SIRegister_TNNet(CL);
26269:   SIRegister_THistoricalNets(CL);
26270:   SIRegister_TNNetDataParallelism(CL);
26271:   SIRegister_TNNetByteProcessing(CL);
26272:   SIRegister_TNNetForByteProcessing(CL);
26273:   SIRegister_TBytePredictionViaNNet(CL);
26274:   SIRegister_TEasyBytePredictionViaNNet(CL);
26275:  Procedure CompareComputing( NN1, NN2 : TNNet);
26276:  Procedure CompareNNStructure( NN, NN2 : TNNet);
26277:  Procedure TestConvolutionAPI( );
26278:  Procedure TestDataParallelism( NN : TNNet);
26279:  //CL.AddDelphiFunction('Procedure
       TestConvolutionOpenCL(platform_id:cl_platform_id;device_id:cl_device_id);
26280:  //CL.AddDelphiFunction('Procedure
       TestFullConnectOpenCL(platform_id:cl_platform_id;device_id:cl_device_id);
26281:  Procedure RebuildPatternOnPreviousPatterns( Calculated : TNNetVolume; LocalWeight : TNNetVolume;
       PrevLayer : TNNetNeuronList; PrevStride : integer; ReLU : boolean; Threshold : TNeuralFloat);
26282:  Procedure RebuildNeuronListOnPreviousPatterns( CalculatedLayer : TNNetNeuronList; CurrentLayer, PrevLayer
       : TNNetNeuronList; PrevStride:integer;ReLU:boolean;Threshold:TNeuralFloat); InputLayer :=
26283:  function NN.AddDenseNetBlockCAI70(InnerConvNum,iConvNeuronCount,0, netconv,ChkUseSeparableConv.Checked,
       ChkMovingNorm.Checked,nil,nil, 0,1,1.0,1,1,1);
26284: end;
26285:
26286: (*----------------------------------------------------------------------------*)
26287: procedure SIRegister_neuralfit(CL: TPSPascalCompiler);
26288: begin
26289:   TCustomLearningRateScheduleObjFn', 'Function ( Epoch : integer) : single');
26290:   TMultiThreadProcItem', 'TObject');
26291:   SIRegister_TNeuralFitBase(CL);
26292:   SIRegister_TNeuralFitWithImageBase(CL);
26293:   TNNetDataAugmentationFn', 'Procedure ( pInput : TNNetVolume; ThreadId : integer);
26294:   TNNetLossFn', 'Function ( ExpectedOutput, FoundOutput : TNNetVolume; ThreadId : integer) :
       TNeuralFloat');
26295:   TNNetGetPairFn', 'Function ( Idx : integer; ThreadId : integer) : TNNetVolumePair');
26296:   TNNetGet2VolumesProc', 'Procedure ( Idx : integer; ThreadId : integer; pInput, pOutput : TNNetVolume);
26297:   SIRegister_TNeuralDataLoadingFit(CL);
26298:   SIRegister_TNeuralFit(CL);
26299:   SIRegister_TNeuralImageLoadingFit(CL);
26300:   SIRegister_TNeuralImageFit(CL);
26301:  Function MonopolarCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26302:  Function BipolarCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26303:  Function BipolarCompare99( A, B : TNNetVolume; ThreadId : integer) : boolean');
26304:  Function ClassCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26305: end;
26306: V 47620
26307:   FFit.FitLoading2(FDiscriminator, 64*10, 500, 500, 64, 35000,
26308:                 @GetDiscriminatorTrainingPair, nil, nil); // This
26309:   FFit.FitLoading22(FDiscriminator, 64*10, 500, 500, 64, 35000,
26310:                 @GetDiscriminatorTrainingProc, nil, nil);  //}
26311:
26312: procedure SIRegister_neuralgeneric(CL: TPSPascalCompiler);
26313: begin
26314:   SIRegister_TIncDec(CL);
26315:   SIRegister_TRandom(CL);
26316:  Function MaxSingle2( x, y : single) : single');
26317:  Function GetMaxDivisor( x, acceptableMax : integer) : integer');
26318:  Function GetMaxAcceptableCommonDivisor( a, b : integer; max_acceptable : integer) : integer');
26319: end;
26320:
26321:  (*----------------------------------------------------------------------------*)
```

```
26322: procedure SIRegister_neuraldatasetsv(CL: TPSPascalCompiler);
26323: begin
26324:   TImageDynArr', 'array of TImage');
26325:   TLabelDynArr', 'array of TLabel');
26326:  Procedure LoadTinyImageIntoTImage( var TI : TTinyImage; var Image : TImage);
26327:  Procedure LoadTISingleChannelIntoImage( var TI : TTinySingleChannelImage; var Image : TImage);
26328:  Procedure ShowNeurons(pNeuronList:TNNetNeuronList;var pImage:TImageDynArr;startImage,filterSize,
       color_encoding:int;ScalePerImage:bool;
26329:  Procedure CreateAscentImages(GrBoxNeurons: TGroupBox; var pImage: TImageDynArr; var pLabelX, pLabelY:
       TLabelDynArr; ImagesNum : integer; inputSize, displaySize, imagesPerRow : integer);
26330:  Procedure CreateNeuronImages(GrBoxNeurons: TGroupBox; var pImage: TImageDynArr; var pLabelX, pLabelY:
       TLabelDynArr; pNeuronList : TNNetNeuronList; filterSize, imagesPerRow, NeuronNum : integer);
26331:  Procedure FreeNeuronImages( var pImage : TImageDynArr; var pLabelX, pLabelY : TLabelDynArr);
26332:  Procedure LoadNNLayersIntoCombo( NN : TNNet; Combo : TComboBox);
26333: end;
26334:
26335: (*----------------------------------------------------------------------------*)
26336: procedure SIRegister_flcFloats(CL: TPSPascalCompiler);
26337: begin
26338:  Function flcDoubleMin( const A, B : Double) : Double');
26339:  Function flcDoubleMax( const A, B : Double) : Double');
26340:  Function flcExtendedMin( const A, B : Extended) : Extended');
26341:  Function flcExtendedMax( const A, B : Extended) : Extended');
26342:  Function flcFloatMin( const A, B : Float) : Float');
26343:  Function flcFloatMax( const A, B : Float) : Float');
26344:  Function flcFloatClip( const Value : Float; const Low, High : Float) : Float');
26345:  Function flcInSingleRange( const A : Float) : Boolean');
26346:  Function flcInDoubleRange( const A : Float) : Boolean');
26347:  Function flcInCurrencyRange0( const A : Float) : Boolean;
26348:  Function flcInCurrencyRange1( const A : Int64) : Boolean;
26349:  Function flcExtendedExponentBase2( const A : Extended; var Exponent : Integer) : Boolean');
26350:  Function flcExtendedExponentBase10( const A : Extended; var Exponent : Integer): Boolean');
26351:  Function flcExtendedIsInfinity( const A : Extended) : Boolean');
26352:  Function flcExtendedIsNaN( const A : Extended) : Boolean');
26353:  CL.AddConstantN('flcSingleCompareDelta','Extended').setExtended( 1.0E-34);
26354:  flcDoubleCompareDelta','Extended').setExtended( 1.0E-280);
26355:  ExtendedCompareDelta','').SetString( DoubleCompareDelta);
26356:  flcExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
26357:  DefaultCompareDelta','').SetString( SingleCompareDelta);
26358:  Function flcFloatZero( const A : Float; const CompareDelta : Float) : Boolean');
26359:  Function flcFloatOne( const A : Float; const CompareDelta : Float) : Boolean');
26360:  Function flcFloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean');
26361:  Function flcFloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult');
26362:  flcSingleCompareEpsilon','Extended).setExtended( 1.0E-5);
26363:  flcDoubleCompareEpsilon','Extended).setExtended( 1.0E-13);
26364:  flcExtendedCompareEpsilon','Extended).setExtended( 1.0E-17);
26365:  flcDefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
26366:  Function flcExtendedApproxEqual( const A, B : Extended; const CompareEpsilon : Double): Boolean');
26367:  Function flcExtendedApproxCompare( const A, B : Extended; const CompareEpsilon Double): TCompareResult');
26368:  Function flcDoubleApproxEqual( const A, B : Double; const CompareEpsilon : Double): Boolean');
26369:  Function flcDoubleApproxCompare(const A,B: Double; const CompareEpsilon: Double): TCompareResult');
26370:  Function flcFloatApproxEqual( const A, B : Float; const CompareEpsilon : Float): Boolean');
26371:  Function flcFloatApproxCompare( const A, B : Float; const CompareEpsilon : Float): TCompareResult');
26372:  Function flcFloatToStringA(const A : Float) : AnsiString');
26373:  Function flcFloatToStringB(const A : Float) : RawByteString');
26374:  Function flcFloatToStringU(const A : Float) : UnicodeString');
26375:  Function flcFloatToString(const A : Float) : String');
26376:  Function flcTryStringToFloatB(const A : RawByteString; out B : Float) : Boolean');
26377:  Function flcTryStringToFloatU(const A : UnicodeString; out B : Float) : Boolean');
26378:  Function flcTryStringToFloat(const A : String; out B : Float) : Boolean');
26379:  Function flcStringToFloatB(const A : RawByteString) : Float');
26380:  Function flcStringToFloatU(const A : UnicodeString) : Float');
26381:  Function flcStringToFloat(const A : String) : Float');
26382:  Function flcStringToFloatDefB(const A : RawByteString; const Default : Float): Float');
26383:  Function flcStringToFloatDefU(const A : UnicodeString; const Default : Float): Float');
26384:  Function flcStringToFloatDef(const A : String; const Default : Float) : Float');
26385:  Procedure FLCFloatStringTest');
26386:  Procedure FLCFloatTest');
26387: end;
26388:
26389: procedure SIRegister_neuralthread(CL: TPSPascalCompiler);
26390: begin
26391:   CL.AddTypeS('TNeuralProc', 'Procedure ( index, threadnum : integer);
26392:   SIRegister_TNeuralThread(CL);
26393:   SIRegister_TNeuralThreadList(CL);
26394:  Procedure NeuralThreadListCreate( pSize : integer);
26395:  Procedure NeuralThreadListFree( );
26396:  Function fNTL : TNeuralThreadList');
26397:  Procedure CreateNeuralThreadListIfRequired( );
26398:  Function NeuralDefaultThreadCount : integer');
26399:  Procedure NeuralInitCriticalSection( var pCritSec : TRTLCriticalSection);
26400:  Procedure NeuralDoneCriticalSection( var pCritSec : TRTLCriticalSection);
26401:  Procedure InitCriticalSection( var pCritSec : TRTLCriticalSection);
26402:  Procedure DoneCriticalSection( var pCritSec : TRTLCriticalSection);
26403:  Function neuralGetProcessId( ) : integer');
26404: end;
26405:
26406: procedure SIRegister_uSysTools(CL: TPSPascalCompiler);
26407: begin
```

```
26408:  Function
        usGetRegExFileList(APath:String;APattern:String;ASubDirs:Boolean;AResultList:TStrings;AAttributes:Integer;AWithPath:Bo
26409:  Procedure usFindFileInPaths(const AFileName:String;const ASearchPaths:String;AMinimumFileSize:Int64;const
        AResultList:TStrings);
26410:  Procedure usPrintWindow( Wnd : HWND; ATo : TBitmap);
26411:  Function usGetWindowByClassTree( const AClassTree: TStringList; AParent:HWND;
        AWithWildCards:Boolean):HWND');
26412:  Function usGetTempFile( AExtension : String) : String');
26413:  Function usDeleteFileToWasteBin( AFileName : string) : boolean');
26414:  Function usExpandEnvVars( AInputString : String) : String');
26415:  Function usFileSize( AFileName : String) : Int64');
26416:  Function usIsMultiprocessor : Boolean');
26417:  Function usProcessorCount : Cardinal');
26418:  Function usIsAdmin: Boolean');
26419:  Function usMultipleStringReplace(AString:Str;AOldPatterns,ANewPatterns:array of
        Str;AFlags:TReplaceFlags):Str;
26420:  Function usMakeFileName( ADesiredFName : String) : String');
26421:  Function usIsLike( AString, APattern : String) : Boolean');
26422:  Function usVarRecToVariant( AValue : TVarRec) : Variant');
26423:  Function usVariantToTypedVarRec( const Item : Variant; VarType : TVarType) : TVarRec');
26424:  Function usVariantToVarRec( const Item : Variant) : TVarRec');
26425:  Procedure usFinalizeVarRec( var Item : TVarRec);
26426:  Function usGetWindowText( wnd : HWND) : WideString');
26427:  Procedure usSetWindowText( wnd : HWND; txt : WideString);
26428:  Function usMultipleWideStringReplace(AString:WideString;AOldPatterns,ANewPatterns:array of WideString;
26429:                                          AFlags: TReplaceFlags):WideString');
26430:  Procedure usAcceptNumericOnly( var Key : Char; Comma : Boolean);
26431:  Function usSameGUID( AGUID1, AGUID2 : TGUID) : Boolean');
26432:  Function usIsBitSet( AValue : Integer; ABitIndex : Byte) : Boolean');
26433:  Function usClientToScreen( AWindow : HWND; var APoint : TPoint) : Boolean');
26434:  Function usGetDataObjectFromFileList( const Directory : string; Files : TStrings) : IDataObject');
26435:  Function usGetDataObjectFromFile( const Directory : string; AFile : String) : IDataObject');
26436:  Function usGetFileListFromDataObject( const DataObject : IDataObject; Files : TStrings) : Integer');
26437:  Function usKeyPressed( AKey : Smallint) : Boolean;
26438:  Function usKeyPressed1( AKey : Byte) : Boolean;
26439:  Function usKeyToogled( AKey : Smallint) : Boolean;
26440:  Function usKeyToogled1( AKey : Byte) : Boolean;
26441:  Function usGetShellFolder(CSIDL : integer) : string');
26442: end;
26443:
26444: function SerializeStream(const AStream : TStream) : String;
26445: procedure DeserializeToStream(const Data : String; const AStream : TStream);
26446: procedure ReplaceImageInImageList(FromImageList: TImageList; FromIndex: Integer;
26447:                                   ToImageList: TImageList; ToIndex: Integer);
26448: Function GetSIDStringFromUser( AUsername : AnsiString; AServerName : AnsiString) : String');
26449:
26450: TestConvolutionAPI();  procedure TestBackProp(); //> 1.5 GByte MemoryTest
26451: https://github.com/ariaghora/noe
26452: https://github.com/Fr0sT-Brutal/awesome-pascal#machine-learning
26453: https://github.com/TheUnknownOnes/theunknownones/tree/master/Libraries/PascalScriptAddons
26454:
26455: procedure SIRegister_URungeKutta4(CL: TPSPascalCompiler);
26456: begin
26457:  CL.AddConstantN('maxfuncs2','LongInt').SetInt( 10);
26458:   //CL.AddTypeS('float', 'extended');
26459:   CL.AddTypeS('TUserFunction2', 'Function ( T, X, XPrime : double) : double');
26460:   TUserCallBackFunction2', 'Function ( T, X, XPrime : double) : Boolean');
26461:   TNData2', 'record X : double; xPrime : double; end');
26462:   TNvector2', 'array[0..maxfuncs2] of TNData2;
26463:  // TFuncVect  = array[1..MaxFuncs] of TUserFunctionV;
26464:   // TNvector = array[0..maxfuncs] of TNData;
26465:   TUserFunctionV2', 'Function ( V : TNVector2) : double');
26466:   TFuncVect2', 'array[1..MaxFuncs2] of TUserFunctionV2;
26467:   TUserCallbackFunctionV2','Function (V:TNVector2) : boolean');
26468:  Procedure
        RungeKutta2ndOrderIC2(LowerLimit:double;UpperLimit:double;InitialValue:double;InitialDeriv:double;
        ReturnInterval:double;CalcInterval:double;var Error:byte;UserFunc:TUserFunctionV2;UserCallBack:
        TUserCallbackFunctionV2);
26469:  Procedure RungeKutta2ndOrderIC_System2(LowerLimit:double; UpperLimit: double; InitialValues : TNVector2;
        ReturnInterval: double; CalcInterval: double; var Error: byte; NumEquations: Integer; Vector: TFuncVect2;
        UserCallBack: TUserCallbackFunctionV2);
26470: end;
26471:
26472: procedure SIRegister_OverbyteIcsUtils(CL: TPSPascalCompiler);
26473: begin
26474:  CL.AddTypeS('TIcsDbcsLeadBytes', 'TSysCharset');
26475:  'MB_ERR_INVALID_CHARS','LongWord').SetUInt( $00000008);
26476:  'WC_ERR_INVALID_CHARS','LongWord').SetUInt( $80);
26477:  CP_UTF16','LongInt').SetInt( 1200);
26478:  CP_UTF16Be','LongInt').SetInt( 1201);
26479:  CP_UTF32','LongInt').SetInt( 12000);
26480:  CP_UTF32Be','LongInt').SetInt( 12001);
26481:   TOBJECT'),'EIcsStringConvertError');
26482:   TCharsetDetectResult', '(cdrAscii,cdrUtf8,cdrUnknown );
26483:   TIcsNormForm','(icsNormalizationOther,icsNormalizationC,icsNormalizationD,icsNormalizationKD );
26484:  TIcsSearchRecW','record Time:Integer;Size:Integer;Attr: '
26485:    Integer;Name:String;ExcludeAttr:Integer;FindHandle:THandle;FindData:TWin32FindData;end');
26486:   TUnicode_String','record Length:Word;MaximumLength:Word;Buffer:WideChar;end');
26487:   TThreadID', 'LongWord;
```

```
26488:  // CL.AddTypeS('PUnicode_String', '^TUnicode_String // will not work');
26489:   SIRegister_TIcsFileStreamW(CL);
26490:  CL.AddConstantN('ICONV_UNICODE','String').SetString( 'UTF-16LE');
26491:  Function IcsIconvNameFromCodePage( CodePage : LongWord) : AnsiString');
26492:  Function IcsIsValidAnsiCodePage( const CP : LongWord) : Boolean');
26493:  Procedure IcsCharLowerA( var ACh : AnsiChar);
26494:  Function IcsGetCurrentThreadID : TThreadID');
26495:  Function IcsGetFreeDiskSpace( const APath : String) : Int64');
26496:  Function IcsGetLocalTimeZoneBias : LongInt');
26497:  Function IcsDateTimeToUTC( dtDT : TDateTime): TDateTime');
26498:  Function IcsUTCToDateTime( dtDT : TDateTime): TDateTime');
26499:  Function IcsGetTickCount : LongWord');
26500:  //AddDelphiFunction('Function IcsWcToMb(CodePage:LongWord; Flags:Cardinal;
        WStr:PWideChar;WStrLen:Integer; MbStr:PAnsiChar;MbStrLen:Integer;DefaultChar:PAnsiChar;UsedDefaultChar:PLon
26501:  //('Function IcsMbToWc(CodePage:LongWord;Flags:Cardinal;MbStr:PAnsiChar;MbStrLen:Int;
        WStr:PWideChar;WStrLen:Int):Int');
26502:  Function IcsGetDefaultWindowsUnicodeChar( CodePage : LongWord) : WideChar');
26503:  Function IcsGetDefaultWindowsAnsiChar( CodePage : LongWord) : AnsiChar');
26504:  Procedure IcsGetAcp( var CodePage : LongWord);
26505:  Function IcsIsDBCSCodePage( CodePage : LongWord) : Boolean');
26506:  Function IcsIsDBCSLeadByte( Ch : AnsiChar; CodePage : LongWord) : Boolean');
26507:  Function IcsIsMBCSCodePage( CodePage : LongWord) : Boolean');
26508:  Function IcsIsSBCSCodePage( CodePage : LongWord) : Boolean');
26509:  Function IcsGetLeadBytes( CodePage : LongWord) : TIcsDbcsLeadBytes');
26510:  Function icsUnicodeToUsAscii4( const Str: UnicodeString; FailCh : AnsiChar) : AnsiString;
26511:  Function icsUnicodeToUsAscii5( const Str: UnicodeString) : AnsiString;
26512:  Function icsUsAsciiToUnicode6( const Str: RawByteString; FailCh : AnsiChar) : UnicodeString;
26513:  Function icsUsAsciiToUnicode7( const Str: RawByteString) : UnicodeString;
26514:  Function icsUnicodeToAnsi8(const Str:WideChar;ACodePage LongWord; SetCodePage : Boolean) : RawByteString;
26515:  Function icsUnicodeToAnsi9(const Str:UnicodeString;ACodePage:LongWord;SetCodePage:Boolean):RawByteString;
26516:  Function icsUnicodeToAnsi10( const Str: UnicodeString) : RawByteString;
26517:  Function icsAnsiToUnicode11( const Str: AnsiChar; ACodePage : LongWord) : UnicodeString;
26518:  Function icsAnsiToUnicode12( const Str: RawByteString; ACodePage : LongWord) : UnicodeString;
26519:  CL.AddDelphiFunction('Function icsAnsiToUnicode13( const Str : RawByteString) : UnicodeString;
26520:  Function IcsBufferToUnicode14(const Buffer,BufferSize:Integer;BufferCodePage:LongWord;out FailedByteCount
        : Integer)UnicodeString;
26521:  Function IcsBufferToUnicode15(const Buffer,
        BufferSize:Integer;BufferCodePage:LongWord;RaiseFailedBytes:Bool): UnicodeString;
26522:  Function IcsGetWideCharCount(const Buffer,BufferSize:Integer;BufferCodePage:LongWord;out
        InvalidEndByteCount: Integer):Integer;
26523:  Function IcsGetWideChars(const Buffer,
        BufferSize:Integer;BufferCodePage:LongWord;Chars:WideChar;CharCount: Integer):Integer;
26524:  Function
        icsStreamWriteString16(AStream:TStream;Str:WideChar;cLen:Int;ACodePage:LongWord;WriteBOM:Bool):Int;
26525:  Function icsStreamWriteString17(AStream TStream;Str:WideChar;cLen :Integer;ACodePage : LongWord) :
        Integer;
26526:  Function icsStreamWriteString18(AStream:TStream;const Str:UnicodeString;ACodePage
        LongWord;WriteBOM:Bool):Int;
27527:  Function icsStreamWriteString19(AStream:TStream;const Str:UnicodeString;ACodePage : LongWord) : Integer;
26528:  Function icsStreamWriteString20( AStream : TStream; const Str : UnicodeString) : Integer;
26529:  Function icsIsUsAscii21( const Str : RawByteString) : Boolean;
26530:  Function icsIsUsAscii22( const Str : UnicodeString) : Boolean;
26531:  Procedure IcsAppendStr( var Dest : RawByteString; const Src : RawByteString);
26532:  Function icsatoi23( const Str : RawByteString) : Integer;
26533:  Function icsatoi24( const Str : UnicodeString) : Integer;
26534:  Function icsatoi6425( const Str : RawByteString) : Int64;
26535:  Function icsatoi6426( const Str : UnicodeString) : Int64;
26536:  Function IcsCalcTickDiff( const StartTick, EndTick : LongWord) : LongWord');
26537:  Function icsStringToUtf827( const Str : UnicodeString) : RawByteString;
26538:  Function icsStringToUtf828( const Str : RawByteString; ACodePage : LongWord) : RawByteString;
26539:  Function icsUtf8ToStringW( const Str : RawByteString) : UnicodeString');
26540:  Function icsUtf8ToStringA( const Str : RawByteString; ACodePage : LongWord) : AnsiString');
26541:  Function icsCheckUnicodeToAnsi( const Str : UnicodeString; ACodePage : LongWord) : Boolean');
26542:  Function icsIsUtf8TrailByte( const B : Byte) : Boolean');
26543:  Function icsIsUtf8LeadByte( const B : Byte) : Boolean');
26544:  Function IcsUtf8Size( const LeadByte : Byte) : Integer');
26545:  Function icsIsLeadChar( Ch : WideChar) : Boolean');
26546:  Function icsIsUtf8Valid29( const Str : RawByteString) : Boolean;
26547:  //Function IsUtf8Valid30( const Buf : Pointer; Len : Integer) : Boolean;
26548:  //Function CharsetDetect31( const Buf : Pointer; Len : Integer) : TCharsetDetectResult;
26549:  Function icsCharsetDetect32( const Str : RawByteString) : TCharsetDetectResult;
26550:  Function IcsCharNextUtf8( const Str : AnsiChar) : AnsiChar');
26551:  Function IcsCharPrevUtf8( const Start, Current : AnsiChar) : AnsiChar');
26552:  Function icsConvertCodepage(const
        Str:RawByteString;SrcCodePage:LongWord;DstCodePage:LongWord):RawByteString');
26553:  Function icshtoin33( Value : WideChar; Len : Integer) : Integer;
26554:  Function icshtoin34( Value : AnsiChar; Len : Integer) : Integer;
26555:  Function icshtoi235( value : WideChar) : Integer;
26556:  Function icshtoi236( value : AnsiChar) : Integer;
26557:  Function IcsBufferToHex37( const Buf, Size : Integer) : String;
26558:  Function IcsBufferToHex38( const Buf, Size : Integer; Separator : Char) : String;
26559:  Function icsIsXDigit39( Ch : WideChar) : Boolean;
26560:  Function icsIsXDigit40( Ch : AnsiChar) : Boolean;
26561:  Function icsXDigit41( Ch : WideChar) : Integer;
26562:  Function icsXDigit42( Ch : AnsiChar) : Integer;
26563:  Function icsIsCharInSysCharSet43( Ch : WideChar; const ASet : TSysCharSet):Boolean;
26564:  Function icsIsCharInSysCharSet44( Ch : AnsiChar; const ASet : TSysCharSet):Boolean;
26565:  Function icsIsDigit45( Ch : WideChar) : Boolean;
26566:  Function icsIsDigit46( Ch : AnsiChar) : Boolean;
```

```
26567:  Function icsIsSpace47( Ch : WideChar) : Boolean;
26568:  Function icsIsSpace48( Ch : AnsiChar) : Boolean;
26569:  Function icsIsCRLF49( Ch : WideChar) : Boolean;
26570:  Function icsIsCRLF50( Ch : AnsiChar) : Boolean;
26571:  Function icsIsSpaceOrCRLF51( Ch : WideChar) : Boolean;
26572:  Function icsIsSpaceOrCRLF52( Ch : AnsiChar) : Boolean;
26573:  Function icsXDigit2( S : PChar) : Integer');
26574:  Function icsstpblk53( PValue : WideChar) : WideChar;
26575:  Function icsstpblk54( PValue : AnsiChar) : AnsiChar;
26576:  Function IcsStrNextChar( const Str : AnsiChar; ACodePage : LongWord) : AnsiChar');
26577:  Function IcsStrPrevChar( const Start, Current : AnsiChar; ACodePage : LongWord) : AnsiChar');
26578:  Function IcsStrCharLength55( const Str: AnsiChar; ACodePage : LongWord) : Integer;
26579:  Function IcsNextCharIndex56( const S: RawByteString; Index : Integer; ACodePage: LongWord): Integer;
26580:  Function IcsGetBomBytes( ACodePage : LongWord) : TBytes');
26581:  Function IcsGetBufferCodepage57( Buf: AnsiChar; ByteCount : Integer) : LongWord;
26582:  Function IcsGetBufferCodepage58( Buf: AnsiChar; ByteCount : Integer; out BOMSize: Integer): LongWord;
26583:  Function IcsSwap16( Value : Word) : Word');
26584:  Procedure IcsSwap16Buf( Src, Dst : Word; WordCount : Integer);
26585:  Function IcsSwap32( Value : LongWord) : LongWord');
26586:  Procedure IcsSwap32Buf( Src, Dst : LongWord; LongWordCount : Integer);
26587:  Function IcsSwap64( Value : Int64) : Int64');
26588:  Procedure IcsSwap64Buf( Src, Dst : Int64; QuadWordCount : Integer);
26589:  Procedure IcsNameThreadForDebugging( AThreadName : AnsiString; AThreadID : TThreadID);
26590:  Function IcsNormalizeString( const S : UnicodeString; NormForm : TIcsNormForm) : UnicodeString');
26591:  Function IcsCryptGenRandom( var Buf, BufSize : Integer) : Boolean');
26592:  Function IcsRandomInt( const ARange : Integer) : Integer');
26593:  Function IcsFileUtcModified( const FileName : String) : TDateTime');
26594:  //Function IcsInterlockedCompareExchange(var Destination:Pointer;Exchange:Pointer;Comperand
        Pointer):Pointer');
26595:  Function IcsExtractFilePathW( const FileName : UnicodeString) : UnicodeString');
26596:  Function IcsExtractFileDirW( const FileName : UnicodeString) : UnicodeString');
26597:  Function IcsExtractFileDriveW( const FileName : UnicodeString) : UnicodeString');
26598:  Function IcsExtractFileNameW( const FileName : UnicodeString) : UnicodeString');
26599:  Function IcsExtractFileExtW( const FileName : UnicodeString) : UnicodeString');
26600:  Function IcsExpandFileNameW( const FileName : UnicodeString) : UnicodeString');
26601:  Function IcsExtractNameOnlyW( FileName : UnicodeString) : UnicodeString');
26602:  Function IcsChangeFileExtW( const FileName, Extension : UnicodeString) : UnicodeString');
26603:  Function IcsStrAllocW( Len : Cardinal) : WideChar');
26604:  Function IcsStrLenW( Str : WideChar) : Cardinal');
26605:  Function IcsAnsiCompareFileNameW59( const S1, S2 : UnicodeString) : Integer;
26606:  Function IcsAnsiCompareFileNameW60( const Utf8S1, Utf8S2 : UTF8String) : Integer;
26607:  Function IcsDirExistsW61( const FileName : WideChar) : Boolean;
26608:  Function IcsDirExistsW62( const FileName : UnicodeString) : Boolean;
26609:  Function IcsDirExistsW63( const Utf8FileName : UTF8String) : Boolean;
26610:  Function IcsFindFirstW64( const Path : UnicodeString; Attr : Integer; var F: TIcsSearchRecW):Integer;
26611:  Function IcsFindFirstW65( const Utf8Path : UTF8String; Attr : Integer;var F: TIcsSearchRecW):Integer;
26612:  Procedure IcsFindCloseW( var F : TIcsSearchRecW);
26613:  Function IcsFindNextW( var F : TIcsSearchRecW) : Integer');
26614:  Function IcsIncludeTrailingPathDelimiterW( const S : UnicodeString) : UnicodeString');
26615:  Function IcsExcludeTrailingPathDelimiterW( const S : UnicodeString) : UnicodeString');
26616:  Function IcsExtractLastDir66( const Path : RawByteString) : RawByteString;
26617:  Function IcsExtractLastDir67( const Path : UnicodeString) : UnicodeString;
26618:  Function IcsFileGetAttrW68( const FileName : UnicodeString) : Integer;
26619:  Function IcsFileGetAttrW69( const Utf8FileName : UTF8String) : Integer;
26620:  Function IcsFileSetAttrW70( const FileName : UnicodeString; Attr : Integer) : Integer;
26621:  Function IcsFileSetAttrW71( const Utf8FileName : UTF8String; Attr : Integer) : Integer;
26622:  Function IcsDeleteFileW72( const FileName : UnicodeString) : Boolean;
26623:  Function IcsDeleteFileW73( const Utf8FileName : UTF8String) : Boolean;
26624:  Function IcsRenameFileW74( const OldName, NewName : UnicodeString) : Boolean;
26625:  Function IcsRenameFileW75( const Utf8OldName, Utf8NewName : UTF8String) : Boolean;
26626:  Function IcsForceDirectoriesW76( Dir : UnicodeString) : Boolean;
26627:  Function IcsForceDirectoriesW77( Utf8Dir : UTF8String) : Boolean;
26628:  Function IcsCreateDirW78( const Dir : UnicodeString) : Boolean;
26629:  Function IcsCreateDirW79( const Utf8Dir : UTF8String) : Boolean;
26630:  Function IcsRemoveDirW80( const Dir : UnicodeString) : Boolean;
26631:  Function IcsRemoveDirW81( const Utf8Dir : UTF8String) : Boolean;
26632:  Function IcsFileAgeW82( const FileName : UnicodeString) : Integer;
26633:  Function IcsFileAgeW83( const Utf8FileName : UTF8String) : Integer;
26634:  Function IcsFileExistsW84( const FileName : UnicodeString) : Boolean;
26635:  Function IcsFileExistsW85( const Utf8FileName : UTF8String) : Boolean;
26636:  Function IcsAnsiLowerCaseW( const S : UnicodeString) : UnicodeString');
26637:  Function IcsAnsiUpperCaseW( const S : UnicodeString) : UnicodeString');
26638:  Function IcsMakeWord( L, H : Byte) : Word');
26639:  Function IcsMakeLong( L, H : Word) : Longint');
26640:  Function IcsHiWord( LW : LongWord) : Word');
26641:  Function IcsHiByte( W : Word) : Byte');
26642:  Function IcsLoByte( W : Word) : Byte');
26643:  Function IcsLoWord( LW : LongWord) : Word');
26644:  Procedure IcsCheckOSError( ALastError : Integer);
26645:  Function IcsIntToStrA( N : Integer) : AnsiString');
26646:  Function IcsIntToHexA( N : Integer; Digits : Byte) : AnsiString');
26647:  Function IcsTrim86( const Str : AnsiString) : AnsiString;
26648:  Function IcsTrim87( const Str : UnicodeString) : UnicodeString;
26649:  Function IcsLowerCase88( const S : AnsiString) : AnsiString;
26650:  Function IcsLowerCase89( const S : UnicodeString) : UnicodeString;
26651:  Function IcsUpperCase90( const S : AnsiString) : AnsiString;
26652:  Function IcsUpperCase91( const S : UnicodeString) : UnicodeString;
26653:  Function IcsUpperCaseA( const S : AnsiString) : AnsiString');
26654:  Function IcsLowerCaseA( const S : AnsiString) : AnsiString');
```

```
26655:  Function IcsCompareTextA( const S1, S2 : AnsiString) : Integer');
26656:  Function IcsTrimA( const Str : AnsiString) : AnsiString');
26657:  Function IcsSameTextA( const S1, S2 : AnsiString) : Boolean');
26658:  Function IcsCompareStr92( const S1, S2 : AnsiString) : Integer;
26659:  Function IcsCompareStr93( const S1, S2 : UnicodeString) : Integer;
26660:  Function IcsCompareText94( const S1, S2 : AnsiString) : Integer;
26661:  Function IcsCompareText95( const S1, S2 : UnicodeString) : Integer;
26662:  Function IcsStrLen96( const Str : AnsiChar) : Cardinal;
26663:  Function IcsStrLen97( const Str : WideChar) : Cardinal;
26664:  Function IcsStrPas98( const Str : AnsiChar) : AnsiString;
26665:  Function IcsStrPas99( const Str : WideChar) : string;
26666:  Function IcsStrCopy100( Dest : AnsiChar; const Source : AnsiChar) : AnsiChar;
26667:  Function IcsStrCopy101( Dest : WideChar; const Source : WideChar) : WideChar;
26668:  Function IcsStrPCopy102( Dest : PChar; const Source : string) : PChar;
26669:  Function IcsStrPCopy103( Dest : AnsiChar; const Source : AnsiString) : AnsiChar;
26670:  Function IcsStrPLCopy104( Dest : PChar; const Source : String; MaxLen : Cardinal) : PChar;
26671:  Function IcsStrPLCopy105( Dest : AnsiChar; const Source : AnsiString; MaxLen : Cardinal) : AnsiChar;
26672:  Function
       IcsStrCompOrdinalW(Str1:WideChar;Str1Length:Int;Str2:WideChar;Str2Length:Int;IgnoreCase:Bool):Int;
26673:  //Func
       RtlCompareUnicodeString(String1:PUNICODE_STRING;String2:PUNICODE_STRING;CaseInsensitive:BOOL):LongInt');
26674:  Function icsIsDebuggerPresent : BOOL');
26675:   SIRegister_TIcsIntegerList(CL);
26676:   SIRegister_TIcsCriticalSection(CL);
26677: end;
26678:
26679: procedure SIRegister_TCustomApplication(CL: TPSPascalCompiler);
26680: begin
26681:   //with RegClassS(CL,'TComponent', 'TCustomApplication') do
26682:   with CL.AddClassN(CL.FindClass('TComponent'),'TCustomApplication') do begin
26683:     Constructor Create( AOwner : TComponent);
26684:     Procedure HandleException( Sender : TObject);
26685:     Procedure Initialize');
26686:     Procedure Run');
26687:     Procedure ShowException( E : Exception);
26688:     Procedure Terminate');
26689:     Procedure Terminate1( AExitCode : Integer);
26690:     Function FindOptionIndex( const S : String; var Longopt : Boolean; StartAt : Integer) : Integer');
26691:     Function GetOptionValue( const S : String) : String');
26692:     Function GetOptionValue1( const C : Char; const S : String) : String');
26693:     Function GetOptionValues( const C : Char; const S : String) : TStringDynArray');
26694:     Function HasOption( const S : String) : Boolean');
26695:     Function HasOption1( const C : Char; const S : String) : Boolean');
26696:     Function CheckOptions(const ShortOptions:String;const Longopts:TStrings;Opts,
       NonOpts:TStrings;AllErrors: Bool):String;
26697:     Function CheckOptions1(const ShortOptions:String;const Longopts:array of string;Opts,NonOpts:TStrings;
       AllErrors:Bool):String;
26698:     Function CheckOptions2(const ShortOptions:String;const Longopts:TStrings; AllErrors:Boolean):String');
26699:     Function CheckOptions3(const ShortOptions:String;const LongOpts:array of
       string;AllErrors:Boolean):String;
26700:     Function CheckOptions4( const ShortOptions:String;const LongOpts:String; AllErrors:Boolean):String');
26701:     Function GetNonOptions(const ShortOptions: String; const Longopts:array of string) : TStringDynArray');
26702:     Procedure GetNonOptions1(const ShortOptions:String;const Longopts:array of string;NonOptions:TStrings);
26703:     Procedure GetEnvironmentList( List : TStrings; NamesOnly : Boolean);
26704:     Procedure GetEnvironmentList1( List : TStrings);
26705:     Procedure Log(EventType : TEventType; const Msg : String);
26706:     Procedure Log1(EventType : TEventType; const Fmt : String; const Args : array of string);
26707:     ExeName', 'string', iptr);
26708:     Terminated', 'Boolean', iptr);
26709:     Title', 'string', iptrw);
26710:     OnException', 'TExceptionEvent', iptrw);
26711:     ConsoleApplication', 'Boolean', iptr);
26712:     Location', 'String', iptr);
26713:     Params', 'String integer', iptr);
26714:     ParamCount', 'Integer', iptr);
26715:     EnvironmentVariable', 'String String', iptr);
26716:     OptionChar', 'Char', iptrw);
26717:     CaseSensitiveOptions', 'Boolean', iptrw);
26718:     StopOnException', 'Boolean', iptrw);
26719:     ExceptionExitCode', 'Longint', iptrw);
26720:     ExceptObject', 'Exception', iptrw);
26721:     ExceptObjectJS', 'JSValue', iptrw);
26722:     EventLogFilter', 'TEventLogTypes', iptrw);
26723:   end;
26724: end;
26725:
26726:
26727: ADD 4.7.6.10 III
26728: CL.AddDelphiFunction('function list_modules(aexename: string): string;
26729: CL.AddDelphiFunction('function list_units(aexename: string): string;
26730: check //ListView_SetItemState(H, I, Data, LVIS_SELECTED); commctrl
26731:
26732: procedure GetResourceHeader(var ResourceHeader: String; const FormName: String;
26733:                                                          const FileSize: Integer);
26734: GetResourceFormFile(List: TStrings; const FormName, FileName: String) : Boolean;
26735: function GetImageBase(const FileName: TFileName): DWORD;
26736: procedure ListViewSelectAll(ListView: TListView; Deselect: Boolean);
26737: function ListView_SetItemState(hwndLV: HWND; i: Integer; data, mask: UINT): Bool;
26738: function ListView_GetItemState(hwndLV: HWND; i, mask: Integer): Integer;
```

```
26739: function CreateOrGetOleObject(const ClassName: string): IDispatch;
26740: function FmtStrToInt(S: string): Integer;
26741: function MakeCellStrLV(const Text: String; Index: Integer; listview: TListview): String;
26742: https://docwiki.embarcadero.com/Libraries/Sydney/en/System.SysUtils.EmptyStr
26743: const emptystring= '';
26744:
26745: procedure SIRegister_TIdHashMessageDigest5(CL: TPSPascalCompiler);
26746: begin
26747:   //with RegClassS(CL,'TIdHashMessageDigest4', 'TIdHashMessageDigest5') do
26748:   with CL.AddClassN(CL.FindClass('TIdHashMessageDigest4'),'TIdHashMessageDigest5') do begin
26749:     Constructor Create;
26750:     Procedure Free;
26751:     Function HashValue(const ASrc: string): T4x4LongWordRecord');
26752:     Function HashValue1(AStream : TStream): T4x4LongWordRecord');
26753:     Function HashValueString(const ASrc: string): T4x4LongWordRecord');
26754:     Function HashValueStream(AStream : TStream): T4x4LongWordRecord');
26755:     Function AsHex( const AValue : T4x4LongWordRecord) : string');
26756:     function hashStreamAsHex(s: TStream): string;
26757:    end;
26758: end;
26759:
26760: procedure SIRegister_SeSHA256(CL: TPSPascalCompiler);
26761: begin
26762:  CL.AddConstantN('seHEADERSIZE','LongInt').SetInt( 80);
26763:  CL.AddConstantN('emptystring','string').setstring('');
26764:  CL.AddTypeS('T32se', 'array [0 .. 31] of byte;
26765:  CL.AddTypeS('THeaderse', 'array [0 .. seHEADERSIZE - 1] of byte;
26766:  //T32 = array [0 .. 31] of byte;
26767:   // To store the block header
26768:  // THeader = array [0 .. HEADERSIZE - 1] of byte;
26769:  // CL.AddTypeS('PSHA256HASH', '^TSHA256HASH // will not work');
26770:  Function CalcSHA256String( Msg : AnsiString) : TSHA256HASH;
26771:  Function CalcSHA256Stream( Stream : TStream) : TSHA256HASH;
26772:  Function seSHA256ToStr( Hash : TSHA256HASH) : String');
26773:  Function SHA256ToBinaryStr( Hash : TSHA256HASH) : AnsiString');
26774:  Function ReverseHash( Hash : string) : string');
26775:  Function GetMemoryStream : TMemoryStream');
26776:  Function T32ToString( const at32 : T32se) : string');
26777:  Function CalcHeaderSHA256( aHeader : THeaderse) : TSHA256HASH;
26778: end;
26779:
26780: procedure SIRegister_BlocksUnit(CL: TPSPascalCompiler);
26781: begin
26782:   CL.AddTypeS('TCrypto', '( tcBitcoin, tcPascalCoin );
26783:   CL.AddTypeS('TNetbc', '( tnMainNet, tnTestNet );
26784:   CL.AddTypeS('PByte', 'Byte');
26785:   SIRegister_TBlockFile(CL);
26786:   CL.AddTypeS('TBlockHeader', record versionNumber : UInt32; aPreviousBlockHas'
26787:    +h: T32se; aMerkleRoot: T32se; time: UInt32; DifficultyTarget : UInt32; nonce: UInt32; end');
26788:   SIRegister_TInputbc(CL);
26789:   SIRegister_TOutputbc(CL);
26790:   SIRegister_TInputsbc(CL);
26791:   SIRegister_TOutputsbc(CL);
26792:   SIRegister_TTransaction(CL);
26793:   SIRegister_TBlockTransactions(CL);
26794:   SIRegister_TBlockRecord(CL);
26795:   CL.AddTypeS(TStartFileBlockFoundNotify', 'Procedure ( const aBlockFiles: tstringlist);
26796:   TFoundFileBlockNotify', 'Procedure ( const aBlockFile : TBlockFi'
26797:    +le; const actualFileBlockNumber, totalBlockFiles : integer; var next : boolean);
26798:   TEndFilesBlockFoundNotify', 'Procedure ( const aBlockFiles : tstringlist);
26799:   TFoundBlockNotify','Procedure(const aBlock: TBlockRecord; var findnext: boolean);
26800:   TBlockProcessStepNotify', 'Procedure ( const aPos, asize : int64);
26801:   TEndProcessBlockFile', 'Procedure ( const aBlockFile : TBlockFile);
26802:   SIRegister_TBlocks(CL);
26803: end;
26804:
26805: procedure SIRegister_DelticsCommandLine(CL: TPSPascalCompiler);
26806: begin
26807:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCommandLine');
26808:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCommandLineSwitch');
26809:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidCommandLine');
26810:   //CL.AddTypeS('PSwitchDef', '^TSwitchDef // will not work');
26811:   CL.AddTypeS('TSwitchDef', 'record Name : String; Switch : String; Default : S'
26812:    +'tring; MinValues : Integer; MaxValues : Integer; Separator : Char; end');
26813:   SIRegister_TCommandLine(CL);
26814:   SIRegister_TCommandLineSwitch(CL);
26815:  Procedure CommandLineToArgs( const aString : String; const aArgs : TStrings);
26816: end;
26817:
26818: (*----------------------------------------------------------------------------*)
26819: procedure SIRegister_DelticsStrUtils(CL: TPSPascalCompiler);
26820: begin
26821:  CL.AddConstantN('LIKENESS_MATCH','Char').SetString( #255);
26822:   CL.AddTypeS('TANSIStringArray', 'array of ANSIString');
26823:   //CL.AddTypeS('TStringArray', 'array of String');
26824:   SIRegister_IStringList(CL);
26825:   SIRegister_IStringTool(CL);
26826:   SIRegister_TStringTool(CL);
26827:  Procedure dArrayInit( var aArray : TANSIStringArray; const aSize : Integer);
```

```
26828:  Procedure ArrayAdd1( var aArray : TANSIStringArray; const aValue : ANSIString);
26829:  Procedure ArrayAdd2( var aArray : TANSIStringArray; const aValue : array of ANSIString);
26830:  Procedure ArrayExtend( var aArray : TANSIStringArray; const aCount : Integer);
26831:  Function Likeness( const A : ANSIString; const B : ANSIString) : Integer');
26832:  Function LikenessEx(const A:ANSIString;const B:ANSIString; var strA:ANSIString;var
        strB:ANSIString):Integer;
26833:  Function LikenessEx2(const A:ANSIString;const B:ANSIString;var strA:ANSIString;var strB:ANSIString;var
        iSame :
26834:                          Int;var iTotal:Int):Int;
26835:  Function CamelCapsToWords( const aString : String) : String');
26836:  Function ReverseStr( const aString : String) : String;
26837:  Function ReverseStr2( const aString : ANSIString) : ANSIString;
26838:  Function dSplit( const aString : String; const aDelim : Char; const aSlices : TStrings) : Integer;
26839:  Function dSplit1( const aString : String; const aDelim : Char; var aSlices : TStringArray) : Integer;
26840:  Procedure dSplit2( const aString : ANSIString; const aDelim : ANSIChar; var aSlices : TANSIStringArray);
26841:  Function StrBeginsWith( const aString : String; const aBeginning : String) : Boolean');
26842:  Function StrBeginsWithText( const aString : String; const aBeginning : String) : Boolean');
26843:  Function dStrContains( const aString : String; const aSubString : String) : Boolean');
26844:  Function StrContainsText( const aString : String; const aSubString : String) : Boolean');
26845:  Function StrEndsWith( const aString : String; const aEnd : String) : Boolean');
26846:  Function StrEndsWithText( const aString : String; const aEnd : String) : Boolean');
26847:  Function StrLPad( const aString : String; const aWidth : Integer; const aTrunc : Boolean) : String');
26848:  Function StrRPad( const aString : String; const aWidth : Integer; const aTrunc : Boolean) : String');
26849:  Function StrDequote( const aString : String) : String');
26850:  Function StrPop( var aString : String; const aDelim : Char) : String');
26851:  Function StrPopQuoted( var aString : String; const aDelim : Char) : String');
26852:  Function dIsAlpha( const aChar : Char) : Boolean');
26853:  Function dIsDigit( const aChar : Char) : Boolean');
26854:  Function dIsLower( const aChar : Char) : Boolean');
26855:  Function dIsUpper( const aChar : Char) : Boolean');
26856:   //SIRegister_TManagedStringList(CL);
26857: function delticsExplode(const Separator, S: string; Limit: Integer = 0): TStringArray;
26858: function delticsImplode(const Glue: string; const Pieces: array of string): string;
26859: end;
26860:
26861: (*----------------------------------------------------------------------------*)
26862: procedure SIRegister_DelticsSysUtils(CL: TPSPascalCompiler);
26863: begin
26864:  CL.AddConstantN('delticsEmptyStr','String').SetString( '');
26865:  //CL.AddConstantN('NullGUID','TGUID').SetString( '{00000000-0000-0000-0000-000000000000}');
26866:   SIRegister_IAutoFree(CL);
26867:   //CL.AddTypeS('Exception', 'Exception');
26868:   SIRegister_ENotImplemented(CL);
26869:   CL.AddClassN(CL.FindClass('TOBJECT'),'EAccessDenied');
26870:   CL.AddTypeS('TGUIDFormat', '( gfDefault, gfNoBraces, gfNoHyphens, gfDigitsOnly );
26871:   CL.AddTypeS('TRoundingStrategy', '( rsDefault, rsAwayFromZero, rsTowardsZero );
26872:   CL.AddTypeS('TriBoolean', '( tbUnknown, tbTRUE, tbFALSE );
26873:  Procedure deltCloneList( const aSource : TStringList; const aDest : TStringList);
26874:  //CL.AddDelphiFunction('Function AutoFree( const aReference : PObject) : IUnknown;
26875:  //('Function AutoFree5( const aReferences : array of PObject) : IUnknown;
26876:  Procedure deltFreeAndNIL( var aObject: Tstrings);
26877:  //Procedure NILRefs( const aObjects : array of PObject);
26878:  Function deltIfThen7( aValue : Boolean; aTrue, aFalse : Boolean) : Boolean;
26879:  Function deltIfThen8( aValue : Boolean; aTrue, aFalse : TObject) : TObject;
26880:  Function deltIfThen9( aValue : Boolean; aTrue, aFalse : Integer) : Integer;
26881:  Function deltIfThen10( aValue : Boolean; aTrue, aFalse : ANSIString) : ANSIString;
26882:  Function deltIfThen11( aValue : Boolean; aTrue, aFalse : String) : String;
26883:  Function deltIfThenInt( aValue : Boolean; aTrue, aFalse : Integer) : Integer');
26884:  Function deltStringIndex( const aString : String; const aCases : array of String : Integer');
26885:  Function deltTextIndex( const aString : String; const aCases : array of String): Integer');
26886:  Function deltMin64( ValueA, ValueB : Int64) : Int64');
26887:  Function deltMin12( ValueA, ValueB : Cardinal) : Cardinal;
26888:  Function deltMin13( ValueA, ValueB : Integer) : Integer;
26889:  Function deltMax64( ValueA, ValueB : Int64) : Int64');
26890:  Function deltMax14( ValueA, ValueB : Cardinal) : Cardinal;
26891:  Function deltMax15( ValueA, ValueB : Integer) : Integer;
26892:  Procedure deltExchange( var A, B, aSize : LongWord);
26893:  Function deltGUIDToString( const aGUID : TGUID; const aFormat : TGUIDFormat) : String');
26894:  Function deltStringToGUID( const aString : String) : TGUID');
26895:  Function deltTryStringToGUID( const aString : String; var aGUID : TGUID) : Boolean');
26896:  Function deltIsNull( const aValue : TGUID) : Boolean');
26897:  Function deltNewGUID : TGUID');
26898:  Function deltSameGUID( const GUIDA, GUIDB : TGUID) : Boolean');
26899:  Procedure deltAddTrailingBackslash( var aString : String);
26900:  Procedure deltRemoveTrailingBackslash( var aString : String);
26901:  Function deltBinToHex( const aBuf : string; const aSize : Integer) : String');
26902:  Function deltReverseBytes16( const aValue : Word) : Word;
26903:  Function deltReverseBytes17( const aValue : LongWord) : LongWord;
26904:  Function deltReverseBytes18( const aValue : Int64) : Int64;
26905:  Function deltRound( const aValue : Extended; const aStrategy : TRoundingStrategy) : Integer');
26906:  //Procedure ForEachComponent(const aComponent:TComponent; const aProc:TComponentProc;const
        aRecursive:Bool; const aClass:TComponentClass);
26907:  Function deltExec( const aEXE : String; const aCommandLine : String) : Cardinal');
26908:  Procedure deltExecAndWait( const aEXE : String; const aCommandLine : String);
26909:  Function deltFindProcess( const aEXEName : String; var aProcess : TProcessEntry32): Boolean');
26910:  Function deltProcessExists( const aEXEName : String) : Boolean');
26911:  Function deltRegisterDLL( const aFileName : String; const aRegistrationProc:String):Boolean');
26912:  Function deltIsTRUE( const aTri : TriBoolean) : Boolean');
26913:  Function deltIsFALSE( const aTri : TriBoolean) : Boolean');
```

```
26914:  Function deltIsKnown( const aTri : TriBoolean) : Boolean');
26915: end;
26916:
26917: procedure SIRegister_U_Splines(CL: TPSPascalCompiler);
26918: begin
26919:  CL.AddConstantN('MaxNoVertices','LongInt').SetInt( 100);
26920:   CL.AddTypeS('TSplineVertex', 'record X : Single; Y : single; end');
26921:   CL.AddTypeS('TSplineRow', 'array of TSplineVertex');
26922:   SIRegister_TBSpline(CL);
26923:   SIRegister_TSplines(CL);
26924:  //CL.AddDelphiFunction('Procedure Register');
26925:  SIRegister_U_SpringMass2(CL);
26926:  CL.AddDelphiFunction('function getPython: string;
26927:  CL.AddDelphiFunction('function getPython2: string;
26928:  CL.AddDelphiFunction('function getPython3: string;
26929: end;
26930:
26931: procedure SIRegister_MARSCoreUtils(CL: TPSPascalCompiler);
26932: begin
26933:  Function CreateCompactGuidStr : string');
26934:  //CL.AddDelphiFunction('Function BooleanToTJSON( AValue : Boolean) : TJSONValue');
26935:  Function SmartConcat(const AArgs:array of string;const ADelimiter:string;const AAvoidDuplicateDelimiter:
       Boolean;const ATrim:Boolean; const ACaseInsensitive:Boolean):string');
26936:  Function EnsurePrefix2( const AString, APrefix : string; const AIgnoreCase : Boolean) : string');
26937:  Function EnsureSuffix( const AString, ASuffix : string; const AIgnoreCase : Boolean) : string');
26938:  Function StringArrayToString( const AArray : array of string; const ADelimiter : string) : string');
26939:  Procedure CopyStream2(ASourceStream,
       ADestStream:TStream;AOverWriteDest:Bool;AThenResetDestPosition:Boolean);
26940:  Function DateToISO8601( const ADate : TDateTime; AInputIsUTC : Boolean) : string');
26941:  Function ISO8601ToDate( const AISODate : string; AReturnUTC : Boolean) : TDateTime');
26942:  Function IsMask( const AString : string) : Boolean');
26943:  Function MatchesMask2( const AString, AMask : string) : Boolean');
26944:  Function StreamToBase64( const AStream : TStream) : string');
26945:  Procedure Base64ToStream( const ABase64 : string; const ADestStream : TStream);
26946:  Function StreamToBytes( const ASource : TStream) : TBytes');
26947:  //Function GetEncodingName( const AEncoding : TEncoding) : string');
26948:  function StringFallback(const AStrings: array of string; const ADefault: string): string;
26949:  function StripPrefix(const APrefix, AString: string): string;
26950: function StripSuffix(const ASuffix, AString: string): string;
26951: end;
26952:
26953: procedure SIRegister_U_CoasterB(CL: TPSPascalCompiler);
26954: begin
26955:   //CL.AddTypeS('float', 'single');
26956:   CL.AddTypeS(tcoasterRealPoint', 'record x : single; y : single; end');
26957:   TTrackPoint', 'record x : single; y : single; angleright : single; '
26958:    +sinangleright : single; cosangleright : single; quadrantright : integer; len'
26959:    +gthright : single; radiusright : single; end');
26960:    TTrackPointArray', 'array of TTrackPoint'); //carts', 'array of TCartLocRec'
26961:    TTrackRec', 'record x : single; y : single; index : integer; distT'
26962:    +oend : single; quadrant : integer; radius: single; angle: single; sinangle '
26963:    +: single; cosangle : single; end');
26964:   TCartLocRec', 'record p1 : tpoint; p2: tpoint; p3: tpoint; p4: tpoint; wr: integer; end');
26965:    TCartLocRecArray', 'array of TCartLocRec');
26966:   TLoopType', '( Insideloop, Outsideloop, none );
26967:   TTemplateMode', '( normal, makeline1, makeline2, makecircle1, makecircle2, movecircle );
26968:   CL.AddTypeS(ARRsoundstr', 'array [0..5] of string');
26969:    //type ARRstr = array [0..5] of string;
26970:   SIRegister_TcoasterLine(CL);
26971:   SIRegister_TcoasterCircle(CL);
26972:   SIRegister_TCoaster(CL);
26973: end;
26974:
26975: procedure SIRegister_TSpring(CL: TPSPascalCompiler);
26976: begin
26977:   //with RegClassS(CL,'tPanel', 'TSpring') do
26978:   with CL.AddClassN(CL.FindClass('tPanel'),'TSpring') do  begin
26979:     K', 'single', iptrw);  M', 'single', iptrw);
26980:     G', 'single', iptrw);  A', 'single', iptrw);
26981:     x', 'single', iptrw);  v', 'single', iptrw);
26982:     C', 'single', iptrw);  X0', 'single', iptrw);
26983:     V0', 'single', iptrw); XEnd', 'single', iptrw);
26984:     InitiallyConstrained', 'Boolean', iptrw);
26985:     Timeinc', 'single', iptrw);
26986:     Scale', 'single', iptrw);
26987:     Stype', 'integer', iptrw);
26988:     delay', 'integer', iptrw);
26989:     pts', 'Tspringpointarray', iptrw);
26990:     step', 'integer', iptrw);
26991:     wraps', 'integer', iptrw);
26992:     drawColor', 'TColor', iptrw);
26993:     Constructor create( Aowner : TPanel; r : Trect);
26994:     Procedure drawspring( Len : single);
26995:     Procedure erasespring');
26996:     Procedure redrawinitial');
26997:     Function Setup(newV0,newX0,newC,newK,newM,newG,newTimeInc:single;newconstraint:bool;
26998:                          Nbrloops,newStep, newdelay: integer;newcolor:TColor):bool');
26999:     Procedure animate(form1: TForm; memo2: Tmemo);
27000:     Procedure animate2(form1: TForm; memo2: Tmemo; detail: boolean);
```

```
27001:     //procedure animate2(form1: TForm; memo2: Tmemo; detail: boolean);
27002:     Procedure sizechanged( sender : TObject);
27003:     Function GetMaxAmp : single');
27004:   end;
27005: end;
27006:
27007: procedure SIRegister_clJsonParser(CL: TPSPascalCompiler);
27008: begin
27009:   SIRegister_EclJSONError(CL);
27010:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONString');
27011:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONPair');
27012:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONObject');
27013:   CL.AddClassN(CL.FindClass('TOBJECT'),'TclJSONArray');
27014:   SIRegister_TclJSONBase(CL);
27015:   SIRegister_TclJSONPair(CL);
27016:   SIRegister_TclJSONValue(CL);
27017:   SIRegister_TclJSONString(CL);
27018:   SIRegister_TclJSONBoolean(CL);
27019:   SIRegister_TclJSONArray(CL);
27020:   SIRegister_TclJSONObject(CL);
27021:  CL.AddConstantN('cUnexpectedDataEndCode','LongInt').SetInt( - 100);
27022:  'cUnexpectedDataSymbolCode','LongInt').SetInt( - 101);
27023:  'cInvalidControlSymbolCode','LongInt').SetInt( - 102);
27024:  'cInvalidUnicodeEscSequenceCode','LongInt').SetInt( - 103);
27025:  'cUnrecognizedEscSequenceCode','LongInt').SetInt( - 104);
27026:  'cUnexpectedDataTypeCode','LongInt').SetInt( - 106);
27027:  function GetConnectionKind2(var strKind: string): Boolean;
27028: end;
27029:
27030: procedure SIRegister_SynHighlighterPython(CL: TPSPascalCompiler);
27031: begin
27032:   CL.AddTypeS(TtkTokenKindpy', '( tkComment, tkIdentifier, tkKey, tkNull, tkNumb'
27033:    +er, tkSpace, tkString, tkSymbol, tkNonKeyword, tkTrippleQuotedString, tkSy'
27034:    +stemDefined, tkHex, tkOct, tkFloat, tkUnknown );
27035:   CL.AddTypeS('TRangeStatepy', '( rsANil,rsComment,rsUnKnown,rsMultilineString'
27036:    +', rsMultilineString2, rsMultilineString3 );
27037:   SIRegister_TSynPythonSyn(CL);
27038: end;
27039:
27040: procedure SIRegister_DudsCommonDelphi(CL: TPSPascalCompiler);
27041: begin
27042:  Function DelphiIDERunning : Boolean');
27043:  Function GetDelphiVersionRegistryKey( Version : Integer) : String');
27044:  Function GetDelphiRootDirectory( Version : Integer) : String');
27045:  Function IsDelphiVersionInstalled( Version : Integer) : Boolean');
27046:  Function GetDelphiEnvironmentVariables(Version:Int;const EnvironmentVariables:TStrings):Bool;
27047:  Function GetDelphiLibraryPaths(Version:Int;const
       LibraryPaths:TStrings;EvaluateMacros:Bool;CheckPaths:Bool):Bool;
27048:  function StrConcat(ArrS:array of string;Space:string):string;
27049:  function StrDelSpc(s:string):string;
27050:  function StrSplit2(S:string;Ch:char):TStringDynArray;
27051:  procedure StrReplaceFirst(var S:string;olds,news:string);
27052:  procedure StrReplaceLast(var S:string;OldS,NewS:string);
27053: procedure ConvertV3ToS(X,Y,Z:extended;var r,a,b:extended);
27054: procedure ConvertSToV3(R,A,B:extended;var X,Y,Z:extended);
27055: procedure InterpolateColor(Color0,Color1:TColor;Count:integer;var TempColor:array of TColor);
27056: function TryStrToextended(const S: string; out Value: extended): Boolean;
27057: function isAVXSupported: Boolean;
27058: function GetEXEVersionData(const FileName: string): TEXEVersionData;
27059: function txtsearch(pat: string; text: string): integer;
27060: procedure GetRegisteredAntiviruses(ProgIDs: TStrings);
27061: type
27062:   TEXEVersionData = record
27063:     CompanyName,
27064:     FileDescription, FileVersion,
27065:     InternalName,
27066:     LegalCopyright, LegalTrademarks,
27067:     OriginalFileName, ProductName,
27068:     ProductVersion, Comments,
27069:     PrivateBuild, SpecialBuild: string;
27070:   end;
27071: type
27072: TNeuronClass = class(TObject)
27073:   private
27074:     FThreshold: Single;
27075:     FInputCount: Integer;
27076:     FInputs:TSingleDynArray;
27077:     FWeights:TSingleDynArray;
27078:     FAnalogOutput: Boolean;
27079:     function GetInput(Index: Integer): Single;
27080:     function GetOutput: Single;
27081:     function GetWeight(Index: Integer): Single;
27082:     procedure SetInput(Index: Integer; const Value: Single);
27083:     procedure SetWeight(Index: Integer; const Value: Single);
27084:     procedure SetInputCount(const Value: Integer);
27085:     function GetLoad: Single;
27086:   public
27087:     Constructor Create(aInputCount:Integer);
27088:     procedure RandomInitialize;
```

```
27089:     Property Load:Single read GetLoad;
27090:     property Output:Single read GetOutput;
27091:     property Threshold:Single read FThreshold write FThreshold;
27092:     property Inputs[Index: Integer]: Single read GetInput write SetInput;
27093:     property Weights[Index: Integer]: Single read GetWeight write SetWeight;
27094:     property InputCount:Integer read FInputCount write SetInputCount;
27095:     property AnalogOutput:Boolean read FAnalogOutput write FAnalogOutput;
27096: end;
27097: TPJConsoleApp:= class(TPJCustomConsoleApp)  Console Application Runner CAR
27098:   public
27099:     // Make all inherited protected properties public
27100:     property StdIn;
27101:     property StdOut;
27102:     property StdErr;
27103:     property CommandLine;
27104:     property CurrentDir;
27105:     property Visible;
27106:     property MaxExecTime;
27107:     property TimeSlice;
27108:     property KillTimedOutProcess;
27109:     property ProcessAttrs;
27110:     property ThreadAttrs;
27111:     property UseNewConsole;
27112:     property ConsoleTitle;
27113:     property ConsoleColors;
27114:     property ScreenBufferSize;
27115:     property WindowPosition;
27116:     property WindowSize;
27117:     property Environment;
27118:     property UnicodeEnvironment;
27119:     property Priority;
27120:     property TimeToLive;
27121:     property ElapsedTime;
27122:     property ProcessInfo;
27123:     property ExitCode;
27124:     property ErrorCode;
27125:     property ErrorMessage;
27126:     property OnStart;
27127:     property OnWork;
27128:     property OnComplete;
27129:   end;
27130:
27131: procedure SIRegister_uHTMLBuilder(CL: TPSPascalCompiler);
27132: begin
27133:   SIRegister_THTMLBase(CL);
27134:   SIRegister_THTMLCell(CL);
27135:   CL.AddTypeS(TAfterAddRow', 'Procedure (Table:THTMLBase; DataSet: TDataSet);
27136:   SIRegister_THTMLItem(CL);
27137:   SIRegister_THTMLParagraph(CL);
27138:   SIRegister_THTMLRow(CL);
27139:   SIRegister_THTMLTable(CL);
27140:   SIRegister_THTMLReport(CL);
27141:   SIRegister_THTMLBuild(CL);
27142: end;
27143: https://github.com/guitorres/htmlbuilder/blob/master/src/uHTMLBuilder.pas
27144:
27145: procedure SIRegister_RestJsonUtils(CL: TPSPascalCompiler);
27146: begin
27147:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidValue');
27148:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidValueForField');
27149:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidSyntax');
27150:   SIRegister_ENoSerializableClass(CL);
27151:   SIRegister_TJsonUtil(CL);
27152:  Function JavaToDelphiDateTime2( const dt : int64) : TDateTime');
27153:  Function DelphiToJavaDateTime2( const dt : TDateTime) : int64');
27154:  Function ISO8601DateToJavaDateTime( const str : String; var ms : Int64) : Boolean');
27155:  Function ISO8601DateToDelphiDateTime( const str : string; var dt : TDateTime) : Boolean');
27156:  Function DelphiDateTimeToISO8601Date2( dt : TDateTime) : string');
27157: end;
27158:
27159: procedure SIRegister_pxQRcode(CL: TPSPascalCompiler);
27160: begin
27161:  CL.AddConstantN('QR_ECLEVEL_L','LongInt').SetInt( 0);
27162:  ('QR_ECLEVEL_M','LongInt').SetInt( 1);
27163:  ('QR_ECLEVEL_Q','LongInt').SetInt( 2);
27164:  ('QR_ECLEVEL_H','LongInt').SetInt( 3);
27165:  Procedure CreateQRCodeBMP( const AText:string;const ABitmapStream:TStream;const ALevel:Byte;const
       ASize:Int);
27166: end;
27167:
27168: procedure SIRegister_DelphiZXingQRCode(CL: TPSPascalCompiler);
27169: begin
27170:   TQRCodeEncoding','( qrcAuto,qrcNumeric,qrcAlphanumeric,qrcISO88591,qrcUTF8NoBOM,qrcUTF8BOM );
27171:   CL.AddTypeS(T2DBooleanArray', 'array of array of Boolean');
27172:   SIRegister_TDelphiZXingQRCode(CL);
27173: end;
27174:
27175: function DownloadArchive( sURL, sArchivoLocal: String ): boolean;
27176: function GetUrlContent(const Url: string): UTF8String;
```

```
procedure FindFiles2List(StartDir,FileMask: string; recursively:boolean; var FilesList: TStringList);
unit UtilsTimeCode;
function TCtoFrames(S:String):Integer;
function FramestoTC(Fi:Integer):String;
function DFtoFrames(S:String):Integer;
function FramestoDF(Fi:Integer):String;
function ValidDropFrame(S:String):Boolean;

unit uPSI_KLibUtils;
procedure SIRegister_KLibUtils(CL: TPSPascalCompiler);
begin
 CL.AddTypeS('TResourceklib', 'record name : string; _type : string; end');
 CL.AddTypeS('TAnonymousMethodklib',{reference to} 'procedure');
 CL.AddTypeS('TCallBackklib', 'procedure(msg: string);
 Procedure deleteFilesInDirklib( pathDir : string; const filesToKeep : array of string);
 Procedure deleteFilesInDirWithStartingFileNameklib(dirName:string;
startingFileName:string;fileType:string);
 Function checkIfFileExistsAndEmptyklib( fileName : string) : boolean');
 Procedure deleteFileIfExistsklib( fileName : string);
 Function getTextFromFileklib( fileName : string) : string');
 Function checkIfThereIsSpaceAvailableOnDriveklib( drive : char; requiredSpaceInBytes:int64): boolean');
 Function getFreeSpaceOnDriveklib( drive : char) : int64');
 Function getIndexOfDriveklib( drive : char) : integer');
 Function getDriveExeklib : char');
 Function getDirSizeklib( path : string) : int64');
 Function getCombinedPathWithCurrentDirklib( pathToCombine : string) : string');
 Function getDirExeklib : string');
 Procedure createDirIfNotExistsklib( dirName : string);
 Function checkIfIsLinuxSubDirklib( subDir : string; mainDir : string) : boolean');
 Function getPathInLinuxStyleklib( path : string) : string');
 Function checkIfIsSubDirklib(subDir : string; mainDir: string; trailingPathDelimiter:char):boolean');
 Function getValidFullPathklib( fileName : string) : string');
 Function checkMD5Fileklib( fileName : string; MD5 : string) : boolean');
 Procedure unzipResourceklib( nameResource : string; destinationDir : string);
 //Function getPNGResource( nameResource : string) : TPngImage');
 Procedure getResourceAsEXEFileklib( nameResource : string; destinationFileName: string);
 Procedure getResourceAsZIPFileklib( nameResource : string; destinationFileName: string);
 Procedure getResourceAsFileklib( resource : TResourceklib; destinationFileName: string);
 Function getResourceAsString( resource : TResourceklib) : string');
 Function getResourceAsStringklib( resource : TResourceklib) : string');
 Function getResourceAsStreamklib( resource : TResourceklib) : TResourceStream');
 Procedure unzipklib( zipFileName : string; destinationDir : string; deleteZipAfterUnzip : boolean);
 //Function checkRequiredFTPProperties( FTPCredentials : TFTPCredentials) : boolean');
 Function getValidItalianTelephoneNumberklib( number : string) : string');
 Function getValidTelephoneNumberklib( number : string) : string');
 Function getRandStringklib( size : integer) : string');
 Function getFirstFileNameInDirklib( dirName : string; fileType: string; fullPath: boolean): string');
 Function getFileNamesListInDirklib( dirName : string; fileType: string; fullPath: boolean): TStringList');
 Procedure saveToFileklib( source : string; fileName : string);
 Function getCombinedPathklib( path1 : string; path2 : string) : string');
 Function getCurrentDayOfWeekAsStringklib : string');
 Function getDayOfWeekAsStringklib( date : TDateTime) : string');
 Function getCurrentDateTimeAsStringklib : string');
 Function getDateTimeAsStringklib( date : TDateTime) : string');
 Function getCurrentDateAsStringklib : string');
 Function getDateAsStringklib( date : TDateTime) : string');
 Function getCurrentTimeStampklib : string');
 Function getCurrentDateTimeAsStringWithFormattingklib( formatting : string) : string');
 Function getDateTimeAsStringWithFormattingklib( value : TDateTime; formatting: string): string');
 Function getCurrentDateTimeklib : TDateTime');
 Function getParsedXMLstringklib( mainString : string) : string');
 Function getDoubleQuotedStringklib( mainString : string): string');
 Function getSingleQuotedStringklib( mainString : string): string');
 Function
getMainStringWithSubStringInsertedklib(mainString:string;insertedString:string;index:integer):string;
 Function getStringWithoutLineBreaksklib( mainString : string; substituteString : string) : string');
 Function getCSVFieldFromStringAsDateklib( mainString :string; index : integer; delimiter : Char): TDate;
 Function getCSVFieldFromStringAsDate1klib(mainString:string;index:integer;formatSettings:TFormatSettings;
delimiter:Char):TDate;
 Function getCSVFieldFromStringAsDoubleklib( mainString:string; index:integer;delimiter:Char):Double;
 Function
getCSVFieldFromStringAsDouble2klib(mainString:string;index:integer;formatSettings:TFormatSettings;
delimiter:Char):Double;
 Function getCSVFieldFromStringAsIntegerklib(mainString:string;index:integer;delimiter:Char):integer');
 Function getCSVFieldFromStringklib( mainString : string; index : integer; delimiter : Char): string');
 Function getNumberOfLinesInStrFixedWordWrapklib( source : string) : integer');
 Function stringToStrFixedWordWrapklib( source : string; fixedLen : Integer) : string');
 Function stringToStringListWithFixedLenklib( source : string; fixedLen : Integer) : TStringList');
 Function stringToStringListWithDelimiterklib( value : string; delimiter : Char) : TStringList');
 Function stringToTStringListklib( source : string) : TStringList');
 Function arrayOfStringToTStringListklib( arrayOfStrings : array of string) : TStringList');
 Procedure splitStringsklib(source:str;delimiter:str;var destFirstString:str;var destSecondString: string);
 Procedure splitStrings2klib(source:str;delimiterPosition:int;var destFirstString:str;var
destSecondString:str);
 Function getMergedStringsklib( firstString : string; secondString : string; delimiter: string): string');
 Function checkIfEmailIsValidklib( email : string) : boolean');
 Function
checkIfMainStringContainsSubStringNoCaseSensitiveklib(mainString:string;subString:string):boolean;
 Function
checkIfMainStringContainsSubStringklib(mainString:str;subString:str;caseSensitiveSearch:bool):bool;
```

```
27259:  Function getDoubleAsStringklib( value : Double; decimalSeparator : char) : string');
27260:  Function getFloatToStrDecimalSeparatorklib : char');
27261:  Procedure tryToExecuteProcedureklib( myProcedure: TAnonymousMethodklib; raiseExceptionEnabled: boolean);
27262:  Procedure tryToExecuteProcedure2klib( myProcedure : TCallBackklib; raiseExceptionEnabled : boolean);
27263:  //Procedure tryToExecuteProcedure3( myProcedure : TProcedure; raiseExceptionEnabled : boolean);
27264:  Procedure executeProcedureklib( myProcedure : TAnonymousMethodklib);
27265:  Procedure executeProcedure2klib( myProcedure : TCallBackklib);
27266:  function setResourceHInstanceklib(aresource: longword): longword;
27267:  function setResourceHInstanceklib(aresource: longword): longword;
27268:  function executeAndWaitExe(fileName:string; params:string; exceptionIfReturnCodeIsNot0:boolean):LongInt;
27269: end;
27270:
27271: procedure SIRegister_KLibWindows(CL: TPSPascalCompiler);
27272: begin
27273:  //CL.AddDelphiFunction('Procedure downloadFile( info : TDownloadInfo; forceOverwrite : boolean);
27274:  Function getFirstPortAvaliableklib( defaultPort : integer; host : string) : integer');
27275:  Function checkIfPortIsAvaliableklib( host : string; port : Word) : boolean');
27276:  Function checkIfAddressIsLocalhostklib( address : string) : boolean');
27277:  Function getIPFromHostNameklib( hostName : string) : string');
27278:  Function getIPklib : string');
27279:  Function checkIfRunUnderWineklib : boolean');
27280:  Function checkIfWindowsArchitectureIsX64klib : boolean');
27281:   CL.AddTypeS('TPIDCredentials', 'record ownerUserName : string; domain : string; end');
27282:   CL.AddTypeS('TWindowsArchitectureklib', '( WindowsX86, WindowsX64 );
27283:  Function getWindowsArchitectureklib : TWindowsArchitectureklib');
27284:  Function checkIfUserIsAdminklib : boolean');
27285:  Function IsUserAnAdminklib : boolean');
27286:   CL.AddTypeS('TShowWindowType','( _SW_HIDE, _SW_SHOWNORMAL, _SW_NORMAL, _SW_S'
27287:    +'HOWMINIMIZED, _SW_SHOWMAXIMIZED, _SW_MAXIMIZE, _SW_SHOWNOACTIVATE, _SW_SHO'
27288:    +'W, _SW_MINIMIZE, _SW_SHOWMINNOACTIVE, _SW_SHOWNA, _SW_RESTORE, _SW_SHOWDEFAULT, _SW_FORCEMINIMIZE, _SW_MAX
);
27289:  Procedure openWebPageWithDefaultBrowserklib( url : string);
27290:  Function
shellExecuteOpenklib(fileName:string;params:string;directory:string;showWindowType:TShowWindowType;
exceptionIfFunctionFails:boolean):integer');
27291:  Function shellExecuteExeAsAdminklib(fileName:string;params:string;showWindowType : TShowWindowType;
exceptionIfFunctionFails:boolean):integer');
27292:  Function shellExecuteExeklib(fileName:string;params:string;showWindowType:TShowWindowType;
exceptionIfFunctionFails:boolean;operation: string):integer');
27293:  Function
myShellExecuteklib(handle:integer;operation:string;fileName:string;params:string;directory:string;
showWindowType:TShowWindowType;exceptionIfFunctionFails:boolean):integer');
27294:  Function shellExecuteExCMDAndWaitklib(params:string;runAsAdmin:boolean;showWindowType:TShowWindowType;
exceptionIfReturnCodeIsNot0:boolean):LongInt');
27295:  Function shellExecuteExAndWaitklib(fileName:string; params:string;runAsAdmin:boolean;showWindowType:
TShowWindowType;exceptionIfReturnCodeIsNot0:boolean) : LongInt');
27296:  Function executeAndWaitExeklib(fileName:string;params:string;exceptionIfReturnCodeIsNot0:bool):LongInt');
27297:  Function netShareklib( targetDir : string; netName : string; netPassw : string;
grantAllPermissionToEveryoneGroup : boolean) : string');
27298:  Procedure addTCP_IN_FirewallExceptionklib(ruleName:string;port:Word;description:string;grouping:string;
executable:string)
27299:  Procedure deleteFirewallExceptionklib( ruleName : string);
27300:  // CL.AddTypeS('TExplicitAccess', 'EXPLICIT_ACCESS_A');
27301:  Procedure grantAllPermissionsNetToTheObjectForTheEveryoneGroupklib( myObject : string);
27302:  Procedure grantAllPermissionsNetToTheObjectForTheUsersGroupklib( myObject : string);
27303:  Procedure grantAllPermissionNetToTheObjectklib( windowsGroupOrUser : string; myObject : string);
27304:  Procedure grantAllPermissionsToTheObjectForTheEveryoneGroupklib( myObject : string);
27305:  Procedure grantAllPermissionsToTheObjectForTheUsersGroupklib( myObject : string);
27306:  Procedure grantAllPermissionsToTheObjectklib( windowsGroupOrUser : string; myObject : string);
27307:  Procedure grantAllPermissionsToTheObjectForTheEveryoneGroup2klib( myObject : string);
27308:  Procedure grantAllPermissionsToTheObjectForTheUsersGroup2klib( myObject : string);
27309:  Procedure grantAllPermissionsToTheObject2klib( windowsGroupOrUser : string; myObject : string);
27310:  Function checkIfWindowsGroupOrUserExistsklib( windowsGroupOrUser : string) : boolean');
27311:  Procedure createDesktopLinkklib( fileName : string; nameDesktopLink : string; description: string);
27312:  Function getDesktopDirPathklib : string');
27313:  Procedure copyDirIntoTargetDirklib( sourceDir : string; targetDir : string; forceOverwrite : boolean);
27314:  Procedure copyDirklib( sourceDir : string; destinationDir : string; silent : boolean);
27315:  Procedure createHideDirklib( dirName : string; forceDelete : boolean);
27316:  Procedure deleteDirectoryIfExistsklib( dirName : string; silent : boolean);
27317:  Procedure myMoveFileklib( sourceFileName : string; targetFileName : string);
27318:  Procedure createEmptyFileIfNotExistsklib( filename : string);
27319:  Procedure createEmptyFileklib( filename : string);
27320:  Function checkIfIsWindowsSubDirklib( subDir : string; mainDir : string) : boolean');
27321:  Function getParentDirklib( source : string) : string');
27322:  Function getValidFullPathInWindowsStyleklib( path : string) : string');
27323:  Function getPathInWindowsStyleklib( path : string) : string');
27324:  Function getStringWithEnvVariablesReadedklib( source : string) : string');
27325:  Function setProcessWindowToForegroundklib( processName : string) : boolean');
27326:  Function getPIDOfCurrentUserByProcessNameklib( nameProcess : string): DWORD');
27327:  Function getWindowsUsernameklib : string');
27328:  Function checkUserOfProcessklib( userName : String; PID : DWORD) : boolean');
27329:  Function getPIDCredentialsklib( PID : DWORD) : TPIDCredentials');
27330:  Function getPIDByProcessNameklib( nameProcess : string) : DWORD');
27331:  Function getMainWindowHandleByPIDklib( PID : DWORD) : DWORD');
27332:  Procedure closeApplicationklib( handle : THandle);
27333:  Function sendMemoryStreamUsing_WM_COPYDATAklib( handle : THandle; data : TMemoryStream) : integer');
27334:  Function sendStringUsing_WM_COPYDATAklib(handle:THandle; data:string; msgIdentifier: integer):integer');
27335:  Procedure mySetForegroundWindowklib( handle : THandle);
27336:  Function checkIfWindowExistsklib( className : string; captionForm : string) : boolean');
```

```
27337:  Function myFindWindowklib( className : string; captionForm : string) : THandle');
27338:  Function checkIfExistsKeyIn_HKEY_LOCAL_MACHINEklib( key : string) : boolean');
27339:  Procedure waitForMultipleklib( processHandle : THandle; timeout : DWORD; modalMode : boolean);
27340:  Procedure waitForklib( processHandle : THandle; timeout : DWORD; modalMode : boolean);
27341:  Procedure raiseLastSysErrorMessageklib');
27342:  Function getLastSysErrorMessageklib : string');
27343:  Function getLocaleDecimalSeparatorklib : char');
27344:  Procedure terminateCurrentProcessklib( exitCode : Cardinal; raiseExceptionEnabled : boolean);
27345:  Procedure myTerminateProcessklib(processHandle:THandle;exitCode:Cardinal;raiseExceptionEnabled: boolean);
27346:  //Function fixedGetNamedSecurityInfo( pObjectName : LPWSTR; ObjectType : SE_OBJECT_TYPE; SecurityInfo :
        SECURITY_INFORMATION; ppsidOwner, ppsidGroup : PPSID; ppDacl, ppSacl : PPACL; var ppSecurityDescriptor :
        PSECURITY_DESCRIPTOR) : DWORD');
27347: end;
27348:
27349: https://github.com/lonewolfonline/delphi-utilities/blob/master/AzuliaUtils.pas
27350:
27351: procedure SIRegister_TAzuliaHTML(CL: TPSPascalCompiler);
27352: begin
27353:   //with RegClassS(CL,'TComponent', 'TAzuliaHTML') do
27354:   with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliaHTML') do  begin
27355:     RegisterMethod('Function HeadTop( Title : string) : string');
27356:     Function HeadClose : string');
27357:     Function BodyTop(LeftMargin,RightMargin:Integer;TextColor,LinkColor,VisitedLinkColor,ActiveLinkColor,
        BackgroundColor,BackgroundImage:string;FixedBackground:bool;Extra:string):string');
27358:     Function BodyClose : string');
27359:     Function ConvertDOSPathToHTML( WindowsPath : string) : string');
27360:     Function ConvertHTMLPathToDOS( WindowsPath : string) : string');
27361:   end;
27362: end;
27363: (*----------------------------------------------------------------------------*)
27364: procedure SIRegister_TAzuliaDisk(CL: TPSPascalCompiler);
27365: begin
27366:   //with RegClassS(CL,'TComponent', 'TAzuliaDisk') do
27367:   with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliaDisk') do  begin
27368:     Function GetDiskLabel( Drive : char) : string');
27369:     Function GetDiskSerial( Drive : char) : string');
27370:     Function GetDiskFileSystem( Drive : char) : string');
27371:     Function GetDiskType( Drive : char) : integer');
27372:     Function GetDiskFlags( Drive : char) : integer');
27373:     Procedure ChangeDiskLabel( Drive : char; NewLabel : string);
27374:   end;
27375: end;
27376: (*----------------------------------------------------------------------------*)
27377: procedure SIRegister_TAzuliaSpeaker(CL: TPSPascalCompiler);
27378: begin
27379:   //with RegClassS(CL,'TComponent', 'TAzuliaSpeaker') do
27380:   with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliaSpeaker') do begin
27381:     Procedure Delay( MSecs : Integer);
27382:     Procedure Play( Freq : Word; MSecs : Integer);
27383:     Procedure Stop');
27384:   end;
27385: end;
27386: (*----------------------------------------------------------------------------*)
27387: procedure SIRegister_TAzuliaFiles(CL: TPSPascalCompiler);
27388: begin
27389:   //with RegClassS(CL,'TOBJECT', 'TAzuliaFiles') do
27390:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TAzuliaFiles') do begin
27391:     Procedure FindRecursive(const path: string;const mask:string;LogFunction:TLogFunct;Re_curse:Boolean);
27392:     Procedure SearchFileExt( const Dir, Ext : string; Files : TStrings);
27393:     Function GetFileSize( const FileName : string) : LongInt');
27394:     Function GetKBFileSize( FileSize : integer) : string');
27395:     Function LongToShortFilename( LongFilename : string) : string');
27396:     Function ShortToLongFilename( LongFilename : string) : string');
27397:     Procedure AppendDataToFilename( Filename, Data : string);
27398:     Procedure SaveDataToFilename( Filename, Data : string);
27399:     Procedure CopyFile( const FileName, DestName : string);
27400:     Procedure MoveFile( const FileName, DestName : string);
27401:     Function HasAttr( const FileName : string; Attr : Word) : Boolean');
27402:     Function ExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle');
27403:     Function GetWindowsDir : string');
27404:     Function GetTempDir : string');
27405:     Function GetINIString( Section, Ident, DefaultString, Filename : string): string');
27406:   end;
27407: end;
27408: (*----------------------------------------------------------------------------*)
27409: procedure SIRegister_TAzuliaStrings(CL: TPSPascalCompiler);
27410: begin
27411:   //with RegClassS(CL,'TOBJECT', 'TAzuliaStrings') do
27412:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TAzuliaStrings') do begin
27413:     Function NPos( C : Char; S : string; N : Byte) : Byte');
27414:     Function IntToRoman( Value : Longint) : string');
27415:     Function RomanToInt( const S : string) : Longint');
27416:     Function GetDirFromStr( DirectoryString : string; IndexFromRight : integer) : string');
27417:     Function RemoveChars( InputString : string; CharToRemove : Char) : string');
27418:     Function GetStringFromGarbage( InputString : string; Index : integer) : string');
27419:     Function SplitupWithGarbage( InputString : string; CommonSplitter : char) : string');
27420:     Function GetCount( Character : char; InputString : string) : integer');
27421:     Function RemoveTrailingChars( StringToRemoveFrom : string; Trailer : char) : string');
27422:   end;
```

```
27423: end;
27424: (*----------------------------------------------------------------------------*)
27425: procedure SIRegister_TSystemRegistry(CL: TPSPascalCompiler);
27426: begin
27427:   //with RegClassS(CL,'TOBJECT', 'TSystemRegistry') do
27428:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TSystemRegistry') do begin
27429:     Function GetStringFromRegistry( Root : integer; Key, Value : string) : string');
27430:     Function SaveStringToRegistry( Root : integer; Key, Value, Name : string) : bool');
27431:     Procedure ToggleBinaryValue( RootKey : integer; Path, Key : string);
27432:     Procedure DeleteValueInRegistry( RootKey : integer; Path, Name : string);
27433:     Procedure RenameValueInRegistry( RootKey : integer; Path, OldName, NewName : string);
27434:   end;
27435: end;
27436: (*----------------------------------------------------------------------------*)
27437: procedure SIRegister_TBrowseForFolderDialog(CL: TPSPascalCompiler);
27438: begin
27439:   //with RegClassS(CL,'TOBJECT', 'TBrowseForFolderDialog') do
27440:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TBrowseForFolderDialog') do  begin
27441:     RegisterProperty('Title', 'string', iptrw);
27442:     Function Execute : Boolean');
27443:     RegisterProperty('Path', 'string', iptr);
27444:   end;
27445: end;
27446: (*----------------------------------------------------------------------------*)
27447: procedure SIRegister_IShellLink(CL: TPSPascalCompiler);
27448: begin
27449:   //with RegClassS(CL,'TOBJECT', 'IShellLink') do
27450:   with CL.AddClassN(CL.FindClass('TOBJECT'),'IShellLink') do begin
27451:     Function GetPath(pszFile:LPSTR;cchMaxPath:integer; var pfd:TWin32FindData;fFlags:DWORD):HResult');
27452:     Function GetIDList( var ppidl : PITEMIDLIST) : HResult');
27453:     Function SetIDList( pidl : PITEMIDLIST) : HResult');
27454:     Function GetDescription( pszName : LPSTR; cchMaxName : integer) : HResult');
27455:     Function SetDescription( pszName : LPSTR) : HResult');
27456:     Function GetWorkingDirectory( pszDir : LPSTR; cchMaxPath : integer) : HResult');
27457:     Function SetWorkingDirectory( pszDir : LPSTR) : HResult');
27458:     Function GetArguments( pszArgs : LPSTR; cchMaxPath : integer) : HResult');
27459:     Function SetArguments( pszArgs : LPSTR) : HResult');
27460:     Function GetHotkey( var pwHotkey : word) : HResult');
27461:     Function SetHotkey( wHotkey : word) : HResult');
27462:     Function GetShowCmd( var piShowCmd : integer) : HResult');
27463:     Function SetShowCmd( iShowCmd : integer) : HResult');
27464:     Function GetIconLocation( pszIconPath LPSTR;cchIconPath:integer;var piIcon: integer): HResult');
27465:     Function SetIconLocation( pszIconPath : LPSTR; iIcon : integer) : HResult');
27466:     Function SetRelativePath( pszPathRel : LPSTR; dwReserved : DWORD) : HResult');
27467:     Function Resolve( Wnd : HWND; fFlags : DWORD) : HResult');
27468:     Function SetPath( pszFile : LPSTR) : HResult');
27469:   end;
27470: end;
27471: (*----------------------------------------------------------------------------*)
27472: procedure SIRegister_AzuliaUtils(CL: TPSPascalCompiler);
27473: begin
27474:   CL.AddTypeS('azucharSet', 'set of char');
27475:   SIRegister_IShellLink(CL);
27476:   SIRegister_TBrowseForFolderDialog(CL);
27477:   SIRegister_TSystemRegistry(CL);
27478:   SIRegister_TAzuliaStrings(CL);
27479:   CL.AddTypeS(TLogFunct', 'Function ( const path : string; const SRec : TSearchRec) : Boolean');
27480:   SIRegister_TAzuliaFiles(CL);
27481:   SIRegister_TAzuliaSpeaker(CL);
27482:   SIRegister_TAzuliaDisk(CL);
27483:   SIRegister_TAzuliaHTML(CL);
27484:   CL.AddTypeS('GrabType', '( GTSCREEN, GTWINDOW, GTCLIENT );
27485:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidDest');
27486:   CL.AddClassN(CL.FindClass('TOBJECT'),'EFCantMove');
27487:  Function azuBoolToStr( value : boolean) : string');
27488:  Procedure azuChangeWallpaper( FName : string; IsTiled, IsStretch : Boolean);
27489:  Function azuDoIExist( WndTitle : string) : Boolean');
27490:  Procedure azuAlert( Text : string);
27491:  Procedure azuTileImage( SourceImage, DestImage : TImage);
27492:  Function azuIncChar( iX : Char) : Char');
27493:  Function azuDecChar( iX : Char) : Char');
27494:  Procedure azuSaveForm( F : TForm; Filename : string);
27495:  Procedure azuLoadForm( F : TForm; Filename : string);
27496:  Function azuToggleBool( totoggle : bool) : bool');
27497:  Function azuremoveTrailingChars( const s : string; chars : azucharSet) : string');
27498:  Function azuremoveLeadChars( const s : string; chars : azucharSet) : string');
27499:  Function azuremoveChars( const s : string; chars : char) : string');
27500:  Procedure azuDelay( MSecs : Integer);
27501:  Function azuSHBrowseDialog( title : string; Handle : integer) : string');
27502:  Procedure azuregisterfiletype( Extention, REGDesc, UserDesc, Icon, Appl : string);
27503:  Procedure azuRemoveFileType( Extention, RegDesc : string);
27504:  Procedure azuGetFunctionNamesFromDLL( DLLName : string; List : TStrings);
27505:  Procedure azuSaveListViewToFile( AListView : TListView; sFileName : string);
27506:  Procedure azuLoadListViewToFile( AListView : TListView; sFileName : string);
27507:  Procedure azuLoadCSV( Filename : string; sg : TStringGrid);
27508:  Function azuJPEGDimensions( Filename : string; var X, Y : Word) : boolean');
27509:  Procedure azuResizeImage( FileName : string; MaxWidth : Integer);
27510:  Function azuSaveJPEGPictureFile(Bitmap:TBitmap;FilePath,FileName:string;Quality:Integer): Boolean');
27511:  Function azuLoadJPEGPictureFile( Bitmap : TBitmap; FilePath, FileName : string) : Boolean');
```

```
27512:  Procedure azuSmoothResize( Src, Dst : TBitmap);
27513:  Function azuMsecToStr( Milli : Cardinal) : string');
27514:  Procedure azuDirToStrings( APath : string; AStrings : TStrings; WithExt, WithPath : boolean);
27515:  Procedure azuSetWallPaper( ImageFileName : string; IsTiled : boolean);
27516:  Function azuGetFileSize( FileName : string) : int64');
27517:  Function azuGetExeByExtension( sExt : string) : string');
27518:  Function azuRefreshScreenIcons : Boolean');
27519:  Function azuIndexToColor( AIndex : integer) : TColor');
27520:  Function azuColorToIndex( AColor : TColor) : integer');
27521:  Function azuTitleCase( Text2 : string) : string');
27522:  Function azuToggleCase( Text2 : string) : string');
27523:  Function azuSentenceCase( Text2 : string) : string');
27524:  Procedure azuGrabScreen( bm : TBitMap; gt : GrabType);
27525:  Procedure azuSaveStringGrid( StringGrid : TStringGrid; FileName : String);
27526:  Procedure azuLoadStringGrid( StringGrid : TStringGrid; FileName : String);
27527:  Function azuGetInetFile( const fileURL, FileName : String) : boolean');
27528:  Procedure azuRunOnStartup( sProgTitle, sCmdLine : string; bRunOnce : boolean);
27529:  Procedure azuRemoveOnStartup( sProgTitle : string);
27530:  CL.AddConstantN('SInvalidDest','String').SetString( 'Destination %s does not exist');
27531:  SFCantMove','String').SetString( 'Cannot move file %s');
27532:  AzuMsg1','String').SetString( 'File "%s" does not exist!');
27533:  AzuMsg2','String').SetString( '"%s" is not a ListView file!');
27534:  sr_WindowMetrics','String').SetString( 'Control Panel\Desktop\WindowMetrics\');
27535:  sr_ShellIconSize','String').SetString( 'Shell Icon Size');
27536: end;
27537:
27538: procedure SIRegister_TALWinInetHTTPClient2(CL: TPSPascalCompiler);
27539: begin
27540:   //with RegClassS(CL,'TALHTTPClient2', 'TALWinInetHTTPClient2') do
27541:   with CL.AddClassN(CL.FindClass('TALHTTPClient2'),'TALWinInetHTTPClient2') do begin
27542:     RegisterMethod('Constructor Create');
27543:     Procedure Free');
27544:     Procedure Connect');
27545:     Procedure Disconnect');
27546:     AccessType', 'TALWinInetHttpInternetOpenAccessType', iptrw);
27547:     InternetOptions', 'TAlWininetHTTPClientInternetOptionSet', iptrw);
27548:     OnStatus', 'TAlWinInetHTTPClientStatusEvent', iptrw);
27549:     IgnoreSecurityErrors', 'Boolean', iptrw);
27550:   end;
27551: end;
27552:
27553: procedure SIRegister_TALHTTPClient2(CL: TPSPascalCompiler);
27554: begin
27555:   //with RegClassS(CL,'TObject', 'TALHTTPClient') do
27556:   with CL.AddClassN(CL.FindClass('TObject'),'TALHTTPClient2') do begin
27557:     Constructor Create');
27558:     Procedure Free');
27559:     Procedure Get(const aUrl:AnsiString;const aRequestFields: TALStrings;const aResponseContent:TStream;
      const aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray;const
      aEncodeRequestFields:Bool);');
27560:     Procedure Get1( const aUrl: AnsiString; const aResponseContent : TStream; const aResponseHeader :
      TALHTTPResponseHeader2; const ARequestHeaderValues : TALNameValueArray);');
27561:     Function Get2(const aUrl: AnsiString; const ARequestHeaderValues : TALNameValueArray) : AnsiString;');
27562:     Function Get3(const aUrl: AnsiString;const aRequestFields:TALStrings;const ARequestHeaderValues:
      TALNameValueArray;const aEncodeRequestFields:Boolean): AnsiString;');
27563:     Procedure Post(const aUrl:AnsiString; const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);');
27564:     Procedure Post1(const aUrl: AnsiString; const aPostDataStream: TStream; const aResponseContent:
      TStream; const aResponseHeader:TALHTTPResponseHeader2; const ARequestHeaderValues: TALNameValueArray);');
27565:     Function Post2(const aUrl: AnsiString; const ARequestHeaderValues: TALNameValueArray) : AnsiString;');
27566:     Function Post3(const aUrl:AnsiString;const aPostDataStream:TStream;const ARequestHeaderValues:
      TALNameValueArray):AnsiString;');
27567:     Procedure PostUrlEncoded(const aUrl:AnsiString;const aRequestFields:TALStrings;const aResponseContent
      : TStream; const aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray;const
      aEncodeRequestFields:Boolean);
27568:     Function PostUrlEncoded1(const aUrl:AnsiString;const aRequestFields:TALStrings;const
      ARequestHeaderValues : TALNameValueArray;const aEncodeRequestFields:Boolean):AnsiString;');
27569:     Procedure PostMultipartFormData(const aUrl: AnsiString; const aRequestFields : TALStrings; const
      aRequestFiles : TALMultiPartFormDataContents; const aResponseContent : TStream; const aResponseHeader' +
27570:       ' : TALHTTPResponseHeader2; const ARequestHeaderValues : TALNameValueArray);');
27571:     Function PostMultiPartFormData1(const aUrl: AnsiString; const aRequestFields: TALStrings; const
      aRequestFiles:TALMultiPartFormDataContents;const ARequestHeaderValues: TALNameValueArray): AnsiString;');
27572:     Procedure Head(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);');
27573:     Function Head1(const aUrl: AnsiString; const ARequestHeaderValues: TALNameValueArray) : AnsiString;');
27574:     Procedure Trace(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);');
27575:     Function trace1( const aUrl : AnsiString; const ARequestHeaderValues : TALNameValueArray) :
      AnsiString;');
27576:     Procedure Put(const aUrl:AnsiString;const aPutDataStream:TStream;const aResponseContent:TStream; const
      aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues: TALNameValueArray;');
27577:     Function Put1(const aURL:Ansistring;const aPutDataStream:TStream;const ARequestHeaderValues:
      TALNameValueArray):AnsiString;;
27578:     Procedure Delete(const aUrl: AnsiString;const aResponseContent:TStream;const aResponseHeader:
      TALHTTPResponseHeader2;const ARequestHeaderValues : TALNameValueArray);');
27579:     Function Delete1(const aURL:Ansistring;const ARequestHeaderValues: TALNameValueArray): AnsiString;');
27580:     Procedure Options( const aUrl : AnsiString; const aResponseContent : TStream; const aResponseHeader :
      TALHTTPResponseHeader2; const ARequestHeaderValues : TALNameValueArray);');
27581:     Function Options1(const aURL:Ansistring; const ARequestHeaderValues:TALNameValueArray) : AnsiString;');
```

```
27582:     RegisterProperty('ConnectTimeout', 'Integer', iptrw);
27583:     SendTimeout', 'Integer', iptrw);
27584:     ReceiveTimeout', 'Integer', iptrw);
27585:     UploadBufferSize', 'cardinal', iptrw);
27586:     ProxyParams', 'TALHTTPClientProxyParams', iptr);
27587:     RequestHeader', 'TALHTTPRequestHeader2', iptr);
27588:     ProtocolVersion', 'TALHTTPProtocolVersion', iptrw);
27589:     UserName', 'AnsiString', iptrw);
27590:     Password', 'AnsiString', iptrw);
27591:     OnUploadProgress', 'TALHTTPClientUploadProgressEvent', iptrw);
27592:     OnDownloadProgress', 'TALHTTPClientDownloadProgressEvent', iptrw);
27593:     OnRedirect', 'TAlHTTPClientRedirectEvent', iptrw);
27594:   end;
27595: end;
27596:
27597: procedure SIRegister_ALHttpClient2(CL: TPSPascalCompiler);
27598: begin
27599:   TALHTTPPropertyChangeEvent', 'Procedure ( sender: Tobject; const PropertyIndex: Integer)');
27600:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )');
27601:   TALHTTPMethod2', '( HTTPmt_Get, HTTPmt_Post, HTTPmt_Head, HTTPmt_'
27602:                    +'Trace, HTTPmt_Put, HTTPmt_Delete, HTTPmt_Options )');
27603:   SIRegister_TALHTTPRequestHeader2(CL);
27604:   TALNameValuePair', 'record Name : ansistring; Value : ansistring; end');
27605:   TALNameValueArray', 'array of TALNameValuePair');
27606:   SIRegister_TALHTTPCookie2(CL);
27607:   SIRegister_TALHTTPCookieCollection2(CL);
27608:   SIRegister_TALHTTPResponseHeader2(CL);
27609:   SIRegister_EALHTTPClientException2(CL);
27610:   SIRegister_TALHTTPClientProxyParams2(CL);
27611:   TAlHTTPClientRedirectEvent', 'Procedure ( sender : Tobject; const NewURL : AnsiString)');
27612:   TALHTTPClientUploadProgressEvent', 'Procedure ( sender : Tobject; Sent: Integer; Total: Integer)');
27613:   TALHTTPClientDownloadProgressEvent', 'Procedure ( sender: Tobject; Read: Integer; Total: Integer)');
27614:   SIRegister_TALHTTPClient2(CL);
27615:  Procedure ALHTTPEncodeParamNameValues2( const ParamValues : TALStrings)');
27616:  //Procedure ALExtractHTTPFields( Separators,WhiteSpace,Quotes: TSysCharSet; Content:PAnsiChar; Strings :
      TALStrings; StripQuotes : Boolean)');
27617:  Function AlRemoveShemeFromUrl2( const aUrl : AnsiString) : ansiString');
27618:  Function AlExtractShemeFromUrl2( const aUrl : AnsiString) : TInternetScheme');
27619:  Function AlExtractHostNameFromUrl2( const aUrl : AnsiString) : AnsiString');
27620:  Function AlExtractDomainNameFromUrl2( const aUrl : AnsiString) : AnsiString');
27621:  Function AlExtractUrlPathFromUrl2( const aUrl : AnsiString) : AnsiString');
27622:  Function AlInternetCrackUrl2(const aUrl:AnsiString;var SchemeName,HostName,UserName,Password,UrlPath,
      ExtraInfo : AnsiString;var PortNumber:integer):Bool;
27623:  Function AlInternetCrackUrl12(const aUrl:AnsiString;var SchemeName,HostName,UserName,Password,UrlPath,
      Anchor: AnsiString;const Query:TALStrings;var PortNumber:integer): Boolean;
27624:  Function AlInternetCrackUrl22( var Url:AnsiString;var Anchor:AnsiString;const Query:TALStrings):
      Boolean;');
27625:  Function AlRemoveAnchorFromUrl2( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;');
27626:  Function AlRemoveAnchorFromUrl12( const aUrl : AnsiString) : AnsiString;');
27627:  Function AlCombineUrl2( const RelativeUrl, BaseUrl : AnsiString) : AnsiString;');
27628:  Function AlCombineUrl12( const RelativeUrl,BaseUrl,Anchor AnsiString;const Query:TALStrings):
      AnsiString;');
27629:  Function ALGmtDateTimeToRfc822Str2( const aValue : TDateTime) : AnsiString');
27630:  Function ALDateTimeToRfc822Str2( const aValue : TDateTime) : AnsiString');
27631:  Function ALTryRfc822StrToGMTDateTime2( const S : AnsiString; out Value : TDateTime) : Boolean');
27632:  Function ALRfc822StrToGMTDateTime2( const s : AnsiString) : TDateTime');
27633:  Function ALTryIPV4StrToNumeric2( const aIPv4Str : AnsiString; var aIPv4Num : Cardinal) : Boolean');
27634:  Function ALIPV4StrToNumeric2( const aIPv4 : AnsiString) : Cardinal');
27635:  Function ALNumericToIPv4Str2( const aIPv4 : Cardinal) : ansiString');
27636:  Function ALIPv4EndOfRange2( const aStartIPv4 : Cardinal; aMaskLength : integer) : Cardinal');
27637:  Function ALZeroIpV62 : TALIPv6Binary');
27638:  Function ALIsValidIPv6BinaryStr2( const aIPV6BinaryStr : ansiString) : boolean');
27639:  Function ALTryIPV6StrToBinary2( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean');
27640:  Function ALIPV6StrTobinary2( const aIPv6 : AnsiString) : TALIPv6Binary');
27641:  Function ALBinaryToIPv6Str2( const aIPv6 : TALIPv6Binary) : ansiString');
27642:  Function ALBinaryStrToIPv6Binary2( const aIPV6BinaryStr : ansiString) : TALIPv6Binary');
27643:  Function ALBinaryStrToIPv6Str2( const aIPV6BinaryStr : ansiString) : ansiString');
27644:  Function ALIPv6EndOfRange2( const aStartIPv6 : TALIPv6Binary; aMaskLength : integer) : TALIPv6Binary');
27645:  Procedure ALIPv6SplitParts2(const aIPv6: TALIPv6Binary;var aLowestPart:UInt64; var aHigestPart: UInt64)');
27646:  CL.AddConstantN('cALHTTPCLient_MsgInvalidURL2','String').SetString('Invalid url ''%s''- only supports
      ''http'' and ''https'' schemes');
27647:  CL.AddConstantN('cALHTTPCLient_MsgInvalidHTTPRequest2','String').SetString('Invalid HTTP Request:Length
      is 0');
27648:  CL.AddConstantN('cALHTTPCLient_MsgEmptyURL2','String').SetString( 'Empty URL');
27649:  function ALHTTPEncode(const AStr: AnsiString): AnsiString;');
27650:  function ALHTTPDecode(const AStr: AnsiString): AnsiString;');
27651: end;
27652:
27653: CL.AddDelphiFunction('procedure RunDosInMemo(DosApp: string; AMemo:TMemo);');
27654:
27655: {type TTextHandler =} procedure TTextHandlerQ(const aText: string);
27656: begin
27657:   memo2.lines.add(atext);
27658: end;
27659: writeln(itoa(JExecute('cmd /C dir *.*',@TTextHandlerQ, true, false)));
27660: writeln(itoa(JExecute3('cmd /C dir *.*',Nil, true, false)));
27661:
27662: memo2.lines.add(getIPs.text);
27663:    srlist:= TStringlist.create;
```

```
27664:    GetDNSServers(srlist)
27665:    writeln(srlist.text)
27666:    srlist.Free;
27667:
27668: https://presse.tuv.com/en/photovoltaics-tuv-rheinland-investigates-potential-for-applications-on-rail-
       infrastructure/
27669:
27670: procedure SIRegister_TRestUtils(CL: TPSPascalCompiler);
27671: begin
27672:   //with RegClassS(CL,'TOBJECT', 'TRestUtils') do
27673:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TRestUtils') do begin
27674:     Function Base64Encode( const AValue : String) : String');
27675:     Function Base64Decode( const AValue : String) : String');
27676:   end;
27677: end;
27678:
27679: (*----------------------------------------------------------------------------*)
27680: procedure SIRegister_TStatusCode(CL: TPSPascalCompiler);
27681: begin
27682:   //with RegClassS(CL,'TOBJECT', 'TStatusCode') do
27683:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TStatusCode') do begin
27684:     Function CONTINUE : TReponseCode');
27685:     Function SWITCHING_PROTOCOLS : TReponseCode');
27686:     Function OK : TReponseCode');
27687:     Function CREATED : TReponseCode');
27688:     Function ACCEPTED : TReponseCode');
27689:     Function NON_AUTHORITATIVE_INFORMATION : TReponseCode');
27690:     Function NO_CONTENT : TReponseCode');
27691:     Function RESET_CONTENT : TReponseCode');
27692:     Function PARTIAL_CONTENT : TReponseCode');
27693:     Function MULTIPLE_CHOICES : TReponseCode');
27694:     Function MOVED_PERMANENTLY : TReponseCode');
27695:     Function FOUND : TReponseCode');
27696:     Function SEE_OTHER : TReponseCode');
27697:     Function NOT_MODIFIED : TReponseCode');
27698:     Function USE_PROXY : TReponseCode');
27699:     Function TEMPORARY_REDIRECT : TReponseCode');
27700:     Function BAD_REQUEST : TReponseCode');
27701:     Function UNAUTHORIZED : TReponseCode');
27702:     Function PAYMENT_REQUIRED : TReponseCode');
27703:     Function FORBIDDEN : TReponseCode');
27704:     Function NOT_FOUND : TReponseCode');
27705:     Function METHOD_NOT_ALLOWED : TReponseCode');
27706:     Function NOT_ACCEPTABLE : TReponseCode');
27707:     Function PROXY_AUTHENTICATION_REQUIRED : TReponseCode');
27708:     Function REQUEST_TIMEOUT : TReponseCode');
27709:     Function CONFLICT : TReponseCode');
27710:     Function GONE : TReponseCode');
27711:     Function LENGTH_REQUIRED : TReponseCode');
27712:     Function PRECONDITION_FAILED : TReponseCode');
27713:     Function REQUEST_ENTITY_TOO_LARGE : TReponseCode');
27714:     Function REQUEST_URI_TOO_LONG : TReponseCode');
27715:     Function UNSUPPORTED_MEDIA_TYPE : TReponseCode');
27716:     Function REQUESTED_RANGE_NOT_SATISFIABLE : TReponseCode');
27717:     Function EXPECTATION_FAILED : TReponseCode');
27718:     Function UNPROCESSABLE_ENTITY : TReponseCode');
27719:     Function INTERNAL_SERVER_ERROR : TReponseCode');
27720:     Function NOT_IMPLEMENTED : TReponseCode');
27721:     Function BAD_GATEWAY : TReponseCode');
27722:     Function SERVICE_UNAVAILABLE : TReponseCode');
27723:     Function GATEWAY_TIMEOUT : TReponseCode');
27724:     Function HTTP_VERSION_NOT_SUPPORTED : TReponseCode');
27725:   end;
27726: end;
27727:
27728: (*----------------------------------------------------------------------------*)
27729: procedure SIRegister_RestUtils(CL: TPSPascalCompiler);
27730: begin
27731:  MediaType_Json','String').SetString( 'application/json');
27732:  MediaType_Xml','String').SetString( 'text/xml');
27733:  LOCALE_PORTUGUESE_BRAZILIAN','String').SetString( 'pt-BR');
27734:  LOCALE_US','String').SetString( 'en-US');
27735:   CL.AddTypeS('TReponseCode', 'record StatusCode : Integer; Reason : string; end');
27736:   SIRegister_TStatusCode(CL);
27737:   SIRegister_TRestUtils(CL);
27738: end;
27739:
27740: procedure SIRegister_DataSetConverter4DUtil(CL: TPSPascalCompiler);
27741: begin
27742:  CL.AddTypeS('TBooleanFieldType', '( bftUnknown, bftBoolean, bftInteger )');
27743:  CL.AddTypeS('TDataSetFieldType', '( dftUnknown, dftJSONObject, dftJSONArray )');
27744:  CL.AddDelphiFunction('Function DateTimeToISOTimeStamp( const dateTime : TDateTime) : string');
27745:  Function DateToISODate( const date : TDateTime) : string');
27746:  Function TimeToISOTime( const time : TTime) : string');
27747:  Function ISOTimeStampToDateTime( const dateTime : string) : TDateTime');
27748:  Function ISODateToDate( const date : string) : TDate');
27749:  Function ISOTimeToTime( const time : string) : TTime');
27750:  Function NewDataSetField(dataSet:TDataSet;const fieldType:TFieldType;const fieldName:string;const
       size:Integer; const origin:string;const displaylabel:string): TField');
```

```
27751:  Function BooleanToJSON( const value : Boolean) : TJSONValue');
27752:  Function BooleanFieldToType( const booleanField:TBooleanField): TBooleanFieldType');
27753:  Function DataSetFieldToType( const dataSetField:TDataSetField): TDataSetFieldType');
27754:  Function MakeValidIdent( const s : string) : string');
27755: end;
27756:
27757:  function wget3(aURL, afile: string; opendoc: boolean): boolean;
27758:  procedure RegisterProtocol(const Name, Describtion, ExecuteStr: string)');
27759:  procedure UnregisterProtocol(const Name: string);');
27760:  function GetNetworkConnections: string;
27761:  function GetNetworkDrives: string;
27762:
27763: wget3('https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Ffoodanddrink.scotsman.com%2Fwp-
       content%2Fuploads%2F2016%2F08%2Fitem-4-9-1024x683.jpg',
27764:             exepath+'examples/downpic.png', true);
27765:
27766: procedure SIRegister_PSResources(CL: TPSPascalCompiler);
27767: begin
27768:  CL.AddConstantN('_sBeginCompile','String').SetString( 'Compiling');
27769:  ('_sSuccessfullyCompiled','String').SetString( 'Succesfully compiled');
27770:  ('_sSuccessfullyExecuted','String').SetString( 'Succesfully executed');
27771:  ('_sRuntimeError','String').SetString( '[Runtime error] %s(%d:%d), bytecode(%d:%d): %s');
27772:  ('_sTextNotFound','String').SetString( 'Text not found');
27773:  ('_sUnnamed','String').SetString( 'Unnamed');
27774:  ('_sEditorTitle','String').SetString( 'Editor');
27775:  ('_sEditorTitleRunning','String').SetString( 'Editor - Running');
27776:  ('_sEditorTitlePaused','String').SetString( 'Editor - Paused');
27777:  ('_sEditorTitleStopped','String').SetString( 'Editor - Stopped');
27778:  ('_sInputBoxTiyle','String').SetString( 'Script');
27779:  ('_sFileNotSaved','String').SetString( 'File has not been saved, save now?');
27780:  ('_isRunningOrPaused','LongInt').Value.ts32 := ord(isRunning) or ord(isPaused);
27781:   SIRegister_TPSUtils(CL);
27782:   //https://github.com/fabriciocolombo/PascalScriptEx/tree/master/src
27783:   CL.AddDelphiFunction('function list_functions: TStringlist;');
27784:   function list_functions2: TStringlist;');
27785:   procedure InternalRaiseException(E: Exception);');
27786:   function ListFiles(ADirectory, AFilter: String; AList: TStrings): Boolean;');
27787:   Procedure CommandToList(S : String; List : TStrings);');
27788:  end;
27789:
27790: CL.AddDelphiFunction('function list_functions: TStringlist;');
27791: procedure InternalRaiseException(E: Exception);');
27792: function ListFiles(ADirectory, AFilter: String; AList: TStrings) : Boolean;
27793: Procedure CommandToList(S : String; List : TStrings);
27794: CL.AddDelphiFunction('procedure SetSynchroTime');
27795: CL.AddDelphiFunction('procedure letSynchroTime');
27796: procedure TimeSync');    //get internettime service
27797: function HttpGetDirect(const Url: string): string;
27798: function HttpGetDirect2(const Url: string): string;
27799: function HttpGetDirect4(const Url, agent: string): string;
27800: function HttpGet_Direct(const Url: string): string;');
27801: function wininet_HttpGetDirect(const Url: string): string;');
27802: function GetWinInetError(ErrorCode:Cardinal): string;
27803: function HttpGetDirect3(const ServerName,Resource,sUserAgent:string;Var Response:AnsiString):Integer;
27804: procedure WinInetCheck(Success: Boolean; aFunction: PChar);
27805: function WebGetData(const UserAgent: string; const Server: string; const Resource: string):string;
27806: procedure WebPostData(const UserAgent: string; const Server: string;
27807:                          const Resource: string; const Data: AnsiString);
27808: function WebPostData2(const UserAgent:string;const Server:string; const Resource:string;
27809:                          const Data:AnsiString): String;
27810: function URLEncode3(const Url: string): string;
27811:
27812: function FloatAsInteger(X: single): integer;
27813: begin
27814:   result:= PInteger(@X)^;
27815: end;
27816: function IntegerAsFloat(X: integer): single;
27817: begin
27818:   result:= PSingle(@X)^;
27819: end;
27820:
27821: procedure ParseString4(s,sep: string; sl: TStrings);');
27822: procedure LoadGridFromfile(SG: TStringGrid; fname: string);
27823: procedure SaveGridTofile(SG: TStringGrid; fname: string);
27824: procedure WriteVectorToGrid(SG: TStringGrid; vname: string; wval: TAffineVector);
27825: procedure WriteStringToGrid(SG: TStringGrid; vname: string; icol: longword; wval: string);
27826: procedure WriteFloatToGrid(SG: TStringGrid; vname: string; icol: longword; wval: double);
27827: https://github.com/P33a/SimTwo/blob/master/Utils.pas
27828: procedure ExtractIconFileToImageList(ImageList: TImageList; const Filename: string);
27829:
27830: procedure SIRegister_SynCrtSock(CL: TPSPascalCompiler);  -->SIRegister_synTHttpRequest
27831: begin
27832:  CL.AddConstantN('HTTP_RESP_STATICFILE','String').SetString( '!STATICFILE');
27833:  'HTTP_RESP_NORESPONSE','String').SetString( '!NORESPONSE');
27834:  'HTTP_DEFAULT_RESOLVETIMEOUT','LongInt').SetInt( 0);
27835:  'HTTP_DEFAULT_CONNECTTIMEOUT','LongInt').SetInt( 60000);
27836:  'HTTP_DEFAULT_SENDTIMEOUT','LongInt').SetInt( 30000);
27837:  'HTTP_DEFAULT_RECEIVETIMEOUT','LongInt').SetInt( 30000);
27838:   CL.AddTypeS('SynUnicode', 'UnicodeString');
```

```
27839:   'SockString', 'RawByteString');
27840:   'SynUnicode', 'WideString');
27841:   'SockString', 'AnsiString');
27842:   S('PPointer', '^Pointer // will not work');
27843:   'synTTextLineBreakStyle', '( tlbsLF, tlbsCRLF )');
27844:   'synUTF8String', 'AnsiString');
27845:   'synUTF8Encode', 'AnsiString');
27846:   'synPtrInt', 'NativeInt');
27847:   'synPtrUInt', 'NativeUInt');
27848:   'synPtrInt', 'integer');
27849:   'synPtrUInt', 'cardinal');
27850:    THttpSocketCompress','function(var DataRawByteString: sockstring; Compress: boolean): AnsiString');
27851:   //CL.AddTypeS('PPtrInt', '^PtrInt // will not work');
27852:   //CL.AddTypeS('PPtrUInt', '^PtrUInt // will not work');
27853:   CL.AddTypeS('TSockHeaders','array of SockString');
27854:   SIRegister_ECrtSocket(CL);
27855:   //CL.AddTypeS('TCrtSocketClass', 'class of TCrtSocket');
27856:   CL.AddTypeS('TCrtSocketLayer', '( cslTCP, cslUDP, cslUNIX )');
27857:   CL.AddTypeS('TCrtSocketPending', '( cspSocketError, cspNoData, cspDataAvailable )');
27858:   //CL.AddTypeS('PTextFile', '^TextFile // will not work');
27859:   SIRegister_TCrtSocket(CL);
27860:   CL.AddTypeS('THttpSocketCompressRec', 'record Name : SockString; Func : THttp'
27861:    +'SocketCompress; CompressMinSize : integer; end');
27862:   CL.AddTypeS('THttpSocketCompressRecDynArray', 'array of THttpSocketCompressRec');
27863:   CL.AddTypeS('THttpSocketCompressSet', 'Integer');
27864:   SIRegister_THttpSocket(CL);
27865:   CL.AddClassN(CL.FindClass('TOBJECT'),'THttpServer');
27866:   SIRegister_THttpServerSocket(CL);
27867:   SIRegister_THttpClientSocket(CL);
27868:   //CL.AddTypeS('THttpClientSocketClass', 'class of THttpClientSocket');
27869:   CL.AddTypeS('TNotifyThreadEvent', 'Procedure ( Sender : TThread)');
27870:   SIRegister_TSynThread(CL);
27871:   CL.AddClassN(CL.FindClass('TOBJECT'),'TSynThreadPoolTHttpServer');
27872:   SIRegister_THttpServerResp(CL);
27873:   //CL.AddTypeS('THttpServerRespClass', 'class of THttpServerResp');
27874:   CL.AddClassN(CL.FindClass('TOBJECT'),'TSynThreadPool');
27875:   SIRegister_TSynThreadPoolSubThread(CL);
27876:   SIRegister_TSynThreadPool(CL);
27877:   SIRegister_TSynThreadPoolTHttpServer(CL);
27878:   CL.AddClassN(CL.FindClass('TOBJECT'),'THttpServerGeneric');
27879:   CL.AddTypeS('THttpServerRequestAuthentication', '( hraNone, hraFailed, hraBas'
27880:    +'ic, hraDigest, hraNtlm, hraNegotiate, hraKerberos )');
27881:   SIRegister_THttpServerRequest(CL);
27882:   CL.AddTypeS('TOnHttpServerRequest','Function (Ctxt : THttpServerRequest): cardinal');
27883:   SIRegister_THttpServerGeneric(CL);
27884:   CL.AddTypeS('synULONGLONG', 'Int64');
27885:   'HTTP_OPAQUE_ID', 'ULONGLONG');
27886:   'HTTP_URL_GROUP_ID', 'HTTP_OPAQUE_ID');
27887:   'HTTP_SERVER_SESSION_ID', 'HTTP_OPAQUE_ID');
27888:   'THttpApiLoggingType', '( hltW3C, hltIIS, hltNCSA, hltRaw )');
27889:   'THttpApiLoggingRollOver', '( hlrSize, hlrDaily, hlrWeekly, hlrMonthly, hlrHourly )');
27890:   'THttpApiLoggingFlag','( hlfLocalTimeRollover, hlfUseUTF'
27891:    +'8Conversion, hlfLogErrorsOnly, hlfLogSuccessOnly )');
27892:   CL.AddTypeS('THttpApiLoggingFlags', 'set of THttpApiLoggingFlag');
27893:   CL.AddTypeS('THttpApiLogField', '( hlfDate, hlfTime, hlfClientIP, hlf'
27894:    +'UserName, hlfSiteName, hlfComputerName, hlfServerIP, hlfMethod, hlfURIStem'
27895:    +', hlfURIQuery, hlfStatus, hlfWIN32Status, hlfBytesSent, hlfBytesRecv, hlfT'
27896:    +'imeTaken, hlfServerPort, hlfUserAgent, hlfCookie, hlfReferer, hlfVersion, hlfHost, hlfSubStatus )');
27897:     'THttpApiLogFields', 'set of THttpApiLogField');
27898:   ('THttpApiRequestAuthentication', '( haBasic, haDigest, haNtlm, haNegotiate, haKerberos )');
27899:   ('THttpApiRequestAuthentications', 'set of THttpApiRequestAuthentication');
27900:   SIRegister_THttpApiServer(CL);
27901:   SIRegister_THttpServer(CL);
27902:   SIRegister_synTURI(CL);
27903:   CL.AddTypeS('THttpRequestAuthentication', '( wraNone, wraBasic, wraDigest, wraNegotiate )');
27904:   CL.AddTypeS('THttpRequestExtendedOptions', 'record IgnoreSSLCertificateErrors'
27905:    +' : Boolean; Auth : record UserName : SynUnicode; Password : SynUnicode; Sc'
27906:    +'heme : THttpRequestAuthentication; end ; end');
27907:   {$IFDEF SIRegister_synTHttpRequest(CL); }
27908:   //CL.AddTypeS('THttpRequestClass', 'class of THttpRequest');
27909:   SIRegister_TWinHttpAPI(CL);
27910:   SIRegister_TWinINet(CL);
27911:   SIRegister_EWinINet(CL);
27912:   SIRegister_TWinHTTP(CL);
27913:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWinHTTP');
27914:   SIRegister_TCurlHTTP(CL);
27915:  Function synOpenCrtSock(const aServer, aPort : SockString) : TCrtSocket');
27916:  Function synOpenHttp(const aServer, aPort : SockString) : THttpClientSocket');
27917:  Function synHttpGet(const server, port : SockString; const url : SockString) : SockString;');
27918:  Function synHttpGet1(const aURI : SockString) : SockString;');
27919:  Function synHttpPost(const server, port : SockString; const url, Data, DataType : SockString) : boolean');
27920:  Function synSendEmail(const Server,From,CSVDest,Subject,Text:SockString;const Headers:SockString;const
       User:SockString;const Pass:SockString;const Port:SockString;const TextCharSet:SockString):bool;
27921:  Function synSendEmailSubject( const Text : string) : SockString');
27922:  'STATUS_SUCCESS','LongInt').SetInt( 200);
27923:  'STATUS_CREATED','LongInt').SetInt( 201);
27924:  'STATUS_NOCONTENT','LongInt').SetInt( 204);
27925:  'STATUS_BADREQUEST','LongInt').SetInt( 400);
27926:  'STATUS_UNAUTHORIZED','LongInt').SetInt( 401);
```

```
27927:  'STATUS_FORBIDDEN','LongInt').SetInt( 403);
27928:  'STATUS_NOTFOUND','LongInt').SetInt( 404);
27929:  'STATUS_SERVERERROR','LongInt').SetInt( 500);
27930:  'STATUS_NOTIMPLEMENTED','LongInt').SetInt( 501);
27931:  Function synStatusCodeToReason( Code : integer) : SockString');
27932:  Function synResolveName(const Name:SockString;Family:Integer;SockProtocol:Int;SockType:int):SockString');
27933:  Function synBase64Encode( const s : SockString) : SockString');
27934:  Function synBase64Decode( const s : SockString) : SockString');
27935:  Function synHtmlEncode( const s : SockString) : SockString');
27936:  Function synGetRemoteMacAddress( const IP : SockString) : SockString');
27937: end;
27938:
27939: procedure SIRegister_THttpConnectionWinInet(CL: TPSPascalCompiler);
27940: begin
27941:   //with RegClassS(CL,'TInterfacedObject', 'THttpConnectionWinInet') do
27942:   with CL.AddClassN(CL.FindClass('TInterfacedObject'),'THttpConnectionWinInet') do begin
27943:     OnConnectionLost', 'THTTPConnectionLostEvent', iptrw);
27944:     Constructor Create( ARaiseExceptionOnError : Boolean)');
27945:     Procedure Free;');
27946:     Function SetAcceptTypes( AAcceptTypes : string) : IHttpConnection');
27947:     Function SetAcceptedLanguages( AAcceptedLanguages : string) : IHttpConnection');
27948:     Function SetContentTypes( AContentTypes : string) : IHttpConnection');
27949:     Function SetHeaders( AHeaders : TStrings) : IHttpConnection');
27950:     Headers', 'TStrings', iptr);
27951:     Procedure Get( AUrl : string; AResponse : TStream)');
27952:     Procedure Post( AUrl : string; AContent : TStream; AResponse : TStream)');
27953:     Procedure Put( AUrl : string; AContent : TStream; AResponse : TStream)');
27954:     Procedure Patch( AUrl : string; AContent : TStream; AResponse : TStream)');
27955:     Procedure Delete( AUrl : string; AContent : TStream; AResponse : TStream)');
27956:     Function GetResponseCode : Integer');
27957:     Function GetResponseHeader( const Header : string) : string');
27958:     Function GetEnabledCompression : Boolean');
27959:     Procedure SetEnabledCompression( const Value : Boolean)');
27960:     Function GetOnConnectionLost : THTTPConnectionLostEvent');
27961:     Procedure SetOnConnectionLost( AConnectionLostEvent : THTTPConnectionLostEvent)');
27962:     Function ConfigureTimeout( const ATimeOut : TTimeOut) : IHttpConnection');
27963:     Function ConfigureProxyCredentials( AProxyCredentials : TProxyCredentials) : IHttpConnection');
27964:     Procedure SetVerifyCert( const Value : boolean)');
27965:     Function GetVerifyCert : boolean');
27966:     Function SetAsync( const Value : Boolean) : IHttpConnection');
27967:     Procedure CancelRequest');
27968:     Function SetOnAsyncRequestProcess( const Value : TAsyncRequestProcessEvent) : IHttpConnection');
27969:     RegisterProperty(''ResponseErrorStatusText', 'string', iptrw);
27970:     CertificateContext', 'PCERT_CONTEXT', iptrw);
27971:     'BasicAuthentication_UserName', 'string', iptrw);
27972:     'BasicAuthentication_Password', 'string', iptrw);
27973:     'CertificateCheckDate', 'Boolean', iptrw);
27974:     'CertificateCheckHostName', 'boolean', iptrw);
27975:     'CertificateCheckAuthority', 'boolean', iptrw);
27976:     'CertificateIgnoreRevocation', 'boolean', iptrw);
27977:     'RaiseExceptionOnError', 'Boolean', iptrw);
27978:   end;
27979: end;
27980:
27981: procedure SIRegister_VelthuisFloatUtils(CL: TPSPascalCompiler);
27982: begin
27983:  Function IsNegativeInfinity( const AValue : Single) : Boolean;');
27984:  Function IsNegativeInfinity1( const AValue : Double) : Boolean;');
27985:  Function IsNegativeInfinity2( const AValue : Extended) : Boolean;');
27986:  Function IsPositiveInfinity( const AValue : Single) : Boolean;');
27987:  Function IsPositiveInfinity1( const AValue : Double) : Boolean;');
27988:  Function IsPositiveInfinity2( const AValue : Extended) : Boolean;');
27989:  Function GetSignificand( const AValue : Single) : UInt32;');
27990:  Function GetSignificand1( const AValue : Double) : UInt64;');
27991:  Function GetSignificand2( const AValue : Extended) : UInt64;');
27992:  Function GetMantissa( const AValue : Single) : UInt32;');
27993:  Function GetMantissa1( const AValue : Double) : UInt64;');
27994:  Function GetMantissa2( const AValue : Extended) : UInt64;');
27995:  Function GetExponent( const AValue : Single) : Integer;');
27996:  Function GetExponent1( const AValue : Double) : Integer;');
27997:  Function GetExponent2( const AValue : Extended) : Integer;');
27998:  Function IsDenormal( const AValue : Single) : Boolean;');
27999:  Function IsDenormal1( const AValue : Double) : Boolean;');
28000:  Function IsDenormal2( const AValue : Extended) : Boolean;');
28001:  Function MakeSingle( Sign : byte; Significand : Longword; Exponent: Integer): Single');
28002:  Function MakeDouble( Sign : byte; Significand : UInt64; Exponent: Integer) : Double');
28003:  Function MakeExtended( Sign : byte; Significand : UInt64; Exponent: Integer): Extended');
28004:   //CL.AddTypeS('PUInt8', '^UInt8 // will not work');
28005:   //CL.AddTypeS('PUInt16', '^UInt16 // will not work');
28006:   //CL.AddTypeS('PUInt32', '^UInt32 // will not work');
28007:   //CL.AddTypeS('PExt80Rec', '^TExt80Rec // will not work');
28008:   CL.AddTypeS('TExt80Rec', 'record Significand : UInt64; ExponentAndSign : Word; end');
28009:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work'); +'; end');
28010:  /'CSingleExponentShift','LongInt').SetInt( 23);
28011:  ('CDoubleExponentShift','LongInt').SetInt( 52);
28012:  ('CSingleExponentMask','LongWord').SetUInt( $FF);
28013:  ('CDoubleExponentMask','LongWord').SetUInt( $7FF);
28014:  ('CExtendedExponentMask','LongWord').SetUInt( $7FFF);
28015:  ('CSingleBias','LongInt').SetInt( CSingleExponentMask shr 1);
```

```
28016:  ('CDoubleBias','LongInt').SetInt( CDoubleExponentMask shr 1);
28017:  ('CExtendedBias','LongInt').SetInt( CExtendedExponentMask shr 1);
28018:  ('CSingleSignMask','LongInt').SetInt( UInt32 ( 1 ) shl 31);
28019:  ('CDoubleSignMask','LongInt').SetInt( UInt64 ( 1 ) shl 63);
28020: end;
28021:
28022:  function BytesToHex2(const Bytes: TBytes): string;
28023:   {Creates a hexadecimal representation of the bytes in an array.
28024:     @param Bytes [in] Byte array.
28025:     return Hexadecimal representation of bytes in array.  }
28026:  function TryHexToBytes(HexStr: string; out Bytes: TBytes): Boolean;  *)  UHexUtils.pas
28027:
28028: CL.AddTypeS('TExeFileKind', '( fkUnknown, fkError, fkDOS, fkExe32, fkExe16, fkDLL32, fkDLL16, fkVXD )');
28029: Function ExeFileType( const FileName : string) : TExeFileKind');
28030: Function getExeFileType( const FileName : string) : TExeFileKind');
28031: function ExeFileType(const FileName: string): TExeFileKind;  UExeFileType.pas
28032: function ListFiles2(const Dir, Wildcard: string; const List: TStrings;
28033:                      IncludeDirs: Boolean = True; RelativeNames: Boolean = False): Boolean;
28034:
28035:  Function LongToShortFilePath( const LongName : string) : string');
28036:  //Function IsDirectory( const DirName : string) : Boolean');
28037:  Function FloatToInt( const F : Double) : Int64');
28038:  Function RFC1123DateStamp : string');
28039:  Function NowGMT : TDateTime');
28040:  Function ParseSQLDateTime( const SQLDateTime : string) : TDateTime');
28041:  Function TryParseSQLDateTime( const SQLDateTime : string; out Value : TDateTime) : Boolean');
28042:  Procedure GetInterface( const Instance : IInterface; const IID : TGUID; out Intf)');
28043:  Function IsBaseFileName( const FileName : string) : Boolean');
28044:  Procedure SetPause( const ADelay : Cardinal)');
28045:  Function URIBaseName( const URI : string) : string');
28046:  Procedure StrSliceRight(URI, Length(URI) - LastSlashPos);
28047:  Function CharInSet4(C: Char; const CharSet: CharSet):Bool;
28048:  Function TryStrToCardinal( const S : string; out Value : Cardinal) : Boolean');
28049:  Function IsEqualBytes0( const BA1, BA2: TBytes; const Count:Cardinal): Boolean;');
28050:  Function IsEqualBytes1( const BA1, BA2: TBytes) : Boolean;');
28051:  Function TryStrToWord( const S : string; out W : Word) : Boolean');
28052:
28053:  * RIRegister_TCompilerRunner Implements a class that executes a compiler and captures its output and exit
28054:  * code. Also provides specialised exception object that captures information
28055:  * about errors.  depends on TConsoleApp Class
28056:
28057: procedure SIRegister_TCompilerRunner(CL: TPSPascalCompiler);
28058: begin
28059:   //with RegClassS(CL,'TObject', 'TCompilerRunner') do
28060:   with CL.AddClassN(CL.FindClass('TObject'),'TCompilerRunner') do begin
28061:     Function Execute( const CommandLine,CurrentPath: string; const OutStream:TStream) : LongWord');
28062:   end;
28063: end;
28064:   hStream:= TStringStream.Create('');
28065:    with TCompilerRunner.create do begin
28066:       //writeln(itoa(errorcode));
28067:       Execute('cmd /c dir *.*', exepath, hstream)
28068:       writeln(hstream.datastring);
28069:      free
28070:    end;
28071:
28072: (*----------------------------------------------------------------------------*)
28073: procedure SIRegister_TConsoleApp(CL: TPSPascalCompiler);
28074: begin
28075:   //with RegClassS(CL,'TObject', 'TConsoleApp') do
28076:   with CL.AddClassN(CL.FindClass('TObject'),'TConsoleApp') do begin
28077:     Constructor Create');
28078:     Function Execute( const CmdLine, CurrentDir : string) : Boolean');
28079:     StdIn', 'THandle', iptrw);
28080:     StdOut', 'THandle', iptrw);
28081:     StdErr', 'THandle', iptrw);
28082:     Visible', 'Boolean', iptrw);
28083:     MaxExecTime', 'LongWord', iptrw);
28084:     TimeSlice', 'Integer', iptrw);
28085:     ProcessHandle', 'THandle', iptr);
28086:     ExitCode', 'LongWord', iptr);
28087:     ErrorCode', 'DWORD', iptr);
28088:     ErrorMessage', 'string', iptr);
28089:     OnWork', 'TNotifyEvent', iptrw);
28090:   end;
28091: end;
28092:
28093: procedure SIRegister_TPipe(CL: TPSPascalCompiler);
28094: begin
28095:   //with RegClassS(CL,'TObject', 'TPipe') do
28096:   with CL.AddClassN(CL.FindClass('TObject'),'TPipe') do begin
28097:     Constructor Create( const Size : LongWord)');
28098:     Procedure Free;');
28099:     Function AvailableDataSize : LongWord');
28100:     Function ReadData( var Buf : string; const BufSize : LongWord; var BytesRead : LongWord) : Boolean');
28101:     Procedure CopyToStream( const Stm : TStream; Count : LongWord)');
28102:     Procedure CopyFromStream( const Stm : TStream; Count : LongWord)');
28103:     Function WriteData( const Buf : string; const BufSize : LongWord) : LongWord');
28104:     Procedure CloseWriteHandle');
```

```
28105:     RegisterProperty('ReadHandle', 'THandle', iptr);
28106:     WriteHandle', 'THandle', iptr);
28107:   end;
28108: end;
28109:
28110: Exception Hierarchy
28111: CL.AddClassN(CL.FindClass('TOBJECT'),'TClass');
28112: CL.AddClassN(CL.FindClass('Exception'),'EAbort');
28113: SIRegister_EHeapException(CL);
28114: CL.AddClassN(CL.FindClass('EHeapException'),'EOutOfMemory');
28115: SIRegister_EInOutError(CL);
28116: //CL.AddTypeS('PExceptionRecord', '^TExceptionRecord // will not work');
28117: SIRegister_EExternal(CL);
28118:   EIntError = class(EExternal);
28119:     EDivByZero = class(EIntError);
28120:     ERangeError = class(EIntError);
28121:     EIntOverflow = class(EIntError);
28122:   EMathError = class(EExternal);
28123:     EInvalidOp = class(EMathError);
28124:     EZeroDivide = class(EMathError);
28125:     EOverflow = class(EMathError);
28126:     EUnderflow = class(EMathError);
28127:   EInvalidPointer = class(EHeapException);
28128:   EInvalidCast = class(Exception);
28129:   EConvertError = class(Exception);
28130:     EAccessViolation = class(EExternal);
28131:       EPrivilege = class(EExternal);
28132:       EStackOverflow = class(EExternal)
28133: end deprecated;
28134:
28135: procedure SIRegister_IHttpConnection2(CL: TPSPascalCompiler);
28136: begin
28137:   //with RegInterfaceS(CL,'IUNKNOWN', 'IHttpConnection') do
28138:   with CL.AddInterface(Cl.FindInterface('IUnknown'),StringToGuid('{B9611100-5243-4874-A777-D91448517116}'),
28139:                                                    'IHttpConnection2') do
28140:   //with CL.AddInterface(CL.FindInterface('IUNKNOWN'),IHttpConnection2, 'IHttpConnection2') do
28141:   begin
28142:
28143: (*----------------------------------------------------------------------------*)
28144: procedure SIRegister_HTTPSender(CL: TPSPascalCompiler);
28145: begin
28146:  CL.AddConstantN('__ABOUT__','String').SetString( '(c) Z.Razor 20.05.2013 mX4 2022');
28147:   CL.AddTypeS('THTTPMethodsend', '( hmGetsend, hmPutsend, hmPostsend, hmDeletesend, hmHeadsend )');
28148:   CL.AddTypeS('THTTPCookie', 'record Domain : String; Name : String; Value : St'
28149:    +'ring; Expires : String; Path : String; HTTPOnly : boolean; end');
28150:   THTTPCookieArray', 'array of THTTPCookie');
28151:   SIRegister_THTTPCookieCollection(CL);
28152:   THTTPResponsesend', 'record StatusCode : Integer; StatusText : Strin'
28153:    +'g; RawHeaders : String; ContentLength: Integer; ContentEncoding: String;'
28154:    +' Location : String; Expires : String; end');
28155:   SIRegister_THTTPHeaders(CL);
28156:   SIRegister_THTTPBasicAuth(CL);
28157:   CL.AddTypeS('TCookieAddEvent', 'Procedure ( Sender : TObject; Cookie : THTTPCookie)');
28158:   TWorkBeginEventsend', 'Procedure ( Sender : TObject; WorkCountMax : int64)');
28159:   TWorkEventsend', 'Procedure ( Sender : TObject; WorkCount : int64)');
28160:   TWorkEndEventsend', 'Procedure ( Sender : TObject)');
28161:   THTTPPostContainerFormField', 'record Name : ansistring; Value : ansistring; end');
28162:   THTTPPostContainerFormFieldArray', 'array of THTTPPostContainerFormField');
28163:   THTTPPostContainerFile', 'record Name : ansistring; FileName : a'
28164:    +'nsistring; ContentType : ansistring; end');
28165:   CL.AddTypeS('THTTPPostContainerFileArray', 'array of THTTPPostContainerFile');
28166:   SIRegister_THTTPPostContainer(CL);
28167:   SIRegister_THTTPSender(CL);
28168:  Function __HTTPEncode( const Text : String) : String');
28169:  Function __HTTPDecode( const Text : String) : String');
28170:  Function __HTMLDecode( const Text : String) : String');
28171:  //CL.AddDelphiFunction('Procedure Register');
28172: end;
28173:
28174: procedure SIRegister_THTTPSender(CL: TPSPascalCompiler);
28175: begin
28176:   //with RegClassS(CL,'TComponent', 'THTTPSender') do
28177:   with CL.AddClassN(CL.FindClass('TComponent'),'THTTPSender') do begin
28178:     RegisterProperty('DefaultEncoding', 'TEncoding', iptrw);
28179:     Response', 'THTTPResponse', iptr);
28180:     ResponseText', 'ansistring', iptr);
28181:     Function Get( const URL : String) : String;');
28182:     Function Post( const URL : String; PostData : ansistring) : String;');
28183:     Function Post1( const URL : String; PostContainer : THTTPPostContainer) : String;');
28184:     Function Post2( const URL : String; PostData : TStringList) : String;');
28185:     Function Put( const URL : String) : String;');
28186:     Procedure Get1( const URL : String; Stream : TStream);');
28187:     Procedure Post3( const URL : String; PostData : ansistring; Stream : TStream);');
28188:     Procedure Post4( const URL : String; PostData : TStringList; Stream : TStream);');
28189:     Procedure Post5( const URL : String; PostContainer : THTTPPostContainer; Stream : TStream);');
28190:     Procedure Put1( const URL : String; Stream : TStream);');
28191:     Procedure Free');
28192:     Constructor Create( AOwner : TComponent)');
28193:     RegisterProperty('Cookies', 'THTTPCookieCollection', iptrw);
```

```
28194:     Proxy', 'String', iptrw);
28195:     ProxyBypass', 'String', iptrw);
28196:     AllowCookies', 'boolean', iptrw);
28197:     AutoRedirects', 'boolean', iptrw);
28198:     ConnectTimeout', 'Integer', iptrw);
28199:     ReadTimeout', 'Integer', iptrw);
28200:     SendTimeout', 'Integer', iptrw);
28201:     UseIECookies', 'boolean', iptrw);
28202:     Headers', 'THTTPHeaders', iptrw);
28203:     BasicAuth', 'THTTPBasicAuth', iptrw);
28204:     OnCookieAdd', 'TCookieAddEvent', iptrw);
28205:     OnWorkBegin', 'TWorkBeginEventsend', iptrw);
28206:     OnWork', 'TWorkEventsend', iptrw);
28207:     OnWorkEnd', 'TWorkEndEventsend', iptrw);
28208:     About', 'String', iptr);
28209:   end;
28210: end;
28211:
28212: Recompiled
28213: C:\maXbox>dir /s /o N *.dcu | find "15/12/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
28214:
28215: 15/12/2022  14:50           485,115 fMain.dcu
28216: 15/12/2022  15:10            71,781 ALHttpClient2.dcu
28217: 15/12/2022  15:10            13,520 BlocksUnit.dcu
28218: 15/12/2022  12:12             2,593 DataSetUtils.dcu
28219: 15/12/2022  13:48             9,213 HttpConnection.dcu
28220: 15/12/2022  15:19            36,445 HTTPSender.dcu
28221: 15/12/2022  15:45            30,863 MathsLib.dcu
28222: 15/12/2022  15:10             6,568 PXLTiming.dcu
28223: 15/12/2022  15:10             7,434 PythonAction.dcu
28224: 15/12/2022  15:10           278,943 PythonEngine.dcu
28225: 15/12/2022  15:10             6,446 SeSHA256.dcu
28226: 15/12/2022  15:10            66,632 uPSI_ALHttpClient2.dcu
28227: 15/12/2022  12:17            69,868 uPSI_AzuliaUtils.dcu
28228: 15/12/2022  15:10            20,494 uPSI_BlocksUnit.dcu
28229: 15/12/2022  12:17             3,523 uPSI_DataSetUtils.dcu
28230: 15/12/2022  14:16            10,563 uPSI_HttpConnection.dcu
28231: 15/12/2022  14:10            20,018 uPSI_HttpConnectionWinInet.dcu
28232: 15/12/2022  15:30            25,313 uPSI_HTTPSender.dcu
28233: 15/12/2022  15:10             8,645 uPSI_PXLTiming.dcu
28234: 15/12/2022  15:10           175,123 uPSI_PythonEngine.dcu
28235: 15/12/2022  15:10             9,511 uPSI_VelthuisFloatUtils.dcu
28236: 15/12/2022  15:10            46,228 VarPyth.dcu
28237: 15/12/2022  15:10             5,682 VelthuisFloatUtils.dcu
28238:
28239: Recompiled Dez. 4.7.6.20
28240:
28241: 07/10/2022  21:16            16,829 FindReplDlg.dcu
28242: 07/10/2022  20:53           483,995 fMain.dcu
28243: 07/10/2022  21:16            35,172 gsUtils.dcu
28244: 07/10/2022  21:16           194,742 IFSI_WinForm1puzzle.dcu
28245: 07/10/2022  21:16           145,651 uPSI_fMain.dcu
28246: 07/10/2022  21:16            10,776 uPSI_IdPOP3.dcu
28247: 07/10/2022  21:16             5,649 uPSI_JvParsing.dcu
28248: 07/10/2022  21:16            21,122 uPSI_uTPLb_AES.dcu
28249: 07/10/2022  21:16            25,457 VListView.dcu
28250: 07/10/2022  21:16            70,846 WinForm1.dcu
28251: 07/10/2022  17:59            82,034 uPSI_Chart.dcu
28252: 07/10/2022  21:16            11,936 uPSI_SynEditMiscProcs.dcu
28253: 07/10/2022  11:08            10,972 uPSI_SynHighlighterPas.dcu
28254: 07/10/2022  17:44           100,843 uPSI_TeCanvas.dcu
28255: 07/10/2022  18:28           177,508 uPSI_TeEngine.dcu
28256: 07/10/2022  17:52            74,924 uPSI_TeeProcs.dcu
28257: 07/10/2022  21:16           105,297 AdoMain.dcu
28258: 07/10/2022  21:16            71,777 ALHttpClient2.dcu
28259: 07/10/2022  21:16            13,508 BlocksUnit.dcu
28260: 07/10/2022  21:16            11,799 frmExportMain.dcu
28261: 07/10/2022  21:16            30,871 MathsLib.dcu
28262: 07/10/2022  21:16            37,441 neuraldatasets.dcu
28263: 07/10/2022  21:16            50,133 neuralfit.dcu
28264: 07/10/2022  21:16           317,297 neuralnetworkCAI.dcu
28265: 07/10/2022  21:16           114,799 neuralvolume.dcu
28266: 07/10/2022  21:16            11,119 PSResources.dcu
28267: 07/10/2022  21:16             6,569 PXLTiming.dcu
28268: 07/10/2022  21:16             7,434 PythonAction.dcu
28269: 07/10/2022  21:16           278,943 PythonEngine.dcu
28270: 07/10/2022  21:16             6,447 SeSHA256.dcu
28271: 07/10/2022  21:16             3,879 simplecomport.dcu
28272: 07/10/2022  21:16            11,331 U_Splines.dcu
28273: 07/10/2022  21:16            66,636 uPSI_ALHttpClient2.dcu
28274: 07/10/2022  21:16            20,494 uPSI_BlocksUnit.dcu
28275: 07/10/2022  21:16            16,019 uPSI_DFFUtils.dcu
28276: 07/10/2022  11:08            12,554 uPSI_frmExportMain.dcu
28277: 07/10/2022  21:16            25,351 uPSI_JclSysUtils.dcu
28278: 07/10/2022  21:16            24,307 uPSI_neuraldatasets.dcu
28279: 07/10/2022  20:55             6,188 uPSI_PSResources.dcu
28280: 07/10/2022  21:16             8,644 uPSI_PXLTiming.dcu
28281: 07/10/2022  21:16           175,125 uPSI_PythonEngine.dcu
28282: 07/10/2022  21:16            46,231 VarPyth.dcu
```

```
28283: 07/10/2022  21:16            19,484 ViewToDoFm.dcu
28284: 07/10/2022  11:08            28,736 uconvMain.dcu
28285: 07/10/2022  16:52            63,798 uPSI_Series.dcu
28286: 07/10/2022  11:08            19,799 uPSI_uconvMain.dcu
28287: 07/10/2022  11:45            47,688 SynHighlighterPas.dcu
28288:
28289: 08/10/2022  00:43           484,364 fMain.dcu
28290: 08/10/2022  13:00            82,689 uPSI_Chart.dcu
28291: 08/10/2022  12:22           102,809 uPSI_TeCanvas.dcu
28292: 08/10/2022  12:46            75,065 uPSI_TeeProcs.dcu
28293: 08/10/2022  00:44            71,779 ALHttpClient2.dcu
28294: 08/10/2022  00:44            13,517 BlocksUnit.dcu
28295: 08/10/2022  16:05            30,866 MathsLib.dcu
28296: 08/10/2022  14:53            19,247 process.dcu
28297: 08/10/2022  16:03            13,555 PSResources.dcu
28298: 08/10/2022  00:44             6,568 PXLTiming.dcu
28299: 08/10/2022  00:44             7,433 PythonAction.dcu
28300: 08/10/2022  00:44           278,838 PythonEngine.dcu
28301: 08/10/2022  00:44             6,446 SeSHA256.dcu
28302: 08/10/2022  00:44            66,636 uPSI_ALHttpClient2.dcu
28303: 08/10/2022  00:33            37,434 uPSI_AzuliaUtils.dcu
28304: 08/10/2022  00:44            20,494 uPSI_BlocksUnit.dcu
28305: 08/10/2022  14:50            17,956 uPSI_process.dcu
28306: 08/10/2022  14:53             6,869 uPSI_PSResources.dcu
28307: 08/10/2022  00:44             8,643 uPSI_PXLTiming.dcu
28308: 08/10/2022  00:44           175,123 uPSI_PythonEngine.dcu
28309: 08/10/2022  00:44            46,252 VarPyth.dcu
28310: 08/10/2022  14:50            22,801 W32VersionInfo.dcu
28311:
28312: 09/10/2022  16:35           211,349 IFSI_SysUtils_max.dcu
28313: 09/10/2022  16:54           195,259 IFSI_WinForm1puzzle.dcu
28314: 09/10/2022  19:16            30,861 MathsLib.dcu
28315: 09/10/2022  19:14            14,498 PSResources.dcu
28316: 09/10/2022  17:54            48,542 uPSI_AzuliaUtils.dcu
28317: 09/10/2022  18:12             7,017 uPSI_PSResources.dcu
28318:
28319: 13/10/2022  16:31            82,689 uPSI_Chart.dcu
28320: 13/10/2022  18:24           178,625 uPSI_TeEngine.dcu
28321: 13/10/2022  16:36            75,069 uPSI_TeeProcs.dcu
28322: 13/10/2022  19:00            30,862 MathsLib.dcu
28323: 13/10/2022  17:01            49,467 uPSI_AzuliaUtils.dcu
28324:
28325: 15/10/2022  12:08           348,647 TeEngine.dcu
28326: 15/10/2022  12:05           179,899 uPSI_TeEngine.dcu
28327: 15/10/2022  17:57            30,859 MathsLib.dcu
28328: 15/10/2022  17:53            59,779 neuralfit.dcu
28329: 15/10/2022  13:45            53,663 uPSI_AzuliaUtils.dcu
28330: 15/10/2022  16:33            29,161 uPSI_neuralfit.dcu
28331:
28332: 16/10/2022  17:27           484,338 fMain.dcu
28333: 16/10/2022  17:30            71,766 ALHttpClient2.dcu
28334: 16/10/2022  17:30            13,517 BlocksUnit.dcu
28335: 16/10/2022  17:25            59,939 neuralfit.dcu
28336: 16/10/2022  17:30             6,566 PXLTiming.dcu
28337: 16/10/2022  17:30             7,427 PythonAction.dcu
28338: 16/10/2022  17:30           278,951 PythonEngine.dcu
28339: 16/10/2022  17:30             6,446 SeSHA256.dcu
28340: 16/10/2022  17:30            66,636 uPSI_ALHttpClient2.dcu
28341: 16/10/2022  17:30            20,495 uPSI_BlocksUnit.dcu
28342: 16/10/2022  17:30             8,645 uPSI_PXLTiming.dcu
28343: 16/10/2022  17:30           175,131 uPSI_PythonEngine.dcu
28344: 16/10/2022  17:30            46,237 VarPyth.dcu
28345:
28346: 17/10/2022  11:11            49,666 JclNTFS2.dcu
28347: 17/10/2022  11:27            30,846 MathsLib.dcu
28348: 17/10/2022  11:26            56,533 uPSI_AzuliaUtils.dcu
28349: 17/10/2022  11:16            47,572 uPSI_JclNTFS2.dcu
28350:
28351: C:\maXbox>dir /s /o N *.dcu | find "23/09/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
28352: 23/09/2022  16:15           483,806 fMain.dcu
28353: 23/09/2022  10:00           194,750 IFSI_WinForm1puzzle.dcu
28354: 23/09/2022  12:43            81,932 uPSI_Chart.dcu
28355: 23/09/2022  11:39            97,061 uPSI_TeCanvas.dcu
28356: 23/09/2022  11:39           177,404 uPSI_TeEngine.dcu
28357: 23/09/2022  11:20            73,888 uPSI_TeeProcs.dcu
28358: 23/09/2022  16:16            71,782 ALHttpClient2.dcu
28359: 23/09/2022  16:16            13,522 BlocksUnit.dcu
28360: 23/09/2022  17:45            30,862 MathsLib.dcu
28361: 23/09/2022  16:16             6,568 PXLTiming.dcu
28362: 23/09/2022  16:16             7,434 PythonAction.dcu
28363: 23/09/2022  16:16           278,867 PythonEngine.dcu
28364: 23/09/2022  16:06            10,517 RestUtils.dcu
28365: 23/09/2022  16:16             6,446 SeSHA256.dcu
28366: 23/09/2022  16:16            66,632 uPSI_ALHttpClient2.dcu
28367: 23/09/2022  17:45            31,846 uPSI_AzuliaUtils.dcu
28368: 23/09/2022  16:16            20,494 uPSI_BlocksUnit.dcu
28369: 23/09/2022  16:16             8,642 uPSI_PXLTiming.dcu
28370: 23/09/2022  16:16           175,123 uPSI_PythonEngine.dcu
28371: 23/09/2022  16:15            11,240 uPSI_RestUtils.dcu
```

```
28372: 23/09/2022  16:16            46,235 VarPyth.dcu
28373: 23/09/2022  15:11            63,422 uPSI_Series.dcu
28374:
28375: ****************************************************************
28376: Release Notes maXbox 4.7.6.20 Dez 2022 mX476
28377: ***************************************************************
28378: Add 75 Units + 21 Tutorials
28379:
28380: 1441 unit uPSI_neuralgeneric.pas; CAI
28381: 1442 unit uPSI_neuralthread.pas; CAI
28382: 1443 unit uPSI_uSysTools; TuO
28383: 1444 unit upsi_neuralsets; mX4
28384: 1445 unit uPSI_uWinNT.pas mX4
28385: 1446 unit uPSI_URungeKutta4.pas ICS
28386: 1447 unit uPSI_UrlConIcs.pas ICS
28387: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
28388: 1449 unit uPSI_Numedit2 mX4
28389: 1450 unit uPSI_PsAPI_3.pas mX4
28390: 1451 unit uPSI_SeSHA256.pas
28391: 1452 unit IdHashMessageDigest_max2;
28392: 1453 unit uPSI_BlocksUnit.pas
28393: 1454 unit uPSI_DelticsCommandLine.pas
28394: 1455 unit uPSI_DelticsStrUtils;
28395: 1456 unit uPSI_DelticsBitField;
28396: 1457 unit uPSI_DelticsSysUtils;
28397: 1458 unit uPSI_ALIniFiles2.pas
28398: 1459 unit uPSI_StarCalc2.pas
28399: 1460 unit uPSI_IdHashMessageDigest2.pas
28400: 1461 unit uPSI_U_Splines;
28401: 1462 unit uPSI_U_CoasterB.pas;
28402: 1463 U_SpringMass2.pas
28403: 1464 uPSI_MARSCoreUtils;
28404: 1465 unit uPSI_clJsonParser.pas
28405: 1466 unit uPSI_SynHighlighterPython.pas
28406: 1467 unit uPSI_DudsCommonDelphi;
28407: 1468 unit uPSI_AINNNeuron;
28408: 1469 unit uPSI_PJConsoleApp2;
28409: 1470 unit uPSI_PJPipeFilters2;
28410: 1471 unit uPSI_uHTMLBuilder;
28411: 1472 unit uPSI_PJPipe2;
28412: 1473 uPSI_WinApiDownload,
28413: 1474 uPSI_pxQRcode,    //beta
28414: 1475 unit uPSI_neuralplanbuilder2
28415: 1476 unit uPSI_DelphiZXingQRCode;
28416: 1477 unit uPSI_RestJsonUtils;
28417: 1478 unit UtilsTimeCode;
28418: 1479 unit uPSC_classes2.pas; //TList
28419: 1480 unit uPSC_std2.pas
28420: 1481 unit uPSI_maxIniFiles.pas
28421: 1482 unit uROPSImports.pas
28422: 1483 unit uROPSServerLink.pas
28423: 1484 unit uPSI_KLibUtils;
28424: 1485 unit uPSI_PathFunc2;  //inno setup
28425: 1486 unit KLibVC_Redist.pas;
28426: 1487 unit HTTPApp2.pas;
28427: 1488 unit uPSI_XCollection2;
28428: 1489 unit uPSI_KLibWindows;
28429: 1490 unit KlibConstants;
28430: 1491 unit uPSI_AzuliaUtils.pas
28431: 1492 unit uPSI_ALHttpClient2;
28432: 1493 unit uPSI_ALWininetHttpClient2;
28433: 1494 unit uPSI_UtilsMax41.pas
28434: 1495 unit uPSI_JclSysUtils1;
28435: 1496 unit uPSI_RestUtils;
28436: 1497 unit uPSI_TeEngine2.pas
28437: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
28438: 1499 unit uPSI_PSResources.pas
28439: 1500 unit uPSI_TeCanvas2_1.pas
28440: 1501 unit uPSI_DataSetConverter4DUtil;
28441: 1502 unit uPSI_neuralfit2.pas;
28442: 1503 unit uPSI_SynCrtSock.pas
28443: 1504 uPSI_RunElevatedSupport.pas
28444: 1505 unit synTHttpRequest.pas;
28445: 1506 unit uPSI_VelthuisFloatUtils.pas
28446: 1507 unit HttpConnection.pas
28447: 1508 unit uPSI_HttpConnectionWinInet.pas
28448: 1509 unit UHexUtils.pas
28449: 1510 unit UExeFileType.pas
28450: 1511 unit uPSI_UConsoleApp.pas
28451: 1512 unit uPSI_CompilersURunner.pas
28452: 1513 unit uPSI_HttpConnection.pas
28453: 1514 unit uPSI_DataSetUtils.pas
28454: 1515 unit uPSI_HTTPSender.pas
28455:
28456: Total of Function Calls: 36205
28457: SHA1: 4.7.6.20 BF042A1DC1C479C706100BB83F8DEE351554F74B
28458: CRC32: 4A7231F8 32.4 MB (34,057,496 bytes)
28459: Compilation Timestamp 2022-12-15 14:45:59 UTC Signing time 15 Dec 2022 15:50:51
28460: Entry Point 24930856 - Contained Sections 10
```

```
28461:
28462: Amount of Functions: 21460
28463: Amount of Procedures: 12722
28464: Amount of Constructors: 2009
28465: Amount of Destructors: 14
28466: Totals of Calls: 36205
28467: SHA1: of 4.7.6.20 BF042A1DC1C479C706100BB83F8DEE351554F74B
28468: CRC32: 4A7231F8:  34057496 bytes
28469: ▯▯▯ mX4 executed: 15/12/2022 16:50:31  Runtime: 0:15:18.987  Memload: 41% use
28470:
28471: Total of Function Calls: 36205
28472: SHA1: 4.7.6.20 BF042A1DC1C479C706100BB83F8DEE351554F74B
28473: CRC32: 4A7231F8 32.4 MB (34,057,496 bytes)
28474: Compilation Timestamp 2022-12-15 14:45:59 UTC Signing time 15 Dec 2022 15:50:51
28475: Entry Point 24930856 - Contained Sections 10
28476: ---------------------------------------------------------------
28477:
28478: Total of Function Calls: 35958
28479: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
28480: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
28481: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
28482: Entry Point 24783132 - Contained Sections 10
28483: Amount of Functions: 21314
28484: Amount of Procedures: 12650
28485: Amount of Constructors: 1980
28486: Amount of Destructors: 14
28487: Totals of Calls: 35958
28488: SHA1: of 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
28489: CRC32: 9E995FA2:  33805592 bytes
28490: ▯▯▯ mX4 executed: 17/10/2022 12:23:45  Runtime: 0:14:16.719  Memload: 36% use
28491:
28492: ****************************************************************
28493: Release Notes maXbox 4.7.6.10 II November 2021 mX476
28494: ****************************************************************
28495: Add 10 Units + 3 Tutorials
28496:
28497: 1441 unit uPSI_neuralgeneric.pas; CAI
28498: 1442 unit uPSI_neuralthread.pas; CAI
28499: 1443 unit uPSI_uSysTools; TuO
28500: 1444 unit upsi_neuralsets; mX4
28501: 1445 unit uPSI_uWinNT.pas mX4
28502: 1446 unit uPSI_URungeKutta4.pas ICS
28503: 1447 unit uPSI_UrlConIcs.pas ICS
28504: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
28505: 1449 unit uPSI_Numedit2 mX4
28506: 1450 unit uPSI_PsAPI_3.pas mX4
28507:
28508: Total of Function Calls: 35078
28509: SHA1: of 4.7.6.10  D4B0A36E42E9E89642A140CCEE2B7CCDDE3D041A
28510: CRC32: B8F2450F 30.6 MB (32,101,704 bytes)
28511:
28512: Amount of Functions: 20591
28513: Amount of Procedures: 12332
28514: Amount of Constructors: 1943
28515: Amount of Destructors: 14
28516: Totals of Calls: 34880
28517: SHA1: of 4.7.6.10 CF939E3A8D4723DB1DEF383C5FC961E06728C58F
28518: CRC32: 38F88218:  32022344 bytes
28519: ▯▯▯ mX4 executed: 14/11/2021 11:02:18  Runtime: 0:12:36.968  Memload: 42% use
28520:
28521: ****************************************************************
28522: Tech Notes maXbox 4.7.5.90 V October 2021 mX47
28523: ****************************************************************
28524: add uPSI_flcFloats.pas to complete Fundamentals5
28525: Registermethod: count, capacity in TNNetList
28526:  with CL.Add(TNNetList) do RegisterMethod(@TNNetList.Count, 'Count');
28527: alias flots() - floattostr()
28528: Function flots( const A : Float) : String');
28529: plus resources MOON_FULL, maxbox4logo.jpg, EARTH.bmp
28530:  Bugfix TSize Type
28531:  CL.AddClassN(CL.FindClass('TOBJECT'),'TInteger'); Bigints for BigFloat
28532:  - RegisterMethod(FindClass('TObject'),'TBigFloat'),'Procedure Assign4( A : TInteger);
28533: fix type TSize, add type TIntegerArray
28534:   CL.AddTypeS('TSize', 'record cx: Longint; cy: Longint; end');   //not anymore in JvVCLUtils
28535:   asize:= MakeSize(2,34);
28536:   wGetTextExtentPoint32(aDC, 'f' ,4, asize)
28537:   writeln(botostr(GetViewportExtEx( adc, asize)));
28538: Function AddLayer( pLayer : TNNetLayer) : TNNetLayer;
28539: Function AddLayer1( strData : string) : TNNetLayer;
28540: Function AddLayer2( pLayers : array of TNNetLayer) : TNNetLayer;
28541: TestConvolutionAPI Routine internal
28542:
28543: Docu of main Neural Unit for maXbox: http://www.softwareschule.ch/examples/uPSI_NeuralNetworkCAI.txt
28544: http://www.softwareschule.ch/examples/uPSI_neuralnetworkcai.txt
28545: http://www.softwareschule.ch/examples/uPSI_neuralvolume.txt
28546: http://www.softwareschule.ch/examples/uPSI_neuraldatasets.txt
28547: http://www.softwareschule.ch/examples/uPSI_neuralfit.txt
28548: Source https://github.com/joaopauloschuler/neural-api
28549:
```

```
28550: ******************************************************************************
28551: Procedure
28552: Proc SIZE 12456
28553: Proc ****************************Now the Proc list*******************
28554: Procedure( ACol, ARow : Int; Items : TStrings)
28555: Procedure( Agg : TAggregate)
28556: Procedure( ASender : TComponent; const AReplyStatus : TReplyStatus)
28557: Procedure( ASender : TComponent; const AString :Str; var AMsg : TIdMessage)
28558: Procedure( ASender : TComponent; var AMsg : TIdMessage)
28559: Procedure( ASender : TObject; const ABytes : Int)
28560: Procedure( ASender : TObject; VStream : TStream)
28561: Procedure( AThread : TIdThread)
28562: Procedure( AWebModule : TComponent)
28563: Procedure( Column : TColumn)
28564: Procedure(const AUsername:Str; const APassword:Str; AAuthenticationResult: Bool)
28565: Procedure( const iStart : Int; const sText :Str)
28566: Procedure(Control:TCustomTabControl; TabIndex:Int; const Rect : TRect; Active :Bool)
28567: Procedure( Database : TDatabase; LoginParams : TStrings)
28568: Proc DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:TReconcileAction)
28569: Proc ( DATASET : TDATASET)
28570: Proc ( DataSet:TDataSet;E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
28571: Procedure( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
28572: Procedure( DataSet : TDataSet; UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction)
28573: Procedure( DATASET : TDATASET; var ACCEPT :Bool)
28574: Procedure( DBCtrlGrid : TDBCtrlGrid; Index : Int)
28575: Procedure( Done : Int)
28576: Procedure( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
28577: Procedure(HeaderControl:TCustomHeaderControl;Sect:THeaderSection;const Rect:TRect;Pressed:Bool)
28578: Procedure(HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
28579: Procedure( HeaderControl : THeaderControl; Section : THeaderSection)
28580: Procedure(HeaderControl:THeaderControl;Section:THeaderSection;const Rect:TRect;Pressed:Bool)
28581: Procedure(HeaderControl:THeaderControl;Sectin:THeaderSection;Width:Int;State:TSectionTrackState)
28582: Procedure(Sender:TCustomListView;const ARect:TRect;Stage:TCustomDrawStage;var DefaultDraw: Bool)
28583: Procedure(Sender: TCustomListView; const ARect : TRect; var DefaultDraw :Bool)
28584: Procedure(Sender: TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
28585: Procedure(Sender: TCustomTreeView; const ARect : TRect; var DefaultDraw :Bool)
28586: Procedure(SENDER: TFIELD; const TEXT :Str)
28587: Procedure(SENDER: TFIELD; var TEXT :Str; DISPLAYTEXT :Bool)
28588: Procedure(Sender: TIdTelnet; const Buffer :Str)
28589: Procedure(Sender: TIdTelnet; Status : TIdTelnetCommand)
28590: Procedure(SENDER: TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED :Bool)
28591: Procedure(SENDER: TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
28592: Procedure(SENDER: TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : Int)
28593: Procedure(Sender: TObject; ACol, ARow : Longint; const Value :Str)
28594: Procedure(Sender: TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
28595: Procedure(Sender: TObject; ACol, ARow : Longint; var CanSelect :Bool)
28596: Procedure(Sender: TObject; ACol, ARow : Longint; var Value :Str)
28597: Procedure(Sender: TObject; Button : TMPBtnType)
28598: Procedure(Sender: TObject; Button : TMPBtnType; var DoDefault :Bool)
28599: Procedure(Sender: TObject; Button : TUDBtnType)
28600: Procedure(Sender: TObject; Canvas: TCanvas; PageRect:TRect; var DoneDrawing:Bool)
28601: Procedure(SenderTObject;ClientSocket:TServerClientWinSocket;var SocketThread: TServerClientThread)
28602: Procedure(Sender: TObject; Column : TListColumn)
28603: Procedure(Sender: TObject; Column : TListColumn; Point : TPoint)
28604: Procedure(Sender: TObject; Connecting :Bool)
28605: Procedure(Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
28606: Procedure(Sendr:TObject;const Rect:TRect;DataCol:Int;Column:TColumn;State:TGridDrawState)
28607: Procedure( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
28608: Procedure(Sender:TObject;const UserStr:str;var DateAndTime:TDateTime;var AllowChange:Bool)
28609: Procedure(Sender:TObject; E : Exception; var Handled :Bool)
28610: Procedure(Sender:TObject; FromIndex, ToIndex : Longint)
28611: Procedure(Sender:TObject; FromSection,ToSection: THeaderSection; var AllowDrag:Bool)
28612: Procedure(Sender:TObject; Index : LongInt)
28613: Procedure(Sender:TObject; Item : TListItem)
28614: Procedure(Sender:TObject; Item : TListItem; Change : TItemChange)
28615: Procedure(Sender:TObject;Item: TListItem; Change:TItemChange;var AllowChange:Boolean)
28616: Procedure(Sender:TObject; Item : TListItem; Selected :Bool)
28617: Procedure(Sender:TObject; Item : TListItem; var AllowEdit :Bool)
28618: Procedure(Sender:TObject; Item : TListItem; var S :Str)
28619: Procedure(Sender:TObject; Item1, Item2 : TListItem; Data : Int; var Compare : Int)
28620: Procedure(Sender:TObject; ModalResult : TModalResult; var CanClose :Bool)
28621: Procedure(Sender:TObject; Month : LongWord; var MonthBoldInfo : LongWord)
28622: Procedure(Sender:TObject; NewTab : Int; var AllowChange :Bool)
28623: Procedure(Sender:TObject; Node : TTreeNode)
28624: Procedure(Sender:TObject; Node : TTreeNode; var AllowChange :Bool)
28625: Procedure(Sender:TObject; Node : TTreeNode; var AllowCollapse :Bool)
28626: Procedure(Sender:TObject; Node : TTreeNode; var AllowEdit :Bool)
28627: Procedure(Sender:TObject; Node : TTreeNode; var AllowExpansion :Bool)
28628: Procedure(Sender:TObject; Node : TTreeNode; var S :Str)
28629: Procedure(Sender:TObject; Node1, Node2 : TTreeNode; Data : Int; var Compare : Int)
28630: Procedure(Sender : TObject; NumObjects, NumChars : Int; var SaveClipboard :Bool)
28631: Procedure(Sender : TObject; Rect : TRect)
28632: Procedure(Sender:TObject; Request:TWebRequest;Response:TWebResponse; var Handled:Bool)
28633: Procedure(Sender:TObject;Shift:TShiftState;X,Y:Int;Orient:TPageScrollerOrientation;var Delta:Int
28634: Procedure(Sender TObject; Socket : TCustomWinSocket)
28635: Procedure(Sender:TObject;Socket:TCustomWinSocket;ErrorEvent:TErrorEvent;var ErrorCode: Int)
28636: Procedure( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
28637: Procedure(Sender: TObject; Socket : TSocket; var ClientSocket: TServerClientWinSocket)
```

```
28638: Procedure(SENDER: TOBJECT; SOURCE : TMENUITEM; REBUILD :Bool)
28639: Procedure(Sender: TObject; StartPos, EndPos : Int; var AllowChange :Bool)
28640: Procedure(Sender: TObject; TabCanvas: TCanvas;R:TRect; Index: Int; Selected :Bool)
28641: Procedure(Sender: TObject; TabIndex : Int; var ImageIndex : Int)
28642: Procedure(Sender: TObject; Thread : TServerClientThread)
28643: Procedure(Sender: TObject; TickCount :Card; var Reset :Bool)
28644: Procedure(Sender: TObject; Username, Password :Str)
28645: Procedure(Sender: TObject; var AllowChange :Bool)
28646: Procedure(Sender: TObject; var AllowChange:Boolean; NewValue:SmallInt; Direction:TUpDownDirection)
28647: Procedure(Sender: TObject; var Caption :Str; var Alignment : THTMLCaptionAlignment)
28648: Procedure( Sender : TObject; var Continue :Bool)
28649: Procedure(Sender:TObject;var dest:str;var NumRedirect:Int;var Handled:bool;var VMethod:TIdHTTPMethod)
28650: Procedure(Sender : TObject; var Username :Str)
28651: Procedure(Sender : TObject; Wnd : HWND)
28652: Procedure(Sender : TToolbar; Button : TToolButton)
28653: Procedure(Sender:TToolBar;const ARect:TRect;Stage:TCustomDrawStage;var DefaultDraw:Bool)
28654: Procedure(Sender : TToolBar; const ARect : TRect; var DefaultDraw :Bool)
28655: Procedure(Sender : TToolbar; Index : Int; var Allow :Bool)
28656: Procedure(Sender : TToolbar; Index : Int; var Button : TToolButton)
28657: Procedure(StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
28658: Procedure(StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
28659: Procedure(var FieldNames:TWideStrings;SQL:WideStr;var BindAllFields:Bool;var TableName:WideStr)
28660: Procedure(var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
28661: procedure(Sender: TObject)
28662: procedure(Sender: TObject; var Done:Bool)
28663: procedure(Sender: TObject; var Key: Word; Shift: TShiftState);
28664: Proc _T(Name: tbtString; v: Variant);
28665: Proc AbandonSignalHandler( RtlSigNum : Int)
28666: Proc Abort
28667: Proc About1Click( Sender : TObject)
28668: Proc Accept( Socket : TSocket)
28669: Proc AESSymetricExecute(const plaintext, ciphertext, password:Str)
28670: Proc AESEncryptFile(const plaintext, ciphertext, password:Str)
28671: Proc AESDecryptFile(const replaintext, ciphertext, password:Str)
28672: Proc AESEncryptString(const plaintext:str;var ciphertext:str;password:Str)
28673: Proc AESDecryptString(var plaintext:str;const ciphertext:str;password:Str)
28674: Proc Add(Addend1, Addend2 : TMyBigInt)
28675: Proc ADD(const AKEY, AVALUE : VARIANT)
28676: Proc Add(const Key :Str; Value : Int)
28677: Proc ADD(const NAME, FIELDS :Str; OPTIONS : TINDEXOPTIONS)
28678: Proc ADD(FIELD : TFIELD)
28679: Proc ADD(ITEM : TMENUITEM)
28680: Proc ADD(POPUP : TPOPUPMENU)
28681: Proc AddCharacters( xCharacters : TCharSet)
28682: Proc AddDriver( const Name :Str; List : TStrings)
28683: Proc AddImages( Value : TCustomImageList)
28684: Proc AddIndex(const Name,Fields:str;Options:TIndexOptions;const DescFields:str)
28685: Proc AddLambdaTransitionTo( oState : TniRegularExpressionState)
28686: Proc AddLoader( Loader : TBitmapLoader)
28687: Proc ADDPARAM( VALUE : TPARAM)
28688: Proc AddPassword( const Password :Str)
28689: Proc AddStandardAlias( const Name, Path, DefaultDriver :Str)
28690: Proc AddState( oState : TniRegularExpressionState)
28691: Proc AddStrings( Strings : TStrings);
28692: Proc AddStrings(Strings: TStrings);
28693: Proc AddStrings1( Strings : TWideStrings);
28694: Proc AddStringTerm(var sString:str;const sTerm:str;const sSeparator:str)
28695: Proc AddToRecentDocs( const Filename :Str)
28696: Proc AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
28697: Proc AllFunctionsList1Click( Sender : TObject)
28698: Proc AllObjectsList1Click(Sender: TObject);
28699: Proc Allocate( AAllocateBytes : Int)
28700: Proc AllResourceList1Click(Sender: TObject);
28701: Proc AnsiAppend( var dst : Ansistr; const src : Ansistr)
28702: Proc AnsiAssign( var dst : Ansistr; var src : Ansistr)
28703: Proc AnsiDelete( var dst : Ansistr; index, count : Int)
28704: Proc AnsiFree( var s : Ansistr)
28705: Proc AnsiFromWide( var dst : Ansistr; const src : WideString)
28706: Proc AnsiInsert( var dst : Ansistr; const src : Ansistr; index : Int)
28707: Proc AnsiSetLength( var dst : Ansistr; len : Int)
28708: Proc Ansistr_to_stream( const Value : Ansistr; Destin : TStream)
28709: Proc AntiFreeze;
28710: Proc APPEND
28711: Proc Append( const S : WideString)
28712: Proc Append(S:Str);
28713: Proc AppendByte( var VBytes : TIdBytes; AByte : byte)
28714: Proc AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
28715: Proc AppendChunk( Val : OleVariant)
28716: Proc AppendData( const Data : OleVariant; HitEOF :Bool)
28717: Proc AppendStr( var Dest :Str; S :Str)
28718: Proc AppendString( var VBytes : TIdBytes; const AStr :Str; ALength : Int)
28719: Proc ApplyRange
28720: Proc Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
28721: Proc Arrange( Code : TListArrangement)
28722: Proc Assert(expr :Bool; const msg:Str);
28723: Proc Assert2(expr :Bool; const msg:Str);
28724: Proc Assign( AList : TCustomBucketList)
28725: Proc Assign( Other : TObject)
28726: Proc Assign( Source : TDragObject)
```

```
28727: Proc Assign( Source : TPersistent)
28728: Proc Assign(Source: TPersistent)
28729: Proc Assign2(mystring, mypath:Str);
28730: Proc AssignCurValues( Source : TDataSet);
28731: Proc AssignCurValues1( const CurValues : Variant);
28732: Proc ASSIGNFIELD( FIELD : TFIELD)
28733: Proc ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
28734: Proc AssignFile(var F: Text; FileName:Str)
28735: Proc AssignFile(var F: TextFile; FileName:Str)
28736: Proc AssignFileRead(var mystring, myfilename:Str);
28737: Proc AssignFileWrite(mystring, myfilename:Str);
28738: Proc AssignTo( Other : TObject)
28739: Proc AssignValues( Value : TParameters)
28740: Proc ASSIGNVALUES( VALUE : TPARAMS)
28741: Proc AssociateExtension( IconPath, ProgramName, Path, Extension :Str)
28742: Proc Base64_to_stream( const Base64 : Ansistr; Destin : TStream)
28743: Proc Base64ToVar( NatData : Pointer; const SoapData : WideString);
28744: Proc Base64ToVar1( var V : Variant; const SoapData : WideString);
28745: Proc BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
28746: Proc BcdDivide( Dividend, Divisor :Str; var bcdOut : TBcd);
28747: Proc BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
28748: Proc BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
28749: Proc BcdDivide3( const Dividend : TBcd; const Divisor :Str; var bcdOut : TBcd);
28750: Proc BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
28751: Proc BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
28752: Proc BcdMultiply2( const bcdIn : TBcd; const StringIn :Str; var bcdOut : TBcd);
28753: Proc BcdMultiply3( StringIn1, StringIn2 :Str; var bcdOut : TBcd);
28754: Proc BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
28755: Proc BcdToBytes( Value : TBcd; Bytes : array of byte)
28756: Proc Beep
28757: Proc BeepOk
28758: Proc BeepQuestion
28759: Proc BeepHand
28760: Proc BeepExclamation
28761: Proc BeepAsterisk
28762: Proc BeepInformation
28763: Proc BEGINDRAG(IMMEDIATE:BOOLEAN)
28764: Proc BeginLayout
28765: Proc BeginTimer( const Delay, Resolution :Card)
28766: Proc BeginUpdate
28767: Proc BeginUpdate;
28768: Proc BigScreen1Click(Sender: TObject);
28769: Proc BinToHex(Buffer: PChar; Text: PChar; BufSize: Int);
28770: Proc BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits :Bool);
28771: Proc BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits :Bool);
28772: Proc BitsToBooleans2( const Bits : Int; var B : TBooleanArray; AllBits :Bool);
28773: Proc BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits:Bool);
28774: Proc BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
28775: Proc BooleansToBits( var Dest : Byte; const B : TBooleanArray);
28776: Proc BooleansToBits1( var Dest : Word; const B : TBooleanArray);
28777: Proc BooleansToBits2( var Dest : Int; const B : TBooleanArray);
28778: Proc BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
28779: Proc BreakPointMenuClick( Sender : TObject)
28780: Proc BRINGTOFRONT
28781: Proc BringToFront;
28782: Proc btnBackClick( Sender : TObject)
28783: Proc btnBrowseClick( Sender : TObject)
28784: Proc BtnClick( Index : TNavigateBtn)
28785: Proc btnLargeIconsClick( Sender : TObject)
28786: Proc BuildAndSendRequest( AURI : TIdURI)
28787: Proc BuildCache
28788: Proc BurnMemory( var Buff, BuffLen : Int)
28789: Proc BurnMemoryStream( Destructo : TMemoryStream)
28790: Proc CalculateFirstSet
28791: Proc Cancel
28792: Proc CancelDrag;
28793: Proc CancelEdit
28794: Proc CANCELHINT
28795: Proc CancelRange
28796: Proc CancelUpdates
28797: Proc CancelWriteBuffer
28798: Proc Capture1(ADest:TStream;out VLineCount:Int;const ADelim:str;const AIsRFCMessage:Bool;
28799: Proc Capture2(ADest: TStrings; const ADelim:Str; const AIsRFCMessage:Boolean);
28800: Proc Capture3(ADest:TStrings;out VLineCount:Int;const ADelim:str;const AIsRFCMessage:Bool
28801: Proc CaptureScreenFormat(vname:Str; vextension:Str);
28802: Proc CaptureScreenPNG(vname:Str);
28803: Proc CardinalsToI64(var I: Int64; const LowPart, HighPart:Card);
28804: Proc CASCADE
28805: Proc CastNativeToSoap(Info:PTypeInfo;var SoapData:WideStr;NatData:Pointer;var IsNull:Bool)
28806: Proc CastSoapToVariant(SoapInfo:PTypeInfo;const SoapData:WideString;NatData:Pointer);
28807: Proc cbPathClick( Sender : TObject)
28808: Proc cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
28809: Proc cedebugAfterExecute( Sender : TPSScript)
28810: Proc cedebugBreakpoint(Sender: TObject;const FileName:str;Position,Row,Col:Card)
28811: Proc cedebugCompile( Sender : TPSScript)
28812: Proc cedebugExecute( Sender : TPSScript)
28813: Proc cedebugIdle( Sender : TObject)
28814: Proc cedebugLineInfo(Sender:TObject; const FileName:Str; Position,Row,Col:Card)
28815: Proc CenterHeight( const pc, pcParent : TControl)
```

```
28816: Proc CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
28817: Proc CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
28818: Proc Change
28819: Proc ChangeBiDiModeAlignment(var Alignment: TAlignment);
28820: Proc Changed
28821: Proc ChangeDir( const ADirName :Str)
28822: Proc ChangeDirUp
28823: Proc ChangeEntryTransitions( oNewState : TniRegularExpressionState)
28824: Proc ChangeLevelBy( Value : TChangeRange)
28825: Proc ChDir(const s:Str)
28826: Proc Check(Status: Int)
28827: Proc CheckCommonControl( CC : Int)
28828: Proc CHECKFIELDNAME( const FIELDNAME :Str)
28829: Proc CHECKFIELDNAMES( const FIELDNAMES :Str)
28830: Proc CheckForDisconnect(const ARaiseExceptionIfDisconnected:bool;const AIgnoreBuffer:bool)
28831: Proc CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected :Bool)
28832: Proc CheckToken( T : Char)
28833: Proc CheckToken(t:char)
28834: Proc CheckTokenSymbol( const S :Str)
28835: Proc CheckTokenSymbol(s:str)
28836: Proc CheckToolMenuDropdown( ToolButton : TToolButton)
28837: Proc Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
28838: Proc CIED65ToCIED50( var X, Y, Z : Extended)
28839: Proc CIELABToBGR( const Source, Target : Pointer; const Count :Card);
28840: Proc CipherFile1Click(Sender: TObject);
28841: Proc Clear;
28842: Proc Clear1Click( Sender : TObject)
28843: Proc ClearColor( Color : TColor)
28844: Proc CLEARITEM( AITEM : TMENUITEM)
28845: Proc ClearMapping
28846: Proc ClearSelection( KeepPrimary :Bool)
28847: Proc ClearWriteBuffer
28848: Proc Click
28849: Proc Close
28850: Proc closeMP3;
28851: Proc Close1Click( Sender : TObject)
28852: Proc CloseDatabase( Database : TDatabase)
28853: Proc CloseDataSets
28854: Proc CloseDialog
28855: Proc CloseFile(var F: Text);
28856: Proc Closure
28857: Proc CMYKToBGR(const Source,Target:Pointer;const BitsPerSample:Byte;Count:Card);
28858: Proc CMYKToBGR1(const C,M,Y,K, Target:Pointer;const BitsPerSample: Byte; Count:Card);
28859: Proc CodeCompletionList1Click( Sender : TObject)
28860: Proc ColEnter
28861: Proc Collapse
28862: Proc Collapse( Recurse :Bool)
28863: Proc ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
28864: Proc CommaSeparatedToStringList( AList : TStrings; const Value :Str)
28865: Proc CommitFreeAndNil( var Transaction : TDBXTransaction)
28866: Proc Compile1Click( Sender : TObject)
28867: Proc ComponentCount1Click(Sender: TObject);
28868: Proc Compress(azipfolder, azipfile:Str)
28869: Proc DeCompress(azipfolder, azipfile:Str)
28870: Proc XZip(azipfolder, azipfile:Str)
28871: Proc XUnZip(azipfolder, azipfile:Str)
28872: Proc Connect( const ATimeout : Int)
28873: Proc Connect( Socket : TSocket)
28874: Proc Console1Click(Sender: TObject);
28875: Proc Continue
28876: Proc ContinueCount( var Counter : TJclCounter)
28877: Proc CONTROLDESTROYED(CONTROL:TCONTROL)
28878: Proc ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : Int)
28879: Proc ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : Int)
28880: Proc ConvertImage(vsource, vdestination:Str);
28881: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
28882: Proc ConvertBitmap(vsource, vdestination:Str);
28883: Proc ConvertToGray(Cnv: TCanvas);
28884: Proc Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Int; Length : Int)
28885: Proc Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Int; Count : Int);
28886: Proc Copy1( Source : TBytes; Offset : Int; Buffer : TValueBuffer; Count : Int);
28887: Proc CopyBytesToHostLongWord(const Source:TIdBytes;const ASourceIdx:Int;var VDest:LongWord)
28888: Proc CopyBytesToHostWord(const ASource:TIdBytes;const ASourceIndex:Int;var VDest: Word)
28889: Proc CopyFrom( mbCopy : TMyBigInt)
28890: Proc CopyMemoryStream( Source, Destination : TMemoryStream)
28891: Proc CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
28892: Proc CopyTIdByteArray(const ASource:array of Byte; const ASourceIndex:Int; var VDest:array of Byte; const
       ADestIndex : Int; const ALength : Int)
28893: Proc CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
28894: Proc CopyTIdCardinal(const ASource:Card;var VDest:TIdBytes;const ADestIndex:Int)
28895: Proc CopyTIdInt64(const ASource: Int64;var VDest: TIdBytes; const ADestIndex : Int)
28896: Proc CopyTIdIPV6Address(const ASource:TIdIPv6Address;var VDest:TIdBytes;const ADestIdx:Int)
28897: Proc CopyTIdLongWord(const ASource:LongWord;var VDest:TIdBytes;const ADestIndex:Int)
28898: Proc CopyTIdNetworkLongWord(const ASource:LongWord;var VDest:TIdBytes;const ADestIndex:Int)
28899: Proc CopyTIdNetworkWord(const ASource:Word;var VDest:TIdBytes;const ADestIndex:Int)
28900: Proc CopyTIdString(const ASource:Str;var VDest:TIdBytes;const ADestIndex:Int;ALength:Int)
28901: Proc CopyTIdWord( const ASource: Word; var VDest: TIdBytes; const ADestIndex : Int)
28902: Proc CopyToClipboard
```

```
28903: Proc CountParts
28904: Proc CreateDataSet
28905: Proc CreateEmptyFile( const FileName :Str)
28906: Proc CreateFileFromString( const FileName, Data :Str)
28907: Proc CreateFromDelta( Source : TPacketDataSet)
28908: Proc CREATEHANDLE
28909: Proc CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
28910: Proc CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine :Str)
28911: Proc CreateProcAsUserEx(const UserDoman,UserNme,Passwo,CommandLine:str;const Environ:PChar)
28912: Proc CreateTable
28913: Proc CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
28914: Proc CSyntax1Click(Sender: TObject);
28915: Proc CurrencyToComp( Value : Currency; var Result : Comp)
28916: Proc CURSORPOSCHANGED
28917: Proc CutFirstDirectory(var S:Str)
28918: Proc DataBaseError(const Message:Str)
28919: Proc DateTimeToString( var Result :Str; Format :Str; DateTime : TDateTime);
28920: Proc DateTimeToString(var Result:Str; const Format:Str; DateTime: TDateTime)
28921: Proc DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
28922: Proc DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
28923: Proc DBIError(errorCode: Int)
28924: Proc DebugOutput( const AText :Str)
28925: Proc Debugln(DebugLOGFILE:Str; Event_message:Str);
28926: Proc DebugRun1Click( Sender : TObject)
28927: Proc Dec;
28928: Proc DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
28929: Proc DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
28930: Proc DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
28931: Proc DecodeDateMonthWeek(const AVal:TDateTime;out AYear,AMonth,AWeekOfMonth,ADayOfWeek :Word)
28932: Proc DecodeDateTime(const AVal:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMSec:Word)
28933: Proc DecodeDateWeek(const AValue: TDateTime;out AYear,AWeekOfYear,ADayOfWeek: Word)
28934: Proc DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
28935: Proc DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
28936: Proc DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
28937: Proc Decompile1Click( Sender : TObject)
28938: Proc DefaultDrawColumnCell(const Rect:TRect;DataCol:Int;Column:TColum;State:TGridDrawState)
28939: Proc DefaultDrawDataCell(const Rect: TRect; Field: TField; State: TGridDrawState)
28940: Proc DeferLayout
28941: Proc defFileread
28942: Proc DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
28943: Proc DelayMicroseconds( const MicroSeconds : Int)
28944: Proc Delete
28945: Proc Delete(const AFilename :Str)
28946: Proc Delete(const Index : Int)
28947: Proc DELETE(INDEX : Int)
28948: Proc Delete(Index : LongInt)
28949: Proc Delete(Node : TTreeNode)
28950: Proc Delete(var s: AnyString; ifrom, icount: Longint);
28951: Proc DeleteAlias( const Name :Str)
28952: Proc DeleteDriver( const Name :Str)
28953: Proc DeleteIndex( const Name :Str)
28954: Proc DeleteKey( const Section, Ident :Str)
28955: Proc DeleteRecords
28956: Proc DeleteRecords( AffectRecords : TAffectRecords)
28957: Proc DeleteString( var pStr :Str; const pDelStr :Str)
28958: Proc DeleteTable
28959: Proc DelphiSite1Click(Sender: TObject);
28960: Proc Deselect
28961: Proc Deselect( Node : TTreeNode)
28962: Proc DestroyComponents
28963: Proc DestroyHandle
28964: Proc Diff( var X : array of Double)
28965: Proc Diff(var X: array of Double);
28966: Proc DirCreate( const DirectoryName :Str);
28967: Proc DISABLEALIGN
28968: Proc DisableConstraints
28969: Proc Disconnect
28970: Proc Disconnect( Socket : TSocket)
28971: Proc Dispose
28972: Proc Dispose(P: PChar)
28973: Proc DivMod( Dividend : Int; Divisor : Word; var Result, Remainder : Word)
28974: Proc DoKey( Key : TDBCtrlGridKey)
28975: Proc DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
28976: Proc DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
28977: Proc Dormant
28978: Proc DoubleToBcd1( const AValue : Double; var bcd : TBcd);
28979: Proc DoubleToBytes( Value : Double; Bytes : array of byte)
28980: Proc DoubleToComp( Value : Double; var Result : Comp)
28981: Proc doWebCamPic(picname:Str); //eg: c:\mypic.png
28982: Proc Draw( Canvas : TCanvas; X, Y, Index : Int; Enabled :Bool);
28983: Proc Draw(X, Y: Int; Graphic: TGraphic);
28984: Proc Draw1(Canvas:TCanvas;X,Y,Indx:Int;ADrawingStyle:TDrawingStyl;AImageType:TImageType;Enabled:Bool);
28985: Proc DrawArrow(ACanvas:TCanvas; Direction:TScrollDirection;Location:TPoint;Size:Int)
28986: Proc DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Int; Shadow :Bool)
28987: Proc DrawChevron(ACanvas:TCanvas; Direction: TScrollDirection; Location:TPoint; Size : Int)
28988: Proc DrawColumnCell( const Rect : TRect; DataCol : Int; Column : TColumn; State : TGridDrawState)
28989: Proc DrawFocusRect(const Rect: TRect);
28990: Proc DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
```

```
28991: Proc DRAWMENUITEM(MENUITEM: TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE: TOWNERDRAWSTATE)
28992: Proc DrawOverlay(Canvas:TCanvas;X,Y:Int;ImageIndex:Int;Overlay:TOverlay;Enabled:Bool);
28993: Proc DrawOverlay1(Canvas:TCanvas;X,Y:Int;ImageIndex:Int;Overlay:TOverlay;ADrawingStyle:
       TDrawingStyle;AImageType:TImageType;Enabled:Bool);
28994: Proc drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: Int);
28995: Proc DrawPolyLine(const Canvas:TCanvas;var Points:TPointArray;const ClipRect: TRect)
28996: Proc DropConnections
28997: Proc DropDown
28998: Proc DumpDescription( oStrings : TStrings)
28999: Proc DumpStateTable( oStrings : TStrings)
29000: Proc EDIT
29001: Proc EditButtonClick
29002: Proc EditFont1Click( Sender : TObject)
29003: Proc Ellipse(X1, Y1, X2, Y2: Int);
29004: Proc Ellipse1( const Rect : TRect);
29005: Proc EMMS
29006: Proc Encode( ADest : TStream)
29007: Proc ENDDRAG(DROP:BOOLEAN)
29008: Proc EndEdit( Cancel :Bool)
29009: Proc EndTimer
29010: Proc EndUpdate
29011: Proc EraseSection( const Section :Str)
29012: Proc Error( const Ident :Str)
29013: Proc Error(Ident:Int)
29014: Proc ErrorFmt( const Ident :Str; const Args : array of const)
29015: Proc ErrorStr( const Message :Str)
29016: Proc ErrorStr(Message:str)
29017: Proc Exchange( Index1, Index2 : Int)
29018: Proc Exchange(Index1, Index2: Int);
29019: Proc Exec( FileName, Parameters, Directory :Str)
29020: Proc ExecProc
29021: Proc ExecSQL( UpdateKind : TUpdateKind)
29022: Proc Execute
29023: Proc Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
29024: Proc ExecuteAndWait( FileName :Str; Visibility : Int)
29025: Proc ExecuteCommand(executeFile, paramstring:Str)
29026: Proc ExecuteShell(executeFile, paramstring:Str)
29027: Proc ShellExecuteAndWait(executeFile, paramstring:Str);
29028: Proc ExitThread(ExitCode: Int); stdcall;
29029: Proc ExitProcess(ExitCode: Int); stdcall;
29030: Proc Expand( AUserName :Str; AResults : TStrings)
29031: Proc Expand( Recurse :Bool)
29032: Proc ExportClipboard1Click( Sender : TObject)
29033: Proc ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
29034: Proc ExtractContentFields( Strings : TStrings)
29035: Proc ExtractCookieFields( Strings : TStrings)
29036: Proc ExtractFields(Separators,WhiteSpace:TSysCharSet; Content:PChar; Strings : TStrings)
29037: Proc ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
29038: Proc ExtractHTTPFields(Separators,WhiteSpace: TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
29039: Proc ExtractQueryFields( Strings : TStrings)
29040: Proc FastDegToGrad
29041: Proc FastDegToRad
29042: Proc FastGradToDeg
29043: Proc FastGradToRad
29044: Proc FastRadToDeg
29045: Proc FastRadToGrad
29046: Proc FileClose( Handle : Int)
29047: Proc FileClose(handle: Int)
29048: Proc FilesFromWildcard(Dir,Mask:str;var Files:TStringList;Subdirs,ShowDirs,Multitask:Bool)
29049: Proc FileStructure( AStructure : TIdFTPDataStructure)
29050: Proc FillByte2(var X: Byte ; count: Int; value: byte)
29051: Proc FillBytes( var VBytes : TIdBytes; const ACount : Int; const AValue : Byte)
29052: Proc FillChar( Buffer : TRecordBuffer; Length : Int; value : Byte)
29053: Proc FillChar2(var X: PChar ; count: Int; value: char)
29054: Proc FillCharS(var p:Str; count: Int; value: char); //fix3.8
29055: Proc FillIPList
29056: Proc FillRect(const Rect: TRect);
29057: Proc FillTStrings( AStrings : TStrings)
29058: Proc FilterOnBookmarks( Bookmarks : array of const)
29059: Proc FinalizePackage(Module: HMODULE)
29060: Proc FindClose;
29061: Proc FindClose2(var F: TSearchRec)
29062: Proc FindMatches( const sString:str; xNotify:TniRegularExpressionMatchFoundEvent);
29063: Proc FindMatches1(const sString:str;iStart:Int;xNotify:TniRegularExpressnMatchFoundEvent);
29064: Proc FindNearest( const KeyValues : array of const)
29065: Proc FinishContext
29066: Proc FIRST
29067: Proc FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
29068: Proc FloatToDecimal(var Result:TFloatRec;const Val:extend;ValueType:TFloatValue;Precis,Decs:Int);
29069: Proc FloodFill( X, Y : Int; Color : TColor; FillStyle : TFillStyle)
29070: Proc FlushSchemaCache( const TableName :Str)
29071: Proc FmtStr(var Result:Str; const Format:Str; const Args: array of const)
29072: Proc FOCUSCONTROL(CONTROL:TWINCONTROL)
29073: Proc Form1Close( Sender : TObject; var Action : TCloseAction)
29074: Proc FormActivate( Sender : TObject)
29075: Proc FormatLn(const format:Str; const args: array of const);  //alias
29076: Proc FormClose( Sender : TObject; var Action : TCloseAction)
29077: Proc FormCreate( Sender : TObject)
```

```
29078: Proc FormDestroy( Sender : TObject)
29079: Proc FormKeyPress( Sender : TObject; var Key : Char)
29080: Proc FormOutput1Click(Sender: TObject);
29081: Proc FormToHtml( Form : TForm; Path :Str)
29082: Proc FrameRect(const Rect: TRect);
29083: Proc Frame3D(Canvas:TCanvas;var Rect:TRect;TopColor,BottomColor:TColor; Width : Int)
29084: Proc NotebookHandlesNeeded( Notebook : TNotebook)
29085: Proc Free( Buffer : TRecordBuffer)
29086: Proc Free( Buffer : TValueBuffer)
29087: Proc Free;
29088: Proc FreeAndNil(var Obj:TObject)
29089: Proc FreeImage
29090: Proc FreeMem(P: PChar; Size: Int)
29091: Proc FreeTreeData( Tree : TUpdateTree)
29092: Proc Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Int)
29093: Proc FullCollapse
29094: Proc FullExpand
29095: Proc GenerateDPB(sl: TStrings; var DPB:Str; var DPBLength: Short);  //InterBase
29096: Proc GenerateTPB(sl: TStrings; var TPB:Str; var TPBLength: Short);
29097: Proc OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
29098: Proc Get1( AURL :Str; const AResponseContent : TStream);
29099: Proc Get1( const ASourceFile :Str; ADest : TStream; AResume :Bool);
29100: Proc Get2(const ASourceFile,ADestFile:Str;const ACanOverwrite: bool; AResume: Bool);
29101: Proc GetAliasNames( List : TStrings)
29102: Proc GetAliasParams( const AliasName :Str; List : TStrings)
29103: Proc GetApplicationsRunning( Strings : TStrings)
29104: Proc getBox(aURL, extension:Str);
29105: Proc GetCommandTypes( List : TWideStrings)
29106: Proc GetConfigParams( const Path, Section :Str; List : TStrings)
29107: Proc GetConnectionNames( List : TStrings; Driver :Str; DesignMode :Bool)
29108: Proc GetConvFamilies( out AFamilies : TConvFamilyArray)
29109: Proc GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
29110: Proc GetDatabaseNames( List : TStrings)
29111: Proc GetDataPacket( DataSet : TDataSet; var RecsOut : Int; out Data : OleVariant)
29112: Proc GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize:longWORD; var ImageSize : longWORD)
29113: Proc GetDir(d: byte; var s:Str)
29114: Proc GetDirList( const Search :Str; List : TStrings; Recursive :Bool)
29115: Proc GetDriverNames( List : TStrings)
29116: Proc GetDriverNames( List : TStrings; DesignMode :Bool)
29117: Proc GetDriverParams( const DriverName :Str; List : TStrings)
29118: Proc GetEMails1Click( Sender : TObject)
29119: Proc getEnvironmentInfo;
29120: Func getEnvironmentString:Str;
29121: Proc GetFieldNames(const DatabaseName, TableName :Str; List : TStrings)
29122: Proc GetFieldNames(const TableName :Str; List : TStrings)
29123: Proc GetFieldNames(const TableName :Str; List : TStrings);
29124: Proc GetFieldNames(const TableName : WideString; List : TWideStrings);
29125: Proc GETFIELDNAMES(LIST : TSTRINGS)
29126: Proc GetFieldNames1(const TableName :Str; List : TStrings);
29127: Proc GetFieldNames1(const TableName :Str; SchemaName :Str; List : TStrings);
29128: Proc GetFieldNames2(const TableName : WideString;SchemaName : WideString; List : TWideStrings);
29129: Proc GetFieldNames3(const TableName : WideString; List : TWideStrings);
29130: Proc GetFileAttributeList( const Items : TStrings; const Attr : Int)
29131: Proc GetFileAttributeListEx( const Items : TStrings; const Attr : Int)
29132: Proc GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
29133: Proc GetFormatSettings
29134: Proc GetFromDIB( var DIB : TBitmapInfo)
29135: Proc GetFromHDIB( HDIB : HBitmap)
29136: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
29137: Proc GetGEOMap(C_form,apath:Str; const Data:Str); //c_form: [html/json/xml]
29138: Proc GetIcon( Index : Int; Image : TIcon);
29139: Proc GetIcon1(Index:Int; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
29140: Proc GetIndexInfo( IndexName :Str)
29141: Proc GetIndexNames( const TableName, SchemaName :Str; List : TStrings);
29142: Proc GetIndexNames( List : TStrings)
29143: Proc GetIndexNames1(const TableName : WideString; List : TWideStrings);
29144: Proc GetIndexNames2(const TableName, SchemaName : WideString; List : TWideStrings);
29145: Proc GetIndexNames4(const TableName :Str; List : TStrings);
29146: Proc GetInternalResponse
29147: Proc GETITEMNAMES( LIST : TSTRINGS)
29148: Proc GetMem(P: PChar; Size: Int)
29149: Proc GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:Int;GROUPS:array of Int)
29150: Proc GetPackageDescription(ModuleName: PChar):Str)
29151: Proc GetPackageNames( List : TStrings);
29152: Proc GetPackageNames1( List : TWideStrings);
29153: Proc GetParamList( List : TList; const ParamNames : WideString)
29154: Proc GetProcedureNames( List : TStrings);
29155: Proc GetProcedureNames( List : TWideStrings);
29156: Proc GetProcedureNames1(const PackageName :Str; List : TStrings);
29157: Proc GetProcedureNames1(List : TStrings);
29158: Proc GetProcedureNames2(const PackageName, SchemaName :Str; List : TStrings);
29159: Proc GetProcedureNames3(List : TWideStrings);
29160: Proc GetProcedureNames4(const PackageName : Widestring; List : TWideStrings);
29161: Proc GetProcedureNames5(const PackageName,SchemaName: WideString; List : TWideStrings);
29162: Proc GetProcedureParams(ProcedureName : WideString; List : TList);
29163: Proc GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
29164: Proc GetProcedureParams2(ProcedureName,PackageName,SchemaName:Widestring;List:TList);
29165: Proc GetProviderNames( Names : TWideStrings);
29166: Proc GetProviderNames( Proc : TGetStrProc)
```

```
29167: Proc GetProviderNames1( Names : TStrings);
29168: Proc GetQrCode2(Width,Height:Word;Correct_Level:str;const Data:str;apath:Str);
29169: Proc GetQrCode3(Width,Height:Word;Correct_Level:str;const Data:str;apath:str);//no open image
29170: Func GetQrCode4(Width,Height:Word;Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
29171: Proc GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
29172: Proc GetSchemaNames( List : TStrings);
29173: Proc GetSchemaNames1( List : TWideStrings);
29174: Proc getScriptandRunAsk;
29175: Proc getScriptandRun(ascript:Str);
29176: Proc getScript(ascript:Str);   //alias
29177: Proc getWebScript(ascript:Str);   //alias
29178: Proc GetSessionNames( List : TStrings)
29179: Proc GetStoredProcNames( const DatabaseName :Str; List : TStrings)
29180: Proc GetStrings( List : TStrings)
29181: Proc GetSystemTime; stdcall;
29182: Proc GetTableNames(const DatabseName,Pattern:str;Extensins,SystemTables:Bool;List:TStrings)
29183: Proc GetTableNames( List : TStrings; SystemTables :Bool)
29184: Proc GetTableNames( List : TStrings; SystemTables :Bool);
29185: Proc GetTableNames( List : TWideStrings; SystemTables :Bool);
29186: Proc GetTableNames1(List: TStrings; SchemaName : WideString; SystemTables Boolean);
29187: Proc GetTableNames1( List : TStrings; SystemTables :Bool);
29188: Proc GetTableNames2(List: TWideStrings; SchemaName:WideString;SystemTables:Boolean);
29189: Proc GetTransitionsOn( cChar : char; oStateList : TList)
29190: Proc GetVisibleWindows( List : Tstrings)
29191: Proc GoBegin
29192: Proc GotoCurrent( DataSet : TCustomClientDataSet)
29193: Proc GotoCurrent( Table : TTable)
29194: Proc GotoEndlClick(Sender: TObject);
29195: Proc GotoNearest
29196: Proc GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
29197: Proc HandleException( E : Exception; var Handled :Bool)
29198: Proc HANDLEMESSAGE
29199: Proc HandleNeeded;
29200: Proc Head( AURL :Str)
29201: Proc Help( var AHelpContents : TStringList; ACommand :Str)
29202: Proc HexToBinary( Stream : TStream)
29203: Proc HexToBinary(Stream:TStream)
29204: Proc HideDragImage
29205: Proc HideFormCaption( FormHandle : THandle; Hide :Bool)
29206: Proc HideTraybar
29207: Proc HideWindowForSeconds(secs: Int);   {//3 seconds}
29208: Proc HideWindowForSeconds2(secs: Int; apphandle, aself: TForm);   {//3 seconds}
29209: Proc HookOSExceptions
29210: Proc HookSignal( RtlSigNum : Int)
29211: Proc HSLToRGB( const H, S, L : Single; out R, G, B : Single);
29212: Proc HTMLSyntax1Click( Sender : TObject)
29213: Proc IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
29214: Proc IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
29215: Proc ImportfromClipboard1Click( Sender : TObject)
29216: Proc ImportfromClipboard2Click( Sender : TObject)
29217: Proc IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Int)
29218: Proc Incb(var x: byte);
29219: Proc Incb2(var x: byte; inc: integer);
29220: Proc Include1Click( Sender : TObject)
29221: Proc IncludeOFF;  //preprocessing
29222: Proc IncludeON;
29223: Proc Info1Click(Sender: TObject);
29224: Proc InitAltRecBuffers( CheckModified :Bool)
29225: Proc InitContext( Request : TWebRequest; Response : TWebResponse)
29226: Proc InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
29227: Proc InitData( ASource : TDataSet)
29228: Proc InitDelta( ADelta : TPacketDataSet);
29229: Proc InitDelta1( const ADelta : OleVariant);
29230: Proc InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
29231: Proc Initialize
29232: Proc InitializePackage(Module: HMODULE)
29233: Proc INITIATEACTION
29234: Proc initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
29235: Proc InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
29236: Proc InitModule( AModule : TComponent)
29237: Proc InitStdConvs
29238: Proc InitTreeData( Tree : TUpdateTree)
29239: Proc INSERT
29240: Proc Insert(Index : Int; AClass : TClass)
29241: Proc Insert(Index : Int; AComponent : TComponent)
29242: Proc Insert(Index : Int; AObject : TObject)
29243: Proc Insert(Index : Int; const S : WideString)
29244: Proc Insert(Index : Int; Image, Mask : TBitmap)
29245: Proc Insert(Index: Int; const S:Str);
29246: Proc Insert(Index: Int; S:Str);
29247: Proc Insert(s: AnyString; var s2: AnyString; iPos: Longint);
29248: Proc InsertComponent(AComponent:TComponent)
29249: Proc InsertControl(AControl: TControl);
29250: Proc InsertIcon( Index : Int; Image : TIcon)
29251: Proc InsertMasked( Index : Int; Image : TBitmap; MaskColor : TColor)
29252: Proc InsertObject( Index : Int; const S : WideString; AObject : TObject)
29253: Proc InsertObject(Index:Int;S:str;AObject:TObject)
29254: Proc Int16ToBytes( Value : SmallInt; Bytes : array of byte)
```

```
29255: Proc Int32ToBytes( Value : Int; Bytes : array of byte)
29256: Proc Int64ToBytes( Value : Int64; Bytes : array of byte)
29257: Proc I64ToCardinals(I: Int64; var LowPart, HighPart:Card);
29258: Proc InternalBeforeResolve( Tree : TUpdateTree)
29259: Proc InvalidateModuleCache
29260: Proc InvalidateTitles
29261: Proc InvalidDateDayError( const AYear, ADayOfYear : Word)
29262: Proc InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
29263: Proc InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const ABaseDate:TDateTime)
29264: Proc InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
29265: Proc InvalidDayOfWeekInMonthError(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek: Word)
29266: Proc JavaSyntax1Click(Sender: TObject);
29267: Proc JclLocalesInfoList( const Strings : TStrings; InfoType : Int)
29268: Proc KillDataChannel
29269: Proc Largefont1Click( Sender : TObject)
29270: Proc LAST
29271: Proc LaunchCpl( FileName :Str)
29272: Proc Launch( const AFile :Str)
29273: Proc LaunchFile( const AFile :Str)
29274: Proc LetFileList(FileList: TStringlist; apath:Str);
29275: Proc lineToNumber( xmemo :Str; met :Bool)
29276: Proc ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var DefaultDraw:Bool)
29277: Proc
       ListViewCustomDrawSubItem(Sender:TCustomListView;Item:TListItem;SubItem:Int;State:TCustomDrawState;var DefaultDraw:Boo
29278: Proc ListViewData( Sender : TObject; Item : TListItem)
29279: Proc ListViewDataFind(Sender:TObject;Find:TItemFind;const FindString:Str;const
       FindPosition:TPoint;FindData:Pointer;StartIndex:Int;Direction:TSearchDirection;Wrap:Boolean;var Index:Int)
29280: Proc ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Int)
29281: Proc ListViewDblClick( Sender : TObject)
29282: Proc ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
29283: Proc ListDLLExports(const FileName:Str; List: TStrings);
29284: Proc Load( const WSDLFileName : WideString; Stream : TMemoryStream)
29285: Proc LoadBytecode1Click(Sender: TObject);
29286: Proc LoadFilefromResource(const FileName:Str; ms: TMemoryStream);
29287: Proc LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
29288: Proc LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
29289: Proc LoadFromFile( AFileName :Str)
29290: Proc LoadFromFile( const AFileName :Str; const AHeadersOnly :Bool)
29291: Proc LoadFromFile( const FileName :Str)
29292: Proc LOADFROMFILE( const FILENAME :Str; BLOBTYPE : TBLOBTYPE)
29293: Proc LoadFromFile( const FileName :Str; DataType : TDataType)
29294: Proc LoadFromFile( const FileName : WideString)
29295: Proc LoadFromFile( const FileName, FileType :Str; Bitmap : TLinearBitmap)
29296: Proc LoadFromFile(const AFileName:Str)
29297: Proc LoadFromFile(FileName:Str);
29298: Proc LoadFromFile(FileName:str)
29299: Proc LoadFromResourceID( Instance : THandle; ResID : Int)
29300: Proc LoadFromResourceName( Instance : THandle; const ResName :Str)
29301: Proc LoadFromStream( AStream : TStream; const AHeadersOnly :Bool)
29302: Proc LoadFromStream( const Stream : TStream)
29303: Proc LoadFromStream( S : TStream)
29304: Proc LoadFromStream(Stream: TSeekableStream; const Ext :Str; Bitmap : TLinarBitmap)
29305: Proc LoadFromStream(Stream: TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
29306: Proc LoadFromStream( Stream : TStream)
29307: Proc LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
29308: Proc LoadFromStream( Stream : TStream; DataType : TDataType)
29309: Proc LoadFromStream(Stream: TStream);
29310: Proc LoadFromStream1( Stream : TSeekableStream; const FormatExt :Str);
29311: Proc LoadFromStream2( Stream : TStream; const FormatExt :Str);
29312: Proc LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
29313: Proc LoadLastFile1Click( Sender : TObject)
29314: { LoadIcoToImage loads two icons from resource named NameRes,
29315:   into two image lists ALarge and ASmall}
29316: Proc LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes:Str);
29317: Proc LoadMemo
29318: Proc LoadParamsFromIniFile( FFileName : WideString)
29319: Proc Lock
29320: Proc Login
29321: Proc MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
29322: Proc MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
29323: Proc MakeCaseInsensitive
29324: Proc MakeDeterministic( var bChanged :Bool)
29325: Proc MakeGrayPal( var Palette, ColorCount : Int)
29326: // type TVolumeLevel = 0..127;, savaFilePath as C:\MyFile.wav
29327: Proc MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
29328: Proc MakeComplexSound(N:Int;freqlist:TStrgs;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
29329: type MSArray = array[0..1] of TMemoryStream;
29330: Func MakeComplexSound2(N:integer {stream # to use};
       freqlist:TStrings;Duration{mSec}:Integer;pinknoise:Bool;shape:integer;Volume:TVolumeLevel):MSArray;
29331: Proc SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:Int);
29332: Proc SetRectComplexFormatStr( const S :Str)
29333: Proc SetPolarComplexFormatStr( const S :Str)
29334: Proc AddComplexSoundObjectToList(newf,newp,newa,news:Int; freqlist: TStrings);
29335: Proc MakeVisible
29336: Proc MakeVisible( PartialOK :Bool)
29337: Proc Manual1Click( Sender : TObject)
29338: Proc MarkReachable
29339: Proc maXbox;  //shows the exe version data in a win box
29340: Proc MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
```

```
29341: Proc Memo1Change( Sender : TObject)
29342: Proc Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:str;Line,Column:Int;var
       Action:TSynReplaceAction)
29343: Proc Memo1SpecialLineColors(Sender:TObject;Line:Int;var Special:Boolean;var FG,BG:TColor)
29344: Proc Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
29345: Proc Memory1Click(Sender: TObject);
29346: Proc MERGE( MENU : TMAINMENU)
29347: Proc MergeChangeLog
29348: Proc MINIMIZE
29349: Proc MinimizeMaxbox;
29350: Proc MyCopyFile(Name1,Name2:str);
29351: Proc MkDir(const s:Str)
29352: Proc MakeDir(const s:Str);
29353: Proc ChangeDir(const s:Str);
29354: Func  makeFile(const FileName:Str): Int);
29355: Proc mnuPrintFont1Click( Sender : TObject)
29356: Proc ModalStarted
29357: Proc Modified
29358: Proc ModifyAlias( Name :Str; List : TStrings)
29359: Proc ModifyDriver( Name :Str; List : TStrings)
29360: Proc MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4,Skew,Kurtosis:Extended)
29361: Proc MouseToCell( X, Y : Int; var ACol, ARow : Longint)
29362: Proc Move( CurIndex, NewIndex : Int)
29363: Proc Move(CurIndex, NewIndex: Int);
29364: Proc Move2(const Source: TByteArray; var Dest: TByteArray; Count: Int)
29365: Proc MoveChars(const ASource:str;ASourceStart:int;var ADest:str;ADestStart,ALen:int)
29366: Proc moveCube( o : TMyLabel)
29367: Proc MoveTo( Destination : LongInt; AttachMode : TAttachMode)
29368: Proc MoveTo(X, Y: Int);
29369: Proc MoveWindowOrg(DC: HDC; DX, DY: Int);
29370: Proc MovePoint(var x,y:Extended; const angle:Extended);
29371: Proc Multiply( Multiplier1, Multiplier2 : TMyBigInt);
29372: Proc Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Int);
29373: Proc MsgAbout(Handle:Int;const Msg,Caption:str;const IcoName:str='MAINICON';Flags:DWORD=MB_OK);
29374: Proc mxButton(x,y,width,height,top,left,ahandle: Int);
29375: Proc New( Width, Height : Int; PixFormat : TPixelFormat)
29376: Proc New(P: PChar)
29377: Proc New1Click(Sender: TObject);
29378: Proc NewInstance1Click(Sender: TObject);
29379: Proc NEXT
29380: Proc NextMonth
29381: Proc Noop
29382: Proc NormalizePath( var APath :Str)
29383: Proc ObjectBinaryToText(Input, Output: TStream)
29384: Proc ObjectBinaryToText1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
29385: Proc ObjectResourceToText(Input, Output: TStream)
29386: Proc ObjectResourceToText1(Input,Output:TStream;var OriginalFormat:TStreamOriginalFormat)
29387: Proc ObjectTextToBinary(Input, Output: TStream)
29388: Proc ObjectTextToBinary1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
29389: Proc ObjectTextToResource(Input, Output: TStream)
29390: Proc ObjectTextToResource1(Input,Output:TStream;var OriginalFormat:TStreamOriginalFormat)
29391: Proc Open( const Name, Address, Service :Str; Port : Word; Block :Bool)
29392: Proc Open( const UserID : WideString; const Password : WideString);
29393: Proc Open;
29394: Proc open1Click( Sender : TObject)
29395: Proc OpenCdDrive
29396: Proc OpenCloseCdDrive( OpenMode :Bool; Drive : Char)
29397: Proc OpenCurrent
29398: Proc OpenFile(vfilenamepath:Str)
29399: Proc OpenDirectory1Click( Sender : TObject)
29400: Proc OpenDir(adir:Str);
29401: Proc OpenIndexFile( const IndexName :Str)
29402: Proc OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
29403: Proc OpenWriteBuffer( const AThreshhold : Int)
29404: Proc OptimizeMem
29405: Proc Options1( AURL :Str);
29406: Proc OutputDebugString(lpOutputString : PChar)
29407: Proc PackBuffer
29408: Proc Paint
29409: Proc PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch :Bool)
29410: Proc PaintToTBitmap( Target : TBitmap)
29411: Proc PaletteChanged
29412: Proc ParentBiDiModeChanged
29413: Proc PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
29414: Proc PasteFromClipboard;
29415: Proc PasteImage( Source : TLinearBitmap; X, Y : Int)
29416: Proc PathExtractElements(const Source:str;var Drive,Path,FileName,Ext:Str)
29417: Proc PerformEraseBackground(Control: TControl; DC: HDC);
29418: Proc PError( Text :Str)
29419: Proc Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
29420: Proc Pie(X1:Int;Y1:Int;X2:Int;Y2:Int;X3:Int;Y3:Int;X4:Int;Y4:Int);
29421: Proc Play( FromFrame, ToFrame : Word; Count : Int)
29422: Proc playmp3(mpath:Str);
29423: Proc closeMP3;
29424: Proc PlayMP31Click( Sender : TObject)
29425: Proc PointCopy( var Dest : TPoint; const Source : TPoint)
29426: Proc PointMove( var P : TPoint; const DeltaX, DeltaY : Int)
29427: Proc PolyBezier(const Points: array of TPoint);
```

```
29428: Proc PolyBezierTo(const Points: array of TPoint);
29429: Proc Polygon(const Points: array of TPoint);
29430: Proc Polyline(const Points: array of TPoint);
29431: Proc Pop
29432: Proc POPULATEOLE2MENU(SHAREDMENU:HMENU;GROUPS:array of INT;var WIDTHS:array of LONGINT)
29433: Proc PopulationVarianceAndMean(const X: TDynFloatArray;var Variance, Mean: Float)
29434: Proc POPUP( X, Y : Int)
29435: Proc PopupURL(URL : WideString);
29436: Proc POST
29437: Proc Post4(AURL:Str;const ASource: TStrings; const AResponseContent : TStream);
29438: Proc Post5( AURL :Str; const ASource, AResponseContent : TStream);
29439: Proc Post6(AURL:str;const ASource:TIdMultiPartFormDataStream;AResponseContent:TStream);
29440: Proc PostUser( const Email, FirstName, LastName : WideString)
29441: Proc PostKeyEx32(key: Word; const shift: TShiftState; specialkey:Bool);
29442: Proc Pred(X: int64);
29443: Proc Prepare
29444: Proc PrepareStatement
29445: Proc PreProcessXML( AList : TStrings)
29446: Proc PreventDestruction
29447: Proc Print( const Caption :Str)
29448: Proc PrintBitmap(aGraphic: TGraphic; Title:Str);
29449: Proc printf(const format:Str; const args: array of const);
29450: Proc PrintList(Value: TStringList);
29451: Proc PrintImage(aValue:TBitmap;Style:TBitmapStyle);
29452: //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
29453: Proc Printout1Click( Sender : TObject)
29454: Proc ProcessHeaders
29455: Proc PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
29456: Proc ProcessMessage( AMsg : TIdMessage; AHeaderOnly :Bool);
29457: Proc ProcessMessage1(AMsg:TIdMessage;const AStream TStream; AHeaderOnly:Bool);
29458: Proc ProcessMessage2(AMsg: TIdMessage; const AFilename :Str; AHeaderOnly :Bool);
29459: Proc ProcessMessagesOFF;  //application.processmessages
29460: Proc ProcessMessagesON;
29461: Proc ProcessPath(const EditText:str;var Drive:Char;var DirPart:str;var FilePart:str)
29462: Proc ProcessPath1(const EditText:str;var Drive:Char;var DirPart:str;var FilePart:str);
29463: Proc Proclist Size is: 3797 /1415
29464: Proc procMessClick( Sender : TObject)
29465: Proc PSScriptCompile( Sender : TPSScript)
29466: Proc PSScriptExecute( Sender : TPSScript)
29467: Proc PSScriptLine( Sender : TObject)
29468: Proc Push( ABoundary :Str)
29469: Proc PushItem(AItem: Pointer)
29470: Proc Put2( AURL :Str; const ASource, AResponseContent : TStream);
29471: Proc Put2(const ASourceFile:Str; const ADestFile:Str;const AAppend:boolean);
29472: Proc PutLinuxLines(const Value:Str)
29473: Proc Quit
29474: Proc RaiseConversionError( const AText :Str);
29475: Proc RaiseConversionError1( const AText :Str; const AArgs : array of const);
29476: Proc RaiseConversionRegError( AFamily : TConvFamily; const ADescription :Str)
29477: Proc RaiseException(Ex: TIFException; Param:Str);
29478: Proc RaiseExceptionForLastCmdResult;
29479: Proc RaiseLastException;
29480: Proc RaiseException2;
29481: Proc RaiseException3(const Msg:Str);
29482: Proc RaiseExcept(const Msg:Str);
29483: Proc RaiseLastOSError
29484: Proc RaiseLastWin32;
29485: Proc RaiseLastWin32Error)
29486: Proc RaiseListError( const ATemplate :Str; const AData : array of const)
29487: Proc RandomFillStream( Stream : TMemoryStream)
29488: Proc randomize;
29489: Proc Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
29490: Proc RCS
29491: Proc Read( Socket : TSocket)
29492: Proc Readln1(var ast:Str);  //of inputquery
29493: Proc ReadBlobData
29494: Proc ReadBuffer(Buffer:str;Count:LongInt)
29495: Proc ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
29496: Proc ReadSection( const Section :Str; Strings : TStrings)
29497: Proc ReadSections( Strings : TStrings)
29498: Proc ReadSections( Strings : TStrings);
29499: Proc ReadSections1( const Section :Str; Strings : TStrings);
29500: Proc ReadSectionValues( const Section :Str; Strings : TStrings)
29501: Proc ReadStream( AStream:TStream;AByteCount:LongInt;const AReadUntilDisconnect:Bool)
29502: Proc ReadStrings( ADest : TStrings; AReadLinesCount : Int)
29503: Proc ReadVersion2(aFileName:Str; aVersion : TStrings);
29504: Func  ReadVersion(aFileName:Str; aVersion : TStrings):Bool;
29505: Proc Realign;
29506: Proc Rectangle(X1, Y1, X2, Y2: Int);
29507: Proc Rectangle1( const Rect : TRect);
29508: Proc RectCopy( var Dest : TRect; const Source : TRect)
29509: Proc RectFitToScreen( var R : TRect)
29510: Proc RectGrow( var R : TRect; const Delta : Int)
29511: Proc RectGrowX( var R : TRect; const Delta : Int)
29512: Proc RectGrowY( var R : TRect; const Delta : Int)
29513: Proc RectMove( var R : TRect; const DeltaX, DeltaY : Int)
29514: Proc RectMoveTo( var R : TRect; const X, Y : Int)
29515: Proc RectNormalize( var R : TRect)
29516: //  TFileCallbackProc = procedure(filename:str);
```

```
29517: Proc RecurseDirectory(Dir:Str;IncludeSubs:Bool;callback: TFileCallbackProcedure);
29518: Proc RecurseDirectory2(Dir:Str; IncludeSubs :Bool);
29519: Proc RedirectTransition(oOldState:TniRegularExpressionState;oNewState:TniRegularExpressionState)
29520: Proc Refresh;
29521: Proc RefreshData( Options : TFetchOptions)
29522: Proc REFRESHLOOKUPLIST
29523: Proc regExPathfinder(Pathin, fileout, firstp, aregex, ext:Str; asort:Bool);
29524: Proc RegExPathfinder2(Pathin,fileout,firstp, aregex, ext:Str; asort, acopy:Bool);
29525: Proc RegisterAuthenticationMethod(MethodName:Str; AuthClass : TIdAuthenticationClass)
29526: Proc RegisterChanges( Value : TChangeLink)
29527: Proc RegisterConversionFormat(const AExtension:str;AConversionClass:TConversionClass)
29528: Proc RegisterFileFormat(const AExtension,ADescription:str;AGraphicClass:TGraphicClass)
29529: Proc RegisterFileFormat(Extension,AppID:str;Description:str;Executable:str;IconIndex:Int)
29530: Proc ReInitialize( ADelay :Card)
29531: Proc RELEASE
29532: Proc Remove( const AByteCount : Int)
29533: Proc REMOVE( FIELD : TFIELD)
29534: Proc REMOVE( ITEM : TMENUITEM)
29535: Proc REMOVE( POPUP : TPOPUPMENU)
29536: Proc RemoveAllPasswords
29537: Proc RemoveComponent(AComponent:TComponent)
29538: Proc RemoveDir( const ADirName :Str)
29539: Proc RemoveLambdaTransitions( var bChanged :Bool)
29540: Proc REMOVEPARAM( VALUE : TPARAM)
29541: Proc RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
29542: Proc RemoveTransitionTo1( oState : TniRegularExpressionState);
29543: Proc Rename( const ASourceFile, ADestFile :Str)
29544: Proc Rename( const FileName :Str; Reload :Bool)
29545: Proc RenameTable( const NewTableName :Str)
29546: Proc Replace( Index : Int; Image, Mask : TBitmap)
29547: Proc Replace1Click( Sender : TObject)
29548: Proc ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
29549: Proc ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
29550: Proc ReplaceIcon( Index : Int; Image : TIcon)
29551: Proc ReplaceMasked( Index : Int; NewImage : TBitmap; MaskColor : TColor)
29552: Proc ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
29553: Proc ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
29554: Proc Requery( Options : TExecuteOptions)
29555: Proc Reset
29556: Proc Reset1Click( Sender : TObject)
29557: Proc ResizeCanvas( XSiz, YSiz, XPos, YPos : Int; Color : TColor)
29558: Proc ResourceExplore1Click(Sender: TObject);
29559: Proc RestoreContents
29560: Proc RestoreDefaults
29561: Proc RestoreOtherInstance( MainFormClassName, MainFormCaption :Str)
29562: Proc RetrieveHeaders
29563: Proc RevertRecord
29564: Proc RGBAToBGRA(const Source, Target: Pointer; const BitsPerSample:Byte;Count :Card)
29565: Proc RGBToBGR( const Source, Target:Pointer; const BitsPerSample:Byte; Count :Card);
29566: Proc RGBToBGR1(const R,G,B,Target: Pointer; const BitsPerSample : Byte; Count :Card);
29567: Proc RGBToHSL( const R, G, B : Single; out H, S, L : Single);
29568: Proc RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
29569: Proc RGBToHSV( r, g, b : Int; var h, s, v : Int)
29570: Proc RleCompress2( Stream : TStream)
29571: Proc RleDecompress2( Stream : TStream)
29572: Proc RmDir(const S:Str)
29573: Proc Rollback
29574: Proc Rollback( TransDesc : TTransactionDesc)
29575: Proc RollbackFreeAndNil( var Transaction : TDBXTransaction)
29576: Proc RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
29577: Proc RollbackTrans
29578: Proc RoundRect(X1, Y1, X2, Y2, X3, Y3: Int);
29579: Proc RoundToAllocGranularity64( var Value : Int64; Up :Bool)
29580: Proc RoundToAllocGranularityPtr( var Value : Pointer; Up :Bool)
29581: Proc RunDll32Internal(Wnd:HWnd; const DLLName,FuncName,CmdLine:str;CmdShow : Int)
29582: procedure RunDosInMemo(DosApp: string; AMemo:TMemo);');
29583: Proc S_AddMessageToStrings( AMessages : TStrings; AMsg :Str)
29584: Proc S_EBox( const AText :Str)
29585: Proc S_GetEncryptionKeys(DatTime1,DTime2:TDateTime;var StartKey:int;var MultKey:int;var AddKey:int
29586: Proc S_IBox( const AText :Str)
29587: Proc S_ReplaceChar( var cStr :Str; cOldChr, cNewChr : char)
29588: Proc S_ReplaceStringInFile(AFileName:str; ASearchString, AReplaceString :Str)
29589: Proc S_TokenInit( cBuffer : PChar; const cDelimiters :Str)
29590: Proc SampleVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
29591: Proc Save2Click( Sender : TObject)
29592: Proc Saveas3Click( Sender : TObject)
29593: Proc Savebefore1Click( Sender : TObject)
29594: Proc SaveBytesToFile(const Data: TBytes; const FileName:Str);
29595: Proc SaveCanvas2(vCanvas: TCanvas; FileName:Str);
29596: Proc SaveConfigFile
29597: Proc SaveOutput1Click( Sender : TObject)
29598: Proc SaveScreenshotClick(Sender: TObject);
29599: Proc SaveLn(pathname,content:Str); //Saveln(exepath+'mysavelntest.txt', memo2.text);
29600: Proc SaveToClipboardFormat(var AFormat:Word; var AData: THandle; var APalette: HPALETTE)
29601: Proc SaveToClipboardFormat(var Format: Word; var Data : THandle; var APalette: HPALETTE)
29602: Proc SaveToFile( AFileName :Str)
29603: Proc SAVETOFILE( const FILENAME :Str)
29604: Proc SaveToFile( const FileName : WideString)
29605: Proc SaveToFile( const FileName : WideString; Format : TPersistFormat)
```

```
29606: Proc SaveToFile( const FileName, FileType :Str; Bitmap : TLinearBitmap)
29607: Proc SaveToFile(FileName:Str);
29608: Proc SaveToFile(FileName:str)
29609: Proc SaveToStream( AStream : TStream; const AHeadersOnly :Bool)
29610: Proc SaveToStream(OutStream TSeekableStream; const Ext :Str; Bitmap : TLinarBitmap)
29611: Proc SaveToStream( S : TStream)
29612: Proc SaveToStream( Stream : TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
29613: Proc SaveToStream( Stream : TStream)
29614: Proc SaveToStream( Stream : TStream; Format : TDataPacketFormat)
29615: Proc SaveToStream(Stream: TStream);
29616: Proc SaveToStream(Stream:TStream)
29617: Proc SaveToStream1( Stream : TSeekableStream; const FormatExt :Str);
29618: Proc SaveToStream2( Stream : TStream; const FormatExt :Str);
29619: Proc SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
29620: Proc Say(const sText:Str)
29621: Proc SBytecode1Click( Sender : TObject)
29622: Proc ScaleImage(const SourceBitmap, ResizedBitmap:TBitmap;const ScaleAmount:Double)
29623: Proc ScriptExplorer1Click(Sender: TObject);
29624: Proc Scroll( Distance : Int)
29625: Proc Scroll( DX, DY : Int)
29626: Proc ScrollBy(DeltaX, DeltaY: Int);
29627: Proc SCROLLINVIEW(ACONTROL:TCONTROL)
29628: Proc ScrollTabs( Delta : Int)
29629: Proc Search1Click( Sender : TObject)
29630: Proc SearchAndOpenDoc(vfilenamepath:Str)
29631: Proc SearchAndOpenFile(vfilenamepath:Str)
29632: Proc SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr:Str)
29633: Proc SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr:Str; offset: Int);
29634: Proc SearchNext1Click( Sender : TObject)
29635: Proc Select( Node : TTreeNode; ShiftState : TShiftState);
29636: Proc Select1( const Nodes : array of TTreeNode);
29637: Proc Select2( Nodes : TList);
29638: Proc SelectNext( Direction :Bool)
29639: Proc SelectNextPage( GoForward :Bool; CheckTabVisible :Bool)
29640: Proc SelfTestPEM  //unit uPSI_cPEM
29641: Proc Send( AMsg : TIdMessage)
29642: //config forst in const MAILINIFILE = 'maildef.ini';
29643: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
29644: Proc SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
29645: Proc SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
29646: Proc SendMsg( AMsg : TIdMessage; const AHeadersOnly :Bool)
29647: Proc SendMsg(AMsg: TIdMessage; const AHeadersOnly:Bool = False)
29648: Proc SendResponse
29649: Proc SendStream( AStream : TStream)
29650: Proc SendMCICommand(Cmd:Str);   !
29651: Proc Set8087CW( NewCW : Word)
29652: Proc SetAll( One, Two, Three, Four : Byte)
29653: Proc SetAltRecBuffers( Old, New, Cur : PChar)
29654: Proc SetAppDispatcher( const ADispatcher : TComponent)
29655: Proc SetArrayLength;
29656: Proc SetArrayLength2String(arr: T2StringArray; asize1, asize2: Int);  //2 dimension
29657: Proc SetArrayLength2Int(arr: T2IntArray; asize1, asize2: Int);
29658: Proc SetArrayLength2String2(arr: T2StringArray; asize1, asize2: Int);  //all init
29659: Proc SetArrayLength2Int2(arr: T2IntArray; asize1, asize2: Int);
29660: Proc SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: Int);;
29661: Proc Set2DimStrArray(var arr: T2StringArray; asize1, asize2: Int);;
29662: Proc Set2DimIntArray(var arr: T2IntArray; asize1, asize2: Int);;
29663: Proc Set3DimIntArray(var arr: T3IntArray; asize1, asize2, asize3: Int);
29664: Proc Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: Int);
29665: Proc SetAsHandle( Format : Word; Value : THandle)
29666: Proc SetBounds(ALeft, ATop, AWidth, AHeight: Int)
29667: Proc SetCaptureControl(Control: TControl);
29668: Proc SetColumnAttributes
29669: Proc SetCookieField(Values:TStrings;const ADoman,APath:str;AExpires:TDateTime;ASecure:Bool)
29670: Proc SetCustomHeader( const Name, Value :Str)
29671: Proc SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
29672: Proc SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
29673: Proc SetFocus
29674: Proc SetFocus; virtual;
29675: Proc SetInitialState
29676: Proc SetKey
29677: Proc SetLastError(ErrorCode: Int)
29678: Proc SetLength;
29679: Proc SetLength2(var S:Str; NewLength: Int);
29680: Proc SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
29681: Proc SETOLE2MENUHANDLE( HANDLE : HMENU)
29682: Proc SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
29683: Proc SETPARAMS(APOSITION,AMIN,AMAX:Int)
29684: Proc SetParams1( UpdateKind : TUpdateKind);
29685: Proc SetPassword( const Password :Str)
29686: Proc SetPointer( Ptr : Pointer; Size : Longint)
29687: Proc SetPrimalityTest( const Method : TPrimalityTestMethod)
29688: Proc SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
29689: Proc SetProvider( Provider : TComponent)
29690: Proc SetProxy( const Proxy :Str)
29691: Proc SetPSResult( var PSResult : TPSResult; Value : TObject)
29692: Proc SetRange( const StartValues, EndValues : array of const)
29693: Proc SetRangeEnd
```

```
29694: Proc SetRate( const aPercent, aYear : Int)
29695: Proc SetRate(const aPercent, aYear: Int)
29696: Proc Set_ReportMemoryLeaksOnShutdown(abo:Bool)
29697: Proc SetSafeCallExceptionMsg( Msg :Str)
29698: Proc SETSELTEXTBUF(BUFFER:PCHAR)
29699: Proc SetSize( AWidth, AHeight : Int)
29700: Proc SetSize(NewSize:LongInt)
29701: Proc SetString(var s:Str; buffer: PChar; len: Int)
29702: Proc SetStrings( List : TStrings)
29703: Proc SetText( Text : PwideChar)
29704: Proc SetText(Text: PChar);
29705: Proc SetTextBuf( Buffer : PChar)
29706: Proc SETTEXTBUF(BUFFER:PCHAR)
29707: Proc SetTick( Value : Int)
29708: Proc SetTimeout( ATimeOut : Int)
29709: Proc SetTraceEvent( Event : TDBXTraceEvent)
29710: Proc SetUserName( const UserName :Str)
29711: Proc SetWallpaper( Path :Str);
29712: Proc ShellStyle1Click(Sender: TObject);
29713: Proc SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
29714: Proc ShowFileProperties( const FileName :Str)
29715: Proc ShowInclude1Click( Sender : TObject)
29716: Proc ShowInterfaces1Click( Sender : TObject)
29717: Proc ShowLastException1Click( Sender : TObject)
29718: Proc ShowMessage( const Msg :Str)
29719: Proc ShowMessageBig(const aText :Str);
29720: Proc ShowMessageBig2(const aText :Str; aautosize:Bool);
29721: Proc ShowMessageBig3(const aText :Str; fsize: byte; aautosize:Bool);
29722: Proc MsgBig(const aText :Str);        //alias
29723: Proc showmessage(mytext:Str);
29724: Proc ShowMessageFmt( const Msg :Str; Params : array of const)
29725: Proc ShowMessageFmt(const Msg:Str; Params: array of const))
29726: Proc ShowMessagePos( const Msg :Str; X, Y : Int)
29727: Proc ShowMessagePos(const Msg:Str; X: Int; Y: Int))
29728: Proc ShowSearchDialog( const Directory :Str)
29729: Proc ShowSpecChars1Click( Sender : TObject)
29730: Proc ShowBitmap(bmap: TBitmap);  //draw in a form!
29731: Proc ShredFile( const FileName :Str; Times : Int)
29732: Proc Shuffle(vQ: TStringList);
29733: Proc ShuffleList( var List : array of Int; Count : Int)
29734: Proc SimulateKeystroke( Key : byte; Shift : TShiftState)
29735: Proc SinCos( const Theta : Extended; var Sin, Cos : Extended)
29736: Proc SinCosE( X : Extended; out Sin, Cos : Extended)
29737: Proc Site( const ACommand :Str)
29738: Proc SkipEOL
29739: Proc Sleep( ATime :Card)
29740: Proc Sleep( milliseconds :Card)
29741: Func  SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
29742: Proc Slinenumbers1Click( Sender : TObject)
29743: Proc Sort
29744: Proc SortColorArray(ColorArray:TColorArr;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
29745: Proc SoundAlarm;  //beep seq
29746: Proc Speak(const sText:Str)   //async like voice
29747: Proc Speak2(const sText:Str)  //sync
29748: Proc Split(Str:Str;  SubStr:Str; List: TStrings);
29749: Proc SplitNameValue( const Line :Str; var Name, Value :Str)
29750: Proc SplitColumns( const AData :Str; AStrings : TStrings; const ADelim :Str)
29751: Proc SplitColumnsNoTrim(const AData:str; AStrings:TStrings;const ADelim :Str)
29752: Proc SplitLines( AData : PChar; ADataSize : Int; AStrings : TStrings)
29753: Proc SplitString( const AStr, AToken :Str; var VLeft, VRight :Str)
29754: Proc SQLSyntax1Click(Sender: TObject);
29755: Proc SRand( Seed : RNG_IntType)
29756: Proc Start
29757: Proc StartCount( var Counter : TJclCounter; const Compensate :Bool)
29758: Proc StartFileFinder3(spath,aext,searchstr:Str; arecursiv:Bool; reslist: TStringlist);
29759: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
29760: Proc StartTransaction( TransDesc : TTransactionDesc)
29761: Proc Status( var AStatusList : TStringList)
29762: Proc StatusBar1DblClick( Sender : TObject)
29763: Proc StepInto1Click( Sender : TObject)
29764: Proc StepIt
29765: Proc StepOut1Click( Sender : TObject)
29766: Proc Stop
29767: Proc stopmp3;
29768: Proc StartWeb(aurl:Str);
29769: Proc Str(aint: Int; astr:Str); //of system
29770: Proc StrDispose( Str : PChar)
29771: Proc StrDispose(Str: PChar)
29772: Proc StrReplace(var Str:Str; Old, New:Str);
29773: Proc StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch: Bool)
29774: Proc StretchDraw(const Rect: TRect; Graphic: TGraphic);
29775: Proc StringToBytes( Value :Str; Bytes : array of byte)
29776: Proc StrSet(c : Char; I : Int; var s :Str);
29777: Proc StrSplitP(const Delimiter: Char; Input:Str; const Strings: TStrings);
29778: Proc StructureMount( APath :Str)
29779: Proc STYLECHANGED(SENDER:TOBJECT)
29780: Proc Subselect( Node : TTreeNode; Validate :Bool)
29781: Proc Succ(X: int64);
29782: Proc SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares:Extended)
```

```
29783: Proc SwapChar(var X,Y: char);  //swapX follows
29784: Proc SwapFloats( var X, Y : Float)
29785: Proc SwapGrid(grd: TStringGrid);
29786: Proc SwapOrd( var I, J : Byte);
29787: Proc SwapOrd( var X, Y : Int)
29788: Proc SwapOrd1( var I, J : Shortint);
29789: Proc SwapOrd2( var I, J : Smallint);
29790: Proc SwapOrd3( var I, J : Word);
29791: Proc SwapOrd4( var I, J : Int);
29792: Proc SwapOrd5( var I, J :Card);
29793: Proc SwapOrd6( var I, J : Int64);
29794: Proc SymetricCompareFiles(const plaintext, replaintext:Str)
29795: Proc Synchronize1( Method : TMethod);
29796: Proc SyntaxCheck1Click(Sender: TObject);
29797: Proc SysFreeString(const S: WideString); stdcall;
29798: Proc TakeOver( Other : TLinearBitmap)
29799: Proc Talkln(const sText:Str)  //async voice
29800: Proc tbtn6resClick(Sender: TObject);
29801: Proc tbtnUseCaseClick( Sender : TObject)
29802: Proc TerminalStyle1Click(Sender: TObject);
29803: Proc Terminate
29804: Proc texSyntax1Click( Sender : TObject)
29805: Proc TextOut(X, Y: Int; Text:Str);
29806: Proc TextRect( Rect : TRect; X, Y : Int; const Text :Str);
29807: Proc TextRect(Rect: TRect; X: Int; Y: Int; const Text:Str);
29808: Proc TextRect1( var Rect : TRect; var Text :Str; TextFormat:TTextFormat);
29809: Proc TextStart
29810: Proc TILE
29811: Proc TimeStampToBytes( Value : TBcd; Bytes : array of byte)
29812: Proc TitleClick( Column : TColumn)
29813: Proc ToDo
29814: Proc Tone(500 hz, 10000 ms length);
29815: Proc toolbtnTutorialClick(Sender: TObject);
29816: Proc Trace1( AURL :Str; const AResponseContent : TStream);
29817: Proc TransferMode( ATransferMode : TIdFTPTransferMode)
29818: Proc Truncate
29819: Proc Tutorial101Click(Sender: TObject);
29820: Proc Tutorial10Statistics1Click(Sender: TObject);
29821: Proc Tutorial11Forms1Click(Sender: TObject);
29822: Proc Tutorial12SQL1Click(Sender: TObject);
29823: Proc tutorial1Click( Sender : TObject)
29824: Proc tutorial21Click( Sender : TObject)
29825: Proc tutorial31Click( Sender : TObject)
29826: Proc tutorial4Click( Sender : TObject)
29827: Proc Tutorial5Click( Sender : TObject)
29828: Proc Tutorial6Click(Sender: TObject);
29829: Proc Tutorial91Click(Sender: TObject);
29830: Proc UnhookSignal( RtlSigNum : Int; OnlyIfHooked :Bool)
29831: Proc UniqueString(var str: Ansistr)
29832: Proc UnloadLoadPackage(Module: HMODULE)
29833: Proc Unlock
29834: Proc UNMERGE( MENU : TMAINMENU)
29835: Proc UnRegisterChanges( Value : TChangeLink)
29836: Proc UnregisterConversionFamily( const AFamily : TConvFamily)
29837: Proc UnregisterConversionType( const AType : TConvType)
29838: Proc UnRegisterProvider( Prov : TCustomProvider)
29839: Proc UPDATE
29840: Proc UpdateBatch( AffectRecords : TAffectRecords)
29841: Proc UPDATECURSORPOS
29842: Proc UpdateFile
29843: Proc UpdateItems( FirstIndex, LastIndex : Int)
29844: Proc UpdateResponse( AResponse : TWebResponse)
29845: Proc UpdateScrollBar
29846: Proc UpdateView1Click( Sender : TObject)
29847: Proc UpdateExeResource(Const Source,Dest:str);  //!
29848: Proc Val(const s:Str; var n, z: Int)
29849: Proc VarArraySet(c : Variant; I : Int; var s : Variant);
29850: Proc VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
29851: Proc VariantAdd( const src : Variant; var dst : Variant)
29852: Proc VariantAnd( const src : Variant; var dst : Variant)
29853: Proc VariantArrayRedim( var V : Variant; High : Int)
29854: Proc VariantCast( const src : Variant; var dst : Variant; vt : Int)
29855: Proc VariantClear( var V : Variant)
29856: Proc VariantCpy( const src : Variant; var dst : Variant)
29857: Proc VariantDiv( const src : Variant; var dst : Variant)
29858: Proc VariantMod( const src : Variant; var dst : Variant)
29859: Proc VariantMul( const src : Variant; var dst : Variant)
29860: Proc VariantOr( const src : Variant; var dst : Variant)
29861: Proc VariantPutElement( var V : Variant; const data : Variant; i1 : Int);
29862: Proc VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : Int);
29863: Proc VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : Int);
29864: Proc VariantPutElement3(var V: Variant; const data : Variant; i1, i2, i3, i4 : Int);
29865: Proc VariantPutElement4(var V:Variant;const data: Variant; i1,i2,i3, i4, i5 : Int);
29866: Proc VariantShl( const src : Variant; var dst : Variant)
29867: Proc VariantShr( const src : Variant; var dst : Variant)
29868: Proc VariantSub( const src : Variant; var dst : Variant)
29869: Proc VariantXor( const src : Variant; var dst : Variant)
29870: Proc VarCastError;
29871: Proc VarCastError1( const ASourceType, ADestType : TVarType);
```

```
29872: Proc VarInvalidOp
29873: Proc VarInvalidNullOp
29874: Proc VarOverflowError( const ASourceType, ADestType : TVarType)
29875: Proc VarRangeCheckError( const ASourceType, ADestType : TVarType)
29876: Proc VarArrayCreateError
29877: Proc VarResultCheck( AResult : HRESULT);
29878: Proc VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
29879: Proc HandleConversionException( const ASourceType, ADestType : TVarType)
29880: Func VarTypeAsText( const AType : TVarType) :Str
29881: Proc Voice(const sText:Str) //async
29882: Proc Voice2(const sText:Str) //sync
29883: Proc WaitMiliSeconds( AMSec : word)
29884: Proc WaitMS( AMSec : word);
29885: Proc WebCamPic(picname:Str);  //eg: c:\mypic.png
29886: Proc WideAppend( var dst : WideString; const src : WideString)
29887: Proc WideAssign( var dst : WideString; var src : WideString)
29888: Proc WideDelete( var dst : WideString; index, count : Int)
29889: Proc WideFree( var s : WideString)
29890: Proc WideFromAnsi( var dst : WideString; const src : Ansistr)
29891: Proc WideFromPChar( var dst : WideString; src : PChar)
29892: Proc WideInsert( var dst : WideString; const src : WideString; index : Int)
29893: Proc WideSetLength( var dst : WideString; len : Int)
29894: Proc WideString2Stream( aWideString : WideString; oStream : TStream)
29895: Proc WideStringToBytes( Value : WideString; Bytes : array of byte)
29896: Proc WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
29897: Proc WinInet_HttpGet(const Url:Str; Stream:TStream);
29898: Proc HttpGet(const Url:Str; Stream:TStream);
29899: Proc WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : Int)
29900: Proc WordWrap1Click( Sender : TObject)
29901: Proc Write( const AOut :Str)
29902: Proc Write( Socket : TSocket)
29903: Proc Write(S:Str);
29904: Proc WriteBinaryStream( const Section, Name :Str; Value : TStream)
29905: Proc WriteBool( const Section, Ident :Str; Value :Bool)
29906: Proc WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow :Bool)
29907: Proc WriteBuffer(Buffer:str;Count:LongInt)
29908: Proc WriteCardinal( AValue :Card; const AConvert :Bool)
29909: Proc WriteChar( AValue : Char)
29910: Proc WriteDate( const Section, Name :Str; Value : TDateTime)
29911: Proc WriteDateTime( const Section, Name :Str; Value : TDateTime)
29912: Proc WriteFloat( const Section, Name :Str; Value : Double)
29913: Proc WriteHeader( AHeader : TStrings)
29914: Proc WriteInt( AValue : Int; const AConvert :Bool)
29915: Proc WriteInt( const Section, Ident :Str; Value : Longint)
29916: Proc WriteLn( const AOut :Str)
29917: Proc Println( AOut :Str)
29918: Proc Writeln(s:Str);
29919: Proc WriteLog( const FileName, LogLine :Str)
29920: Proc WriteRFCReply( AReply : TIdRFCReply)
29921: Proc WriteRFCStrings( AStrings : TStrings)
29922: Proc WriteSmallInt( AValue : SmallInt; const AConvert :Bool)
29923: Proc WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCnt:Bool;const ASize:Int)
29924: Proc WriteString( const Section, Ident, Value :Str)
29925: Proc WriteStrings( AValue : TStrings; const AWriteLinesCount :Bool)
29926: Proc WriteTime( const Section, Name :Str; Value : TDateTime)
29927: Proc WriteObjectResourceHeader( ObjStream, Output : TStream)
29928: Proc Write16bitResourceHeader(const AName: TBytes; DataSize : Int; Output:TStream)
29929: Proc Write32bitResourceHeader(const AName: TBytes; DataSize : Int; Output:TStream)
29930: Proc WriteDataSetToCSV(DataSet: TDataSet; FileName:Str);
29931: Proc WStrSet(c : AnyString; I : Int; var s : AnyString);
29932: Proc XMLSyntax1Click(Sender: TObject);
29933: Proc XOR_Streams2( Dest, Srce : TMemoryStream)
29934: Proc XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
29935: Proc ZeroFillStream( Stream : TMemoryStream)
29936: Proc XMLSyntax1Click(Sender: TObject);
29937: Proc ZeroMemory( Ptr : Pointer; Length : Longint)
29938: procedure(Control: TWinControl; Index: Int; Rect: TRect; State: Byte)
29939: procedure(Control: TWinControl; Index: Int; var Height: Int)
29940: procedure(Sender,Source: TObject; X, Y: Int; State: TDragState; var Accept:Bool)
29941: procedure(Sender,Source: TObject;X, Y: Int)
29942: procedure(Sender,Target: TObject; X, Y: Int)
29943: procedure(Sender:TObject; ASection, AWidth: Int)
29944: procedure(Sender:TObject; ScrollCode: TScrollCode;var ScrollPos: Int)
29945: procedure(Sender:TObject; Shift: TShiftState; X, Y: Int);
29946: procedure(Sender:TObject; var Action: TCloseAction)
29947: procedure(Sender:TObject; var CanClose:Bool)
29948: procedure(Sender:TObject; var Key: Char);
29949: ProcedureName ProcedureNames ProcedureParametersCursor @
29950:
29951: *************Now Constructors constructor *************
29952:  Size is: 1727 1640 1588
29953:  Attach( VersionInfoData : Pointer; Size : Int)
29954:  constructor Create( ABuckets : TBucketListSizes)
29955:  Create(ACallBackWnd : HWND)
29956:  Create(AClient : TCustomTaskDialog)
29957:  Create(AClient : TIdTelnet)
29958:  Create(ACollection : TCollection)
29959:  Create(ACollection : TFavoriteLinkItems)
29960:  Create(ACollection : TTaskDialogButtons)
```

```
29961:  Create(AConnection : TIdCustomHTTP)
29962:  Create(ACreateSuspended :Bool)
29963:  Create(ADataSet : TCustomSQLDataSet)
29964:  CREATE(ADATASET : TDATASET)
29965:  Create(Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
29966:  Create(AGrid : TCustomDBGrid)
29967:  Create(AGrid : TStringGrid; AIndex : Longint)
29968:  Create(AHTTP : TIdCustomHTTP)
29969:  Create(AListItems : TListItems)
29970:  Create(AOnBytesRemoved : TIdBufferBytesRemoved)
29971:  Create(AOnBytesRemoved : TIdBufferBytesRemoved)
29972:  Create(AOwner : TCommonCalendar)
29973:  Create(AOwner : TComponent)
29974:  CREATE(AOWNER : TCOMPONENT)
29975:  Create(AOwner : TCustomListView)
29976:  Create(AOwner : TCustomOutline)
29977:  Create(AOwner : TCustomRichEdit)
29978:  Create(AOwner : TCustomRichEdit; AttributeType : TAttributeType)
29979:  Create(AOwner : TCustomTreeView)
29980:  Create(AOwner : TIdUserManager)
29981:  Create(AOwner : TListItems)
29982:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
29983:  CREATE(AOWNER : TPERSISTENT)
29984:  Create(AOwner : TPersistent)
29985:  Create(AOwner : TTable)
29986:  Create(AOwner : TTreeNodes)
29987:  Create(AOwner : TWinControl; const ClassName :Str)
29988:  Create(AParent : TIdCustomHTTP)
29989:  Create(AParent : TUpdateTree; AResolver : TCustomResolver)
29990:  Create(AProvider : TBaseProvider)
29991:  Create(AProvider : TCustomProvider);
29992:  Create(AProvider : TDataSetProvider)
29993:  Create(ASocket : TCustomWinSocket; TimeOut : Longint)
29994:  Create(ASocket : TSocket)
29995:  Create(AStrings : TWideStrings)
29996:  Create(AToolBar : TToolBar)
29997:  Create(ATreeNodes : TTreeNodes)
29998:  Create(Autofill :Bool)
29999:  Create(AWebPageInfo : TAbstractWebPageInfo)
30000:  Create(AWebRequest : TWebRequest)
30001:  Create(Collection : TCollection)
30002:  Create(Collection : TIdMessageParts; ABody : TStrings)
30003:  Create(Collection : TIdMessageParts; const AFileName : TFileName)
30004:  Create(Column : TColumn)
30005:  Create(const AConvFamily : TConvFamily; const ADescription :Str)
30006:  Create(const AConvFamily  TConvFamily;const ADescription:str;const AFactor: Double)
30007:  Create(const AConvFamily:TConvFamily;const ADescription:str;const AToCommonProc,
       AFromComProc:TConversionProc)
30008:  Create(const AInitialState :Bool; const AManualReset :Bool)
30009:  Create(const ATabSet : TTabSet)
30010:  Create(const Compensate :Bool)
30011:  Create(const FileMap:TJclCustomFileMapping; Access,Size :Card; ViewOffset : Int64)
30012:  Create(const FileName :Str)
30013:  Create(const FileName :Str;FileMode:Card; const Name:str; Protect:Card; const MaximumSize : Int64; const
       SecAttr : PSecurityAttributes);
30014:  Create(const FileName :Str; FileMode : WordfmShareDenyWrite)
30015:  Create(const MaskValue :Str)
30016:  Create(const Name:str; Protect:Card;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
30017:  Create(const Prefix :Str)
30018:  Create(const sRegularExpression :Str; xFlags : TniRegularExpressionMatchFlags)
30019:  Create(const sRule :Str; xFlags : TniRegularExpressionMatchFlags)
30020:  Create(const sRule :Str; xFlags : TniRegularExpressionMatchFlags)
30021:  Create(CoolBar : TCoolBar)
30022:  Create(CreateSuspended :Bool; ASocket : TServerClientWinSocket)
30023:  Create(CreateSuspended :Bool; ASocket : TServerWinSocket)
30024:  Create(DataSet:TDataSet;const Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const
       FieldName:Widestring;DepFields:TBits;FieldMap:TFieldMap)
30025:  Create(DBCtrlGrid : TDBCtrlGrid)
30026:  Create(DSTableProducer : TDSTableProducer)
30027:  Create(DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
30028:  Create(ErrorCode : DBIResult)
30029:  Create(Field : TBlobField; Mode : TBlobStreamMode)
30030:  Create(Grid : TCustomDBGrid; ColumnClass : TColumnClass)
30031:  Create(HeaderControl : TCustomHeaderControl)
30032:  Create(HTTPRequest : TWebRequest)
30033:  Create(iStart : Int; sText :Str)
30034:  Create(iValue : Int)
30035:  Create(Kind : TMmTimerKind; Notification : TMmNotificationKind)
30036:  Create(MciErrNo : MCIERROR; const Msg :Str)
30037:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
30038:  Create(Message :Str; ErrorCode : DBResult)
30039:  Create(Msg :Str)
30040:  Create(NativeError, Context :Str; ErrCode, PrevError : Int; E : Exception)
30041:  Create(NativeError, Context :Str; ErrorCode, PreviousError : DBResult)
30042:  Create(oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
30043:  Create(oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
30044: Create(oSorce:TniRegularExpressState;oDestinat:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
30045:  Create(Owner: EDBEngineError; ErrorCode: DBIResult; NativeError:Longint;Message: PChar)
30046:  Create(Owner: TCustomComboBoxEx)
```

```
30047:  CREATE(OWNER  TINDEXDEFS; const NAME, FIELDS:Str; OPTIONS: TINDEXOPTIONS)
30048:  Create(Owner: TPersistent)
30049:  Create(Params: TStrings) Create( Size :Card)
30050:  Create(Socket: TSocket; ServerWinSocket : TServerWinSocket)
30051:  Create(StatusBar: TCustomStatusBar)
30052:  Create(WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
30053:  Create(WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
30054:  Create(AHandle:Int)
30055:  Create(AOwner: TComponent); virtual;
30056:  Create(const AURI :Str)
30057:  Create(FileName:str;Mode:Word)
30058:  Create(Instance:THandle;ResName:str;ResType:PChar)
30059:  Create(Stream : TStream)
30060:  Create1( ADataset : TDataset);
30061:  Create1(const FileHandle:THandle;const Name:str;Protect:Card;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
30062:  Create1( const FileName :Str; const AIndexOption : TJclMappedTextReaderIndex);
30063:  Create2( Other : TObject);
30064:  CreateAt(FileMap: TJclCustomFileMapping;Access,Size:Card;ViewOffset:Int64;Address:Pointer)
30065:  CreateError(const anErrCode: Int;const asReplyMessage:Str; const asErrorMessage:Str)
30066:  CreateFmt( MciErrNo : MCIERROR; const Msg :Str; const Args : array of const)
30067:  CreateFromId(Instance:THandle;ResId:Int;ResType:PChar)
30068:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
30069:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Int)
30070:  CreateRes(Ident : Int);
30071:  CreateRes(MciErrNo : MCIERROR; Ident : Int)
30072:  CreateRes(ResStringRec : PResStringRec);
30073:  CreateResHelp( Ident : Int; AHelpContext : Int);
30074:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Int);
30075:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
30076:  CreateSize( AWidth, AHeight : Int)
30077:  Open(const Name:Str; const InheritHandle:Bool; const DesiredAccess :Card)
30078:
30079: ------------------------------------------------------------------------------
30080: unit uPSI_MathMax;
30081: ------------------------------------------------------------------------------
30082:  CONSTS
30083:   Bernstein: Float = 0.2801694990238691330364364912307;  // Bernstein constant
30084:   Cbrt2: Float     = 1.2599210498948731647672106072782;  // CubeRoot(2)
30085:   Cbrt3: Float     = 1.4422495703074083823216383107801;  // CubeRoot(3)
30086:   Cbrt10: Float    = 2.1544346900318837217592935665194;  // CubeRoot(10)
30087:   Cbrt100: Float   = 4.6415888336127788924100763509194;  // CubeRoot(100)
30088:   CbrtPi: Float    = 1.4645918875615232630201425272638;  // CubeRoot(PI)
30089:   Catalan: Float   = 0.9159655941772190150546035149324;  // Catalan constant
30090:   PiJ:   Float     = 3.1415926535897932384626433832795;  // PI
30091:   PI:  Extended =  = 3.1415926535897932384626433832795);
30092:   PiOn2: Float     = 1.5707963267948966192313216916398;  // PI / 2
30093:   PiOn3: Float     = 1.0471975511965977461542144610932;  // PI / 3
30094:   PiOn4: Float     = 0.7853981633974483096156608458198 8; // PI / 4
30095:   Sqrt2: Float     = 1.4142135623730950488016887242097;  // Sqrt(2)
30096:   Sqrt3: Float     = 1.7320508075688772935274463415059;  // Sqrt(3)
30097:   Sqrt5: Float     = 2.2360679774997896964091736687313;  // Sqrt(5)
30098:   Sqrt10: Float    = 3.1622776601683793319988935444327;  // Sqrt(10)
30099:   SqrtPi: Float    = 1.7724538509055160272981674833411;  // Sqrt(PI)
30100:   Sqrt2Pi: Float   = 2.5066282746310005024157652848 11;  // Sqrt(2 * PI)
30101:   TwoPi: Float     = 6.2831853071795864769252867665 59;  // 2 * PI
30102:   ThreePi: Float   = 9.4247779607693797153879301498385;  // 3 * PI
30103:   Ln2: Float       = 0.6931471805599453094172321214581 8; // Ln(2)
30104:   Ln10: Float      = 2.3025850929940456840179914546844;  // Ln(10)
30105:   LnPi: Float      = 1.1447298858494001741434273513531;  // Ln(PI)
30106:   Log2J: Float     = 0.3010299956639811952137388947244 9; // Log10(2)
30107:   Log3: Float      = 0.4771212547196624372950279032551 2; // Log10(3)
30108:   LogPi: Float     = 0.4971498726941338543512682882909;  // Log10(PI)
30109:   LogE: Float      = 0.4342944819032518276511289189166 1; // Log10(E)
30110:   E: Float         = 2.7182818284590452353602874713527;  // Natural constant
30111:   hLn2Pi: Float    = 0.9189385332046727417803297364056 2; // Ln(2*PI)/2
30112:   inv2Pi: Float    = 0.1591549430918953357688837633725143620344596457404 6; // 0.5/Pi
30113:   TwoToPower63: Float    = 9223372036854775808.0;            // 2^63
30114:   GoldenMean: Float      = 1.6180339887498948482045868343656 38;  // GoldenMean
30115:   EulerMascheroni: Float = 0.5772156649015328606065120900824;  // Euler GAMMA
30116:   RadCor : Double        = 57.29577951308232;    {number of degrees in a radian}
30117:   StDelta         : Extended = 0.00001;  {delta for difference equations}
30118:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
30119:   StMaxIterations : Int  = 100;      {max attempts for convergence}
30120:
30121: Proc SIRegister_StdConvs(CL: TPSPascalCompiler);
30122: begin
30123:  MetersPerInch = 0.0254; // [1]
30124:  MetersPerFoot = MetersPerInch * 12;
30125:  MetersPerYard = MetersPerFoot * 3;
30126:  MetersPerMile = MetersPerFoot * 5280;
30127:  MetersPerNauticalMiles = 1852;
30128:  MetersPerAstronomicalUnit = 1.49598E11; // [4]
30129:  MetersPerLightSecond = 2.99792458E8; // [5]
30130:  MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
30131:  MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
30132:  MetersPerCubit = 0.4572; // [6][7]
30133:  MetersPerFathom = MetersPerFoot * 6;
30134:  MetersPerFurlong = MetersPerYard * 220;
```

```
30135:  MetersPerHand = MetersPerInch * 4;
30136:  MetersPerPace = MetersPerInch * 30;
30137:  MetersPerRod = MetersPerFoot * 16.5;
30138:  MetersPerChain = MetersPerRod * 4;
30139:  MetersPerLink = MetersPerChain / 100;
30140:  MetersPerPoint = MetersPerInch * 0.013837; // [7]
30141:  MetersPerPica = MetersPerPoint * 12;
30142:
30143:  SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
30144:  SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
30145:  SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
30146:  SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
30147:  SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
30148:  SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
30149:  CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
30150:  CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
30151:  CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
30152:  CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
30153:  CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
30154:  CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
30155:  CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
30156:  CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
30157:
30158:  CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
30159:  CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
30160:  CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
30161:  CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
30162:  CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
30163:  CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
30164:  CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
30165:  CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
30166:  CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
30167:  CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
30168:  CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
30169:  CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
30170:  CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
30171:  CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
30172:  CubicMetersPerUKGallon = 0.00454609; // [2][7]
30173:  CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
30174:  CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
30175:  CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
30176:  CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
30177:  CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
30178:  CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
30179:  CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
30180:  CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
30181:
30182:  GramsPerPound = 453.59237; // [1][7]
30183:  GramsPerDrams = GramsPerPound / 256;
30184:  GramsPerGrains = GramsPerPound / 7000;
30185:  GramsPerTons = GramsPerPound * 2000;
30186:  GramsPerLongTons = GramsPerPound * 2240;
30187:  GramsPerOunces = GramsPerPound / 16;
30188:  GramsPerStones = GramsPerPound * 14;
30189:
30190:  MaxAngle 9223372036854775808.0;
30191:  MaxTanH 5678.2617031470719747459655389854);
30192:  MaxFactorial( 1754);
30193:  MaxFloatingPoint(1.189731495357231765085759326628E+4932);
30194:  MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
30195:  MaxTanH( 354.8913564466919984216228461865 9);
30196:  MaxFactorial'LongInt'( 170);
30197:  MaxFloatingPointD(1.79769313486231590772930519078 9E+308);
30198:  MinFloatingPointD(2.22507385850720138309023271733 24E-308);
30199:  MaxTanH( 44.36141955583649980270285577332 3);
30200:  MaxFactorial'LongInt'( 33);
30201:  MaxFloatingPointS( 3.40282366920938463463374607431 77E+38);
30202:  MinFloatingPointS( 1.17549435082228750796873653722 22E-38);
30203:  PiExt( 3.14159265358979323846264338327 95);
30204:  RatioDegToRad( PiExt / 180.0);
30205:  RatioGradToRad( PiExt / 200.0);
30206:  RatioDegToGrad( 200.0 / 180.0);
30207:  RatioGradToDeg( 180.0 / 200.0);
30208:  Crc16PolynomCCITT'LongWord $1021);
30209:  Crc16PolynomIBM'LongWord $8005);
30210:  Crc16Bits'LongInt'( 16);
30211:  Crc16Bytes'LongInt'( 2);
30212:  Crc16HighBit'LongWord $8000);
30213:  NotCrc16HighBit','LongWord $7FFF);
30214:  Crc32PolynomIEEE','LongWord $04C11DB7);
30215:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
30216:  Crc32Koopman','LongWord $741B8CD7);
30217:  Crc32Bits','LongInt'( 32);
30218:  Crc32Bytes','LongInt'( 4);
30219:  Crc32HighBit','LongWord $80000000);
30220:  NotCrc32HighBit','LongWord $7FFFFFFF);
30221:
30222:   MinByte       = Low(Byte);
30223:   MaxByte       = High(Byte);
```

```
30224:   MinWord       = Low(Word);
30225:   //MaxWord       = High(Word);
30226:   MaxWord2      = High(Word);
30227:   MinShortInt   = Low(ShortInt);
30228:   MaxShortInt   = High(ShortInt);
30229:   MinSmallInt   = Low(SmallInt);
30230:   MaxSmallInt   = High(SmallInt);
30231:   MinLongWord   = LongWord(Low(LongWord));
30232:   MaxLongWord   = LongWord(High(LongWord));
30233:   MinLongInt    = LongInt(Low(LongInt));
30234:   MaxLongInt    = LongInt(High(LongInt));
30235:   MinInt64      = Int64(Low(Int64));
30236:   MaxInt64      = Int64(High(Int64));
30237:   MinInt    = Int(Low(Int));
30238:   MaxInt    = Int(High(Int));
30239:   MinCardinal   = Cardinal(Low(Cardinal));
30240:   MaxCardinal   = Cardinal(High(Cardinal));
30241:   MinNativeUInt = NativeUInt(Low(NativeUInt));
30242:   MaxNativeUInt = NativeUInt(High(NativeUInt));
30243:   MinNativeInt  = NativeInt(Low(NativeInt));
30244:   MaxNativeInt  = NativeInt(High(NativeInt));
30245:  Func CosH( const Z : Float) : Float;
30246:  Func SinH( const Z : Float) : Float;
30247:  Func TanH( const Z : Float) : Float;
30248:
30249:  //***********from DMath.Dll Lib of types.inc in source\dmath_dll
30250:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
30251:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
30252:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
30253:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
30254:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
30255:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
30256:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
30257:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
30258:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
30259:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
30260:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
30261:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
30262:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
30263:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
30264:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
30265:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
30266:   MaxLog = 709.782712893384O;
30267:   MinLog = -708.3964185322641;
30268:   MaxFac = 170;
30269:   MaxGam = 171.624376956302;
30270:   MaxLgm = 2.556348E+305;
30271:   SingleCompareDelta   = 1.0E-34;
30272:   DoubleCompareDelta   = 1.0E-280;
30273:   {$IFDEF CLR}
30274:   ExtendedCompareDelta = DoubleCompareDelta;
30275:   {$ELSE}
30276:   ExtendedCompareDelta = 1.0E-4400;
30277:   {$ENDIF}
30278:   Bytes1KB  = 1024;
30279:   Bytes1MB  = 1024 * Bytes1KB;
30280:   Bytes1GB  = 1024 * Bytes1MB;
30281:   Bytes64KB = 64 * Bytes1KB;
30282:   Bytes64MB = 64 * Bytes1MB;
30283:   Bytes2GB  = 2 * LongWord(Bytes1GB);
30284:  clBlack32', $FF000000 ));
30285:  clDimGray32', $FF3F3F3F ));
30286:  clGray32', $FF7F7F7F ));
30287:  clLightGray32', $FFBFBFBF ));
30288:  clWhite32', $FFFFFFFF ));
30289:  clMaroon32', $FF7F0000 ));
30290:  clGreen32', $FF007F00 ));
30291:  clOlive32', $FF7F7F00 ));
30292:  clNavy32', $FF00007F ));
30293:  clPurple32', $FF7F007F ));
30294:  clTeal32', $FF007F7F ));
30295:  clRed32', $FFFF0000 ));
30296:  clLime32', $FF00FF00 ));
30297:  clYellow32', $FFFFFF00 ));
30298:  clBlue32', $FF0000FF ));
30299:  clFuchsia32', $FFFF00FF ));
30300:  clAqua32', $FF00FFFF ));
30301:  clAliceBlue32', $FFF0F8FF ));
30302:  clAntiqueWhite32', $FFFAEBD7 ));
30303:  clAquamarine32', $FF7FFFD4 ));
30304:  clAzure32', $FFF0FFFF ));
30305:  clBeige32', $FFF5F5DC ));
30306:  clBisque32', $FFFFE4C4 ));
30307:  clBlancheDalmond32', $FFFFEBCD ));
30308:  clBlueViolet32', $FF8A2BE2 ));
30309:  clBrown32', $FFA52A2A ));
30310:  clBurlyWood32', $FFDEB887 ));
30311:  clCadetblue32', $FF5F9EA0 ));
30312:  clChartReuse32', $FF7FFF00 ));
```

```
30313:  clChocolate32', $FFD2691E ));
30314:  clCoral32', $FFFF7F50 ));
30315:  clCornFlowerBlue32', $FF6495ED ));
30316:  clCornSilk32', $FFFFF8DC ));
30317:  clCrimson32', $FFDC143C ));
30318:  clDarkBlue32', $FF00008B ));
30319:  clDarkCyan32', $FF008B8B ));
30320:  clDarkGoldenRod32', $FFB8860B ));
30321:  clDarkGray32', $FFA9A9A9 ));
30322:  clDarkGreen32', $FF006400 ));
30323:  clDarkGrey32', $FFA9A9A9 ));
30324:  clDarkKhaki32', $FFBDB76B ));
30325:  clDarkMagenta32', $FF8B008B ));
30326:  clDarkOliveGreen32', $FF556B2F ));
30327:  clDarkOrange32', $FFFF8C00 ));
30328:  clDarkOrchid32', $FF9932CC ));
30329:  clDarkRed32', $FF8B0000 ));
30330:  clDarkSalmon32', $FFE9967A ));
30331:  clDarkSeaGreen32', $FF8FBC8F ));
30332:  clDarkSlateBlue32', $FF483D8B ));
30333:  clDarkSlateGray32', $FF2F4F4F ));
30334:  clDarkSlateGrey32', $FF2F4F4F ));
30335:  clDarkTurquoise32', $FF00CED1 ));
30336:  clDarkViolet32', $FF9400D3 ));
30337:  clDeepPink32', $FFFF1493 ));
30338:  clDeepSkyBlue32', $FF00BFFF ));
30339:  clDodgerBlue32', $FF1E90FF ));
30340:  clFireBrick32', $FFB22222 ));
30341:  clFloralWhite32', $FFFFFAF0 ));
30342:  clGainsBoro32', $FFDCDCDC ));
30343:  clGhostWhite32', $FFF8F8FF ));
30344:  clGold32', $FFFFD700 ));
30345:  clGoldenRod32', $FFDAA520 ));
30346:  clGreenYellow32', $FFADFF2F ));
30347:  clGrey32', $FF808080 ));
30348:  clHoneyDew32', $FFF0FFF0 ));
30349:  clHotPink32', $FFFF69B4 ));
30350:  clIndianRed32', $FFCD5C5C ));
30351:  clIndigo32', $FF4B0082 ));
30352:  clIvory32', $FFFFFFF0 ));
30353:  clKhaki32', $FFF0E68C ));
30354:  clLavender32', $FFE6E6FA ));
30355:  clLavenderBlush32', $FFFFF0F5 ));
30356:  clLawnGreen32', $FF7CFC00 ));
30357:  clLemonChiffon32', $FFFFFACD ));
30358:  clLightBlue32', $FFADD8E6 ));
30359:  clLightCoral32', $FFF08080 ));
30360:  clLightCyan32', $FFE0FFFF ));
30361:  clLightGoldenRodYellow32', $FFFAFAD2 ));
30362:  clLightGreen32', $FF90EE90 ));
30363:  clLightGrey32', $FFD3D3D3 ));
30364:  clLightPink32', $FFFFB6C1 ));
30365:  clLightSalmon32', $FFFFA07A ));
30366:  clLightSeagreen32', $FF20B2AA ));
30367:  clLightSkyblue32', $FF87CEFA ));
30368:  clLightSlategray32', $FF778899 ));
30369:  clLightSlategrey32', $FF778899 ));
30370:  clLightSteelblue32', $FFB0C4DE ));
30371:  clLightYellow32', $FFFFFFE0 ));
30372:  clLtGray32', $FFC0C0C0 ));
30373:  clMedGray32', $FFA0A0A4 ));
30374:  clDkGray32', $FF808080 ));
30375:  clMoneyGreen32', $FFC0DCC0 ));
30376:  clLegacySkyBlue32', $FFA6CAF0 ));
30377:  clCream32', $FFFFFBF0 ));
30378:  clLimeGreen32', $FF32CD32 ));
30379:  clLinen32', $FFFAF0E6 ));
30380:  clMediumAquamarine32', $FF66CDAA ));
30381:  clMediumBlue32', $FF0000CD ));
30382:  clMediumOrchid32', $FFBA55D3 ));
30383:  clMediumPurple32', $FF9370DB ));
30384:  clMediumSeaGreen32', $FF3CB371 ));
30385:  clMediumSlateBlue32', $FF7B68EE ));
30386:  clMediumSpringGreen32', $FF00FA9A ));
30387:  clMediumTurquoise32', $FF48D1CC ));
30388:  clMediumVioletRed32', $FFC71585 ));
30389:  clMidnightBlue32', $FF191970 ));
30390:  clMintCream32', $FFF5FFFA ));
30391:  clMistyRose32', $FFFFE4E1 ));
30392:  clMoccasin32', $FFFFE4B5 ));
30393:  clNavajoWhite32', $FFFFDEAD ));
30394:  clOldLace32', $FFFDF5E6 ));
30395:  clOliveDrab32', $FF6B8E23 ));
30396:  clOrange32', $FFFFA500 ));
30397:  clOrangeRed32', $FFFF4500 ));
30398:  clOrchid32', $FFDA70D6 ));
30399:  clPaleGoldenRod32', $FFEEE8AA ));
30400:  clPaleGreen32', $FF98FB98 ));
30401:  clPaleTurquoise32', $FFAFEEEE ));
```

```
30402:  clPaleVioletred32', $FFDB7093 ));
30403:  clPapayaWhip32', $FFFFEFD5 ));
30404:  clPeachPuff32', $FFFFDAB9 ));
30405:  clPeru32', $FFCD853F ));
30406:  clPlum32', $FFDDA0DD ));
30407:  clPowderBlue32', $FFB0E0E6 ));
30408:  clRosyBrown32', $FFBC8F8F ));
30409:  clRoyalBlue32', $FF4169E1 ));
30410:  clSaddleBrown32', $FF8B4513 ));
30411:  clSalmon32', $FFFA8072 ));
30412:  clSandyBrown32', $FFF4A460 ));
30413:  clSeaGreen32', $FF2E8B57 ));
30414:  clSeaShell32', $FFFFF5EE ));
30415:  clSienna32', $FFA0522D ));
30416:  clSilver32', $FFC0C0C0 ));
30417:  clSkyblue32', $FF87CEEB ));
30418:  clSlateBlue32', $FF6A5ACD ));
30419:  clSlateGray32', $FF708090 ));
30420:  clSlateGrey32', $FF708090 ));
30421:  clSnow32', $FFFFFAFA ));
30422:  clSpringgreen32', $FF00FF7F ));
30423:  clSteelblue32', $FF4682B4 ));
30424:  clTan32', $FFD2B48C ));
30425:  clThistle32', $FFD8BFD8 ));
30426:  clTomato32', $FFFF6347 ));
30427:  clTurquoise32', $FF40E0D0 ));
30428:  clViolet32', $FFEE82EE ));
30429:  clWheat32', $FFF5DEB3 ));
30430:  clWhitesmoke32', $FFF5F5F5 ));
30431:  clYellowgreen32', $FF9ACD32 ));
30432:  clTrWhite32', $7FFFFFFF ));
30433:  clTrBlack32', $7F000000 ));
30434:  clTrRed32', $7FFF0000 ));
30435:  clTrGreen32', $7F00FF00 ));
30436:  clTrBlue32', $7F0000FF ));
30437:   // Fixed point math constants
30438:   FixedOne = $10000;  FixedHalf = $7FFF;
30439:   FixedPI  = Round(PI * FixedOne);
30440:   FixedToFloat = 1/FixedOne;
30441:
30442:  Special Types
30443: *****************************************************
30444:   type Complex = record //for complex numbers
30445:     X, Y : Float; end;
30446:   type TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R :'
30447:    + Float; Theta : Float; end;
30448:   type TVector     = array of Float;
30449:   TIntVector  = array of Int;
30450:   TCompVector = array of Complex;
30451:   TBoolVector = array of Boolean;
30452:   TStrVector  = array of String;
30453:   TMatrix     = array of TVector;
30454:   TIntMatrix  = array of TIntVector;
30455:   TCompMatrix = array of TCompVector;
30456:   TBoolMatrix = array of TBoolVector;
30457:   TStrMatrix  = array of TStrVector;
30458:   TByteArray = array[0..32767] of byte; !
30459:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
30460:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
30461:   T2StringArray = array of array of string;
30462:   T2IntArray = array of array of Int;
30463:   AddTypeS('INT_PTR', 'Int
30464:   AddTypeS('LONG_PTR', 'Int
30465:   AddTypeS('UINT_PTR', 'Cardinal
30466:   AddTypeS('ULONG_PTR', 'Cardinal
30467:   AddTypeS('DWORD_PTR', 'ULONG_PTR
30468:   TIntDynArray', 'array of Int
30469:   TCardinalDynArray', 'array of Cardinal
30470:   TWordDynArray', 'array of Word
30471:   TSmallIntDynArray', 'array of SmallInt
30472:   TByteDynArray', 'array of Byte
30473:   TShortIntDynArray', 'array of ShortInt
30474:   TInt64DynArray', 'array of Int64
30475:   TLongWordDynArray', 'array of LongWord
30476:   TSingleDynArray', 'array of Single
30477:   TDoubleDynArray', 'array of Double
30478:   TBooleanDynArray', 'array of Boolean
30479:   TStringDynArray', 'array of string
30480:   TWideStringDynArray', 'array of WideString
30481:   TDynByteArray     = array of Byte;
30482:   TDynShortintArray = array of Shortint;
30483:   TDynSmallintArray = array of Smallint;
30484:   TDynWordArray     = array of Word;
30485:   TDynIntArray  = array of Int;
30486:   TDynLongintArray  = array of Longint;
30487:   TDynCardinalArray = array of Cardinal;
30488:   TDynInt64Array    = array of Int64;
30489:   TDynExtendedArray = array of Extended;
30490:   TDynDoubleArray   = array of Double;
```

```
30491:   TDynSingleArray   = array of Single;
30492:   TDynFloatArray    = array of Float;
30493:   TDynPointerArray  = array of Pointer;
30494:   TDynStringArray   = array of string;
30495:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
30496:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
30497:   TSynSearchOptions = set of TSynSearchOption;
30498:   TFloat = single
30499:   Float = double
30500:
30501: /* Project:IFSI_WinForm1puzzle.pas BaseInclude RunTimeLib for maXbox *: pas_includebox.inc
30502: ---------------------------------------------------------------------------
30503: Proc drawPolygon(vPoints: TXYVector; cFrm: TForm);
30504: Proc drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: Int);
30505: Proc SaveCanvas(vCanvas: TCanvas; FileName:Str);
30506: Proc SaveCanvas2(vCanvas: TCanvas; FileName:Str);
30507: Func CheckStringSum(vstring:Str): Int;
30508: Func HexToInt(HexNum:Str): LongInt;
30509: Func IntToBin(Int: Int):Str;
30510: Func BinToInt(Binary:Str): Int;
30511: Func HexToBin(HexNum:Str):Str; external2
30512: Func BinToHex(Binary:Str):Str;
30513: Func IntToFloat(i: Int): double;
30514: Func AddThousandSeparator(S:Str; myChr: Char):Str;
30515: Func Max3(const X,Y,Z: Int): Int;
30516: Proc Swap(var X,Y: char); // faster without inline
30517: Proc ReverseString(var S:Str);
30518: Func CharToHexStr(Value: Char):Str;
30519: Func CharToUniCode(Value: Char):Str;
30520: Func Hex2Dec(Value: Str002): Byte;
30521: Func HexStrCodeToStr(Value:Str):Str;
30522: Func HexToStr(i: Int; value:Str):Str;
30523: Func UniCodeToStr(Value:Str):Str;
30524: Func CRC16(statement:Str):Str;
30525: Func SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2:Str):Str;
30526: Proc SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr:Str);
30527: Proc SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr:Str; offset: Int);
30528: Proc ExecuteCommand(executeFile, paramstring:Str);
30529: Proc ShellExecuteAndWait(executeFile, paramstring:Str);
30530: Func ExecConsoleApp(const AppName, Parameters:Str; AppOutput: TStrings): DWORD;
30531: Proc SearchAndOpenDoc(vfilenamepath:Str);
30532: Proc ShowInterfaces(myFile:Str);
30533: Func Fact2(av: Int): extended;
30534: Func BoolToStr(B:Bool):Str;
30535: Func GCD(x, y : LongInt) : LongInt;
30536: Func LCM(m,n: longint): longint;
30537: Func GetASCII:Str;
30538: Func GetItemHeight(Font: TFont): Int;
30539: Func myPlaySound(s: pchar; flag,syncflag: Int):Bool;
30540: Func myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
30541: Func getHINSTANCE: longword;
30542: Func getHMODULE: longword;
30543: Func GetASCII:Str;
30544: Func ByteIsOk(const AByte:Str; var VB: Byte):Bool;
30545: Func WordIsOk(const AWord:Str; var VW: Word):Bool;
30546: Func TwentyFourBitValueIsOk(const AValue:Str; var VI: Int):Bool;
30547: Func LongIsOk(const ALong:Str; var VC:Card):Bool;
30548: Func SafeStr(const s:Str):Str;
30549: Func ExtractUrlPath(const FileName:Str):Str;
30550: Func ExtractUrlName(const FileName:Str):Str;
30551: Func IsInternet:Bool;
30552: Func RotateLeft1Bit_u32( Value: uint32): uint32;
30553: Proc LinearRegression(const KnownY:array of Double;const KnownX:array of Double;NData:Int;var
       LF:TStLinEst;ErrorStats:Bool);
30554: Proc getEnvironmentInfo;
30555: Proc AntiFreeze;
30556: Func GetCPUSpeed: Double;
30557: Func IsVirtualPcGuest :Bool;
30558: Func IsVmWareGuest :Bool;
30559: Proc StartSerialDialog;
30560: Func IsWoW64:Bool;
30561: Func IsWow64String(var s:Str):Bool;
30562: Proc StartThreadDemo;
30563: Func RGB(R,G,B: Byte): TColor;
30564: Func Sendln(amess:Str):Bool;
30565: Proc maXbox;
30566: Func AspectRatio(aWidth, aHeight: Int):Str;
30567: Func wget(aURL, afile:Str):Bool;
30568: Proc PrintList(Value: TStringList);
30569: Proc PrintImage(aValue: TBitmap; Style: TBitmapStyle);
30570: Proc getEnvironmentInfo;
30571: Proc AntiFreeze;
30572: Func getBitmap(apath:Str): TBitmap;
30573: Proc ShowMessageBig(const aText :Str);
30574: Func YesNoDialog(const ACaption, AMsg:Str):Bool;
30575: Proc SetArrayLength2String(arr: T2StringArray; asize1, asize2: Int);
30576: Proc SetArrayLength2Int(arr: T2IntArray; asize1, asize2: Int);
30577: //Func myStrToBytes(const Value:Str): TBytes;
30578: //Func myBytesToStr(const Value: TBytes):Str;
```

```
30579: Func SaveAsExcelFile(AGrid:TStringGrid;ASheetName,AFileName:str;open:boolean):Bool;
30580: Func getBitmap(apath:Str): TBitmap;
30581: Proc ShowMessageBig(const aText :Str);
30582: Func StrToBytes(const Value:Str): TBytes;
30583: Func BytesToStr(const Value: TBytes):Str;
30584: Func SaveAsExcelFile(AGrid: TStringGrid;ASheetName,AFileName:Str;open:boolean):Bool;
30585: Func ReverseDNSLookup(const IPAdrs:str;const DNSServer:str;Timeout,Retris:Int;var HostName:str):Bool;
30586: Func FindInPaths(const fileName, paths :Str) :Str;
30587: Proc initHexArray(var hexn: THexArray);
30588: Func josephusG(n,k: Int; var graphout:Str): Int;
30589: Func isPowerof2(num: int64):Bool;
30590: Func powerOf2(exponent: Int): int64;
30591: Func getBigPI:Str;
30592: Proc MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
30593:  Func GetASCIILine:Str;
30594: Proc MakeComplexSound(N:Int{stream # to use};freqlist:TStrings;Duration{mSec}: Int;
30595:                          pinknoise:Bool; shape: Int; Volume: TVolumeLevel);
30596: Proc SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:Int);
30597: Proc AddComplexSoundObjectToList(newf,newp,newa,news:Int; freqlist: TStrings);
30598: Func mapfunc(ax, in_min, in_max, out_min, out_max: Int): Int;
30599: Func mapmax(ax, in_min, in_max, out_min, out_max: Int): Int;
30600: Func isKeypressed:Bool;
30601: Func Keypress:Bool;
30602: Proc StrSplitP(const Delimiter: Char; Input:Str; const Strings: TStrings);
30603: Func ReadReg(Base: HKEY; KeyName, ValueName:Str):Str;
30604: Func ReadRegistry(Base: HKEY; KeyName, ValueName:Str):Str;
30605: Func GetOSName:Str;
30606: Func GetOSVersion:Str;
30607: Func GetOSNumber:Str;
30608: Func getEnvironmentString:Str;
30609: Proc StrReplace(var Str:Str; Old, New:Str);
30610: Proc SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
30611: Func getTeamViewerID:Str;
30612: Proc RecurseDirectory(Dir:str; IncludeSubs:boolean; callback:TFileCallbackProcedure);
30613: Proc RecurseDirectory2(Dir :Str; IncludeSubs :Bool);
30614: Proc WinInet_HttpGet(const Url:Str; Stream:TStream);
30615: Proc GetQrCode2(Width,Height: Word;Correct_Level:str;const Data:str;apath:str);
30616: Func StartSocketService:Bool;
30617: Proc StartSocketServiceForm;
30618: Func GetFileList(FileList: TStringlist; apath:Str): TStringlist;
30619: Func GetFileList1(apath:Str): TStringlist;
30620: Proc LetFileList(FileList: TStringlist; apath:Str);
30621: Proc StartWeb(aurl:Str);
30622: Func GetTodayFiles(startdir, amask:Str): TStringlist;
30623: Func PortTCPIsOpen(dwPort : Word; ipAddressStr:Str):Bool;
30624: Func JavahashCode(val:Str): Int;
30625: Proc PostKeyEx32(key: Word; const shift: TShiftState; specialkey:Bool);
30626: Proc SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName:Str);
30627: Proc HideWindowForSeconds(secs: Int);   {//3 seconds}
30628: Proc HideWindowForSeconds2(secs: Int; apphandle, aself: TForm);   {//3 seconds}
30629: Proc ConvertToGray(Cnv: TCanvas);
30630: Func GetFileDate(aFile:str; aWithTime:Boolean):str;
30631: Proc ShowMemory;
30632: Func ShowMemory2:Str;
30633: Func getHostIP:Str;
30634: Proc ShowBitmap(bmap: TBitmap);
30635: Func GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
30636: Func CreateDBGridForm(dblist: TStringList): TListbox;
30637: Func isService:Bool;
30638: Func isApplication:Bool;
30639: Func isTerminalSession:Bool;
30640:  Func SetPrivilege(privilegeName:Str; enable:Bool):Bool;
30641:  Proc getScriptandRunAsk;
30642:  Proc getScriptandRun(ascript:Str);
30643:  Func VersionCheckAct:Str;
30644:  Proc getBox(aurl, extension:Str);
30645:  Func CheckBox:Str;
30646:  Func isNTFS:Bool;
30647:  //Proc doWebCamPic;
30648:  Proc doWebCamPic(picname:Str);
30649:  Func readm:Str;
30650:  Proc getGEOMapandRunAsk;     //APIKey needed
30651:  Func GetMapX(C_form,apath:Str; const Data:Str):Bool;
30652:  Proc GetGEOMap(C_form,apath:Str; const Data:Str);
30653:  Func GetMapXGeoReverse(C_form:Str; const lat,long:Str):Str;
30654:  //Func RoundTo(const AValue:Extended; const ADigit: TRoundToEXRangeExtended):Extended;
30655:  Func GetGeocodeCoord(C_form:Str; const data:str; atxt:Bool):Str;
30656:  Func GetGeoCoord(C_form:Str; const data:str; atxt:Bool):Str;
30657:     ex.: writeln(GetGeoCoord('xml','church cefalu sicily',true))
30658:  Func DownloadFile(SourceFile, DestFile:Str):Bool;
30659:  Func DownloadFileOpen(SourceFile, DestFile:Str):Bool;
30660:  Func OpenMap(const Data:Str):Bool;
30661:  Func GetGeoCode(C_form,apath:Str; const data:Str; sfile:Bool):Str;
30662:  Func getFileCount(amask:Str): Int;
30663:  Func CoordinateStr(Idx: Int; PosInSec: Double; PosLn: TNavPos):Str;
30664:  Proc Debugln(DebugLOGFILE:Str; E:Str);
30665:  Func IntToFloat(i: Int): double;
30666:  Func AddThousandSeparator(S:Str; myChr: Char):Str;
30667:  Func mymciSendString(cmd: PChar; ret: PChar; len: Int; callback: Int):Card;
```

```
30668:  Func RoundTime(ADate:Str; Rounding: Int; bRound:Bool):Str;
30669:  Func DynamicDllCallName(Dll:Str; const Name:Str; HasResult:Bool; var Returned:Card; const Parameters:
       array of Int):Bool;;
30670:  Func DynamicDllCall(Dll:Str; const Name:str; HasResult:Boolean; var Returned:Card; const Parameters:
       array of Int):Bool;;
30671:  Func GetSpecialFolderID( const Path :Str) : Int;
30672:  Func GetSpecialFolderPath2( FolderID : Int) :Str;
30673: End C:\maXbook\maxbox3\mX3999\maxbox3\source\IFSI_WinForm1puzzle.pas File loaded
30674:
30675: // News of 3.9.8 up
30676: Halt-Stop Program in Menu,WebServer2,Stop Event Recompile,
30677: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
30678: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
30679: JvChart - TJvChart Component - 2009 Public
30680: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
30681: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
30682: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
30683: DMath DLL included incl. Demos
30684: Interface Navigator menu/View/Intf Navigator
30685: Unit Explorer menu/Debug/Units Explorer
30686: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
30687: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
30688: Script History to 9 Files WebServer light /Options/Addons/WebServer
30689: Full Text Finder, JVSimLogic Simulator Package
30690: Halt-Stop Program in Menu, WebServer2, Stop Event ,
30691: Conversion Routines, Prebuild Forms, CodeSearch
30692: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
30693: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
30694: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
30695: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
30696: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
30697: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
30698: IDE Reflection API, Session Service Shell S3
30699: additional SynEdit API, isKeyPressed Routine,Bookmarks,OpenToolsAPI Catalog (OTAC)
30700: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
30701: arduino map() function, PMRandom Generator
30702: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
30703: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
30704: REST Test Lib, Multilang Component, Forth Interpreter
30705: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
30706: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
30707: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
30708: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
30709: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
30710: QRCode Service, add more CFunctions like CDateTime of Synapse
30711: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
30712: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
30713: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
30714: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
30715: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
30716: BOLD Package, Indy Package5, maTRIx. MATHEMAX
30717: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
30718: emax layers: system-package-component-unit-class-function-block
30719: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
30720: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
30721: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
30722: OpenGL Game Demo: ..Options/Add Ons/Reversi
30723: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
30724: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
30725: 7% performance gain (hot spot profiling)
30726: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
30727: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
30728: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
30729: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
30730:
30731: add routines in 3.9.7.5
30732: 097: Proc RIRegister_BarCodeScaner_Routines(S: TPSExec);
30733: 996: Proc RIRegister_DBCtrls_Routines(S: TPSExec);
30734: 069: Proc RIRegister_IdStrings_Routines(S: TPSExec);
30735: 516: Proc RIRegister_JclMultimedia_Routines(S: TPSExec);
30736: 215: Proc RIRegister_PNGLoader_Routines(S: TPSExec);
30737: 374: Proc RIRegister_SerDlgs_Routines(S: TPSExec);
30738: 777: Proc RIRegister_LinarBitmap_Routines(S: TPSExec);
30739: 1216 Proc RIRegistger_uPSI_KDialogs, TKBrowseFolderDialog;
30740:
30741: ////////////////////////// TestUnits //////////////////////////
30742: Proc  SelftestPEM;
30743:   SelfTestCFundamentUtils;
30744:   SelfTestCFileUtils;
30745:   SelfTestCDateTime;
30746:   SelfTestCTimer;
30747:   SelfTestCRandom;
30748:   SelftestAES;
30749:   SelfTestASN1;
30750:   SelfTestX509;
30751:   TestDes:Bool;
30752:   Test3Des:Bool;
30753:   TestAes:Bool;
30754:   SelfTestcTLSUtils;
```

```
30755:   SelfTestCFundamentUtils;
30756:   SelfTestcHTTPUtils
30757:   SelfTestcXMLFunctions
30758:   SelfTestHugeWord
30759:   SelfTestRSA
30760:
30761:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
30762:            Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
30763:   //Note: There's no need for installing a client certificate in the
30764:   //       webbrowser. The server asks the webbrowser to send a certificate but
30765:   //       if nothing is installed the software will work because the server
30766:   //       doesn't check to see if a client certificate was supplied. If you want you can install: file:
c_cacert.p12  password: c_cakey
30767:
30768:   TKObject = class(TObject)
30769:   private
30770:     FParent: TKObjectList;
30771:     Proc SetParent(const Value: TKObjectList);
30772:   protected
30773:     FUpdateLock: Int;
30774:     Proc CallBeforeUpdate; virtual;
30775:     Proc CallAfterUpdate; virtual;
30776:     Proc ParentChanged; virtual;
30777:   public
30778:     constructor Create; virtual;
30779:     Proc Assign(ASource: TKObject); virtual;
30780:     Func EqualProperties(AValue: TKObject):Bool; virtual;
30781:     Proc LockUpdate; virtual;
30782:     Proc UnLockUpdate; virtual;
30783:     Func UpdateUnlocked:Bool; virtual;
30784:     property Parent: TKObjectList read FParent write SetParent;
30785:   end;
30786:
30787:   TKObjectClass = class of TKObject;
30788:   TKObjectList = class(TObjectList)
30789:   protected
30790:     FUpdateLock: Int;
30791:     Proc CallBeforeUpdate; virtual;
30792:     Proc CallAfterUpdate; virtual;
30793:   public
30794:     constructor Create; virtual;
30795:     Func Add(AObject: TObject): Int;
30796:     Proc Assign(ASource: TKObjectList); virtual;
30797:     Func EqualProperties(AValue: TKObjectList):Bool; virtual;
30798:     Proc Insert(Index: Int; AObject: TObject);
30799:     Proc LockUpdate; virtual;
30800:     Proc UnLockUpdate; virtual;
30801:     Func UpdateUnlocked:Bool; virtual;
30802:   end;
30803:
30804:   TGraphicControl = class(TControl)
30805:   private
30806:     FCanvas: TCanvas;
30807:     Proc WMPaint(var Message: TWMPaint); message WM_PAINT;
30808:   protected
30809:     Proc Paint; virtual;
30810:     property Canvas: TCanvas read FCanvas;
30811:   public
30812:     constructor Create(AOwner: TComponent); override;
30813:     destructor Destroy; override;
30814:   end;
30815:
30816:   TCustomControl = class(TWinControl)
30817:   private
30818:     FCanvas: TCanvas;
30819:     Proc WMPaint(var Message: TWMPaint); message WM_PAINT;
30820:   protected
30821:     Proc Paint; virtual;
30822:     Proc PaintWindow(DC: HDC); override;
30823:     property Canvas: TCanvas read FCanvas;
30824:   public
30825:     constructor Create(AOwner: TComponent); override;
30826:     destructor Destroy; override;
30827:   end;
30828:     RegisterPublishedProperties;
30829:     ('ONCHANGE', 'TNotifyEvent', iptrw);
30830:     ('ONCLICK', 'TNotifyEvent', iptrw);
30831:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
30832:     ('ONENTER', 'TNotifyEvent', iptrw);
30833:     ('ONEXIT', 'TNotifyEvent', iptrw);
30834:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
30835:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
30836:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
30837:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
30838:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
30839: //*********************************************************************
30840:   // To stop the while loop, click on Options/Show Include (boolean switch)!
30841:   Control a loop in a script with a form event:
30842:   IncludeON;   //control the while loop
```

```
30843:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
30844:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
30845:   include example:
30846:   {$I .\web\mX47520\flcMaths.inc}
30847:
30848: //-------------------------------------------------------------------------
30849: //**************mX4 ini-file Configuration*************************************
30850: //-------------------------------------------------------------------------------
30851:  using config file maxboxdef.ini       menu/Help/Config File
30852:
30853: //*** Definitions for maXbox mX47 ***
30854: [FORM]
30855: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
30856: FONTSIZE=14
30857: EXTENSION=txt
30858: SCREENX=1386
30859: SCREENY=1077
30860: MEMHEIGHT=350
30861: PRINTFONT=Courier New //GUI Settings
30862: FONTNAME= Courier New //GUI Settings for editor
30863: LINENUMBERS=Y   //line numbers at gutter in editor at left side
30864: EXCEPTIONLOG=Y  //store excepts+success in 2 log files see below! -menu Debug/Show Last Exceptions
30865: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
30866: BOOTSCRIPT=Y    //enabling load a boot script
30867: MEMORYREPORT=Y  //shows memory report on closing maXbox
30868: MACRO=Y         //expand macros (see below) incode e.g. #path:C:\maXbox\works2021\maxbox4\docs\
30869: NAVIGATOR=N     //shows Func list at the right side of editor
30870: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
30871: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
30872: DEP=Y           // activate Data Execution Prevention
30873: INDENT=Y        //shows indent vertical lines! (prepare for code folding) stop it with=N
30874: [WEB]
30875: IPPORT=8080     //for internal webserver - menu /Options/Add Ons/WebServer
30876: IPHOST=192.168.1.53  //run as Administrator!
30877: ROOTCERT=filepathY
30878: SCERT=filepathY
30879: RSAKEY=filepathY
30880: VERSIONCHECK=Y  //set to =N for better starttime and less web traffic!
30881: APP=C:\WINDOWS\System32\calc.exe  //set path to an external app
30882: MYSCRIPT=E:\maxbox4\mXGit39991\examples\330_myclock.txt //start script of menu /View/MyScript
30883:
30884: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
30885: Also possible to set report memory in script to override ini setting
30886: Proc Set_ReportMemoryLeaksOnShutdown(abo:Bool)
30887:
30888: After Change the ini file you can reload the file with ../Help/Config Update
30889: //-----------------------------------------------------------------------------
30890: //**************mX4 maildef.ini ini-file Configuration********************
30891: //-------------------------------------------------------------------------
30892: //*** Definitions for maXMail ***
30893: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
30894: [MAXMAIL]
30895: HOST=getmail.softwareschule.ch
30896: USER=mailusername
30897: PASS=password
30898: PORT=110
30899: SSL=Y
30900: BODY=Y
30901: LAST=5
30902:
30903: Command Line Interface CLI   -f-s-m-r-t as ParamStr(2); compile with -c
30904: ---------------------------------------------------------
30905: Ex.: S_ShellExecute(ExtractFilePath(ParamStr(0))+'maXbox4.exe',
30906:          myscript + ParamStr(2),seCmdOpen)
30907: Ex.: C:\maXbox\mx4\mx4_v38\mx4\source>maxbox4_7.exe -c examples\440_xml_tutor2.txt
30908:
30909:   if (ParamStr(2) = '-f') then begin
30910:        Application.BringToFront;  //maximize?
30911:        //maxform1.Show;
30912:      end;
30913:      if (ParamStr(2) = '-s') then begin
30914:          maxform1.Show;
30915:      end;
30916:      Compile1Click(self);
30917:      maxform1.memo2.lines.add('CLI Console Call Log at: ' +DateTimeToStr(Now));
30918:      hlog.Add('>>>> Start Console Call Exe: {App_name} v{App_ver}{80@}{now}');
30919:      if (ParamStr(2) = 'm') then begin
30920:        //Compile1Click(self);!
30921:        Application.Minimize;
30922:      end;
30923:     if (ParamStr(2) = 'r') then begin
30924:        Application.run;
30925:      end;
30926:       if (ParamStr(2) = '-t') then begin     //new4
30927:        Application.terminate;
30928:      end;
30929: ---------------------------------------------------------
30930: S_ShellExecute(ExePath+'maXbox4.exe',
30931:  ExePath+'examples\773_streamfibonacci_first.txt t',secmdopen); -terminate
```

```
30932: ADO Connection String:
30933: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
30934: \452_dbtreeview2access.txt \452_dbtrv3accessUML2.txt
30935: program TestDbTreeViewMainForm2_ACCESS;
30936:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
30937:           +Exepath+'\examples\detail.mdb;Persist Security Info=False';
30938:  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;"'
30939:
30940: Func CreateAccessDatabase(FileName:Str):Str;
30941: var cat: OLEVariant;
30942: begin
30943:   Result:= '';
30944:   try
30945:     cat:= CreateOleObject('ADOX.Catalog');
30946:     cat.Create('Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + FileName + ';
30947:     cat:= NULL;
30948:   except
30949:     writeln(ExceptionToString(ExceptionType, ExceptionParam));
30950:     writeln('ADOX.Catalog create failed ');
30951:     //on e: Exception do Result := e.message;
30952:   end;
30953: end;
30954:
30955: ADOCommand1.CommandText:= 'CREATE TABLE TABLE1 ('
30956:                            +'ID AUTOINCREMENT,'
30957:                            +'FirstName varchar(255) NOT NULL,'
30958:                            +'LastName varchar(255),'
30959:                            +'Age int )';
30960:
30961: OpenSSL Lib:  unit ssl_openssl_lib;
30962: {$IFDEF CIL}
30963: const
30964:   {$IFDEF LINUX}
30965:   DLLSSLName = 'libssl.so';
30966:   DLLUtilName = 'libcrypto.so';
30967:   {$ELSE}
30968:   DLLSSLName = 'ssleay32.dll';
30969:   DLLUtilName = 'libeay32.dll';
30970:   {$ENDIF}
30971: {$ELSE}
30972: var
30973:   {$IFNDEF MSWINDOWS}
30974:     {$IFDEF DARWIN}
30975:     DLLSSLName:Str = 'libssl.dylib';
30976:     DLLUtilName:Str = 'libcrypto.dylib';
30977:     {$ELSE}
30978:     DLLSSLName:Str = 'libssl.so';
30979:     DLLUtilName:Str = 'libcrypto.so';
30980:     {$ENDIF}
30981:   {$ELSE}
30982:   DLLSSLName:Str = 'ssleay32.dll';
30983:   DLLSSLName2:Str = 'libssl32.dll';
30984:   DLLUtilName:Str = 'libeay32.dll';
30985:   {$ENDIF}
30986: {$ENDIF}
30987:
30988: Proc getHTTP_PNG(vimage: TImage; vURL:Str);
30989: var pngStream: TMemoryStream;
30990: begin
30991:   with TLinearBitmap.Create do
30992:   try
30993:     pngStream:= TMemoryStream.Create;
30994:     try
30995:       HTTPget(vURL, pngStream)
30996:     except
30997:       showmessage(E.message)
30998:     end
30999:     pngStream.Position:= 0;
31000:     LoadFromStream2(pngstream,'JPG');
31001:     vimage.Picture:= NIL;
31002:     AssignTo(vimage.picture);
31003:     //SaveToFile(ExePath+'mX4_open.png');
31004:   finally
31005:     Dispose;
31006:     Free;
31007:     pngStream.Free;
31008:   end;
31009: end;
31010:
31011: Proc ByteArray2Stream(ASource: TBytes; ADest: TStream);
31012: var i: Integer;
31013: begin
31014:    for i:= 0 to Length(ASource)-1 do
31015:       ADest.Write(chr(ASource[i]), 1);
31016: end;
31017:
31018:     buffstr:Str;
```

```
31019:     ADest:= TMemoryStream.create;
31020:     ByteArray2Stream([0,23,3,234,8,45,255], aDest);
31021:     SetLength(buffstr,1);
31022:     ADest.Position:= 0;
31023:     for it:= 0 to adest.size-1 do begin
31024:       adest.read(buffstr, 1);
31025:       writeln('decode: '+itoa(ord(buffstr[1])));
31026:     end;
31027:    aDest.free;
31028:
31029: Proc SearchByteSeq(AHexString:Str; adata: RawByteString);
31030: var dataToSearch: RawByteString;
31031:   nextValueStr:Str;
31032:   nextValueByte: integer; //byte;
31033:   errorPos, index: integer;
31034: begin
31035:   repeat
31036:     AHexString:= Trim(AHexString);
31037:     nextValueStr:= copy(AHexString,1,2);
31038:     delete(AHexString,1,2);
31039:     val('$'+nextValueStr, nextValueByte, errorPos);
31040:     //Func ValBinary( const S :Str; var code : Int) : longint;
31041:     writeln(itoa(errorpos))
31042:     if errorPos = 0 then
31043:       dataToSearch:= datatoSearch + chr(nextValueByte);
31044:   until AHexString = '';
31045:   if length(dataToSearch) > 0 then begin
31046:     index:= pos(dataToSearch, adata)-1;
31047:     if index = 0 then
31048:       writeln('Data not found')
31049:     else
31050:       writeln('Data found at position '+itoa( index));
31051:   end;
31052: end;
31053:
31054: Pictures from:
31055: http://www.softwareschule.ch/images/atomimage1.png
31056: http://www.softwareschule.ch/images/atomimage2.png
31057: http://www.softwareschule.ch/images/maxboxlogo.bmp
31058: http://www.softwareschule.ch/images/maxlindau2.png
31059: http://www.softwareschule.ch/images/citymax.bmp
31060: http://www.softwareschule.ch/images/oposlogo.gif
31061:
31062: //-----------------------------------------------------------------------------
31063: //**************mX4 Macro Tags  ************************************************
31064: //------------------------------------------------------------ ----------------------
31065: asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
       E:\maxbox\docs\maxbox_extract_funclist399.txt end
31066:
31067: //Tag Macros in ini-file configure
31068:
31069:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
31070:
31071: //Tag Macros
31072: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
31073: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
31074: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
31075: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
31076: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
31077: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
31078: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
31079: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
31080: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
31081:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
31082: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
31083: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
31084:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
31085: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
31086:        [perftime numprocessthreads,getIPAddress(getComputerNameWin),
31087:         timetoStr(time), mbversion]),11);
31088: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
31089:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
31090:
31091: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
31092: //#tech!perf: 0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
31093:
31094: //Replace Macros
31095:   SearchAndCopy(memo1.lines, ' timet14:40:27oStr(time), 6);
31096:   SearchAndCopy(memo1.lines, 'atetoS17/02/2017tr(date), 6);
31097:   SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maXbox3\docs\
31098:   SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maXbox3\
31099:   SearchAndCopy(memo1.lines, 'maxbox_functions.txt
31100:   SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maXbox3\Source
31101:   ref: netcologne.dl.sourceforge.net/project/maxbox/maxbox3.zip
31102:   SearchAndCopy(memo1.lines,'#tech'perf:0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
31103: [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
31104: //#tech!perf: 0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
31105:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
31106:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist476.txt
```

```
31107:
31108: //-----------------------------------------------------------------------------
31109: //**************mX4 ToDo List Tags  ../Help/ToDo List**************************
31110: //------------------------------------------------------ --------------------
31111:     while I < sl.Count do begin
31112: //   if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
31113:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
31114:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
31115:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
31116:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
31117:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
31118:         BreakupToDo(Filename, sl, I, 'TODO', False, True) //only priority info TODO
31119:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
31120:         BreakupToDo(Filename, sl, I, 'DONE', False, True) //only priority info DONE
31121:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
31122:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
31123:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
31124:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
31125:       Inc(I);
31126:     end;
31127:
31128: //-----------------------------------------------------------------------------
31129: //**************mX4 Public  Tools API ****************************************
31130: //-----------------------------------------------------------------------------
31131:  file : unit uPSI_fMain.pas;                {$OTAP} Open Tools API Catalog
31132:  // Those functions concern the editor and preprocessor, all of the IDE
31133:  Example: Call it with maxform1.Info1Click(self)
31134:  Note: Call all Methods with maxForm1., e.g.:
31135:                        maxForm1.ShellStyle1Click(self);
31136:
31137: Proc SIRegister_fMain(CL: TPSPascalCompiler);
31138: begin
31139:  Const('BYTECODE','String 'bytecode.txt'
31140:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
31141:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
31142:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
31143:  Const('PSINC','String PS Includes (*.inc)|*.INC
31144:  Const('DEFFILENAME','String 'firstdemo.txt
31145:  Const('DEFINIFILE','String 'maxboxdef.ini
31146:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
31147:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
31148:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
31149:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
31150:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
31151:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml;
31152:  Const('INCLUDEBOX','String 'pas_includebox.inc
31153:  Const('BOOTSCRIPT','String 'maxbootscript.txt
31154:  Const('MBVERSION','String '4.7.6.20
31155:  Const('VERSION','String'4.7.6.20
31156:  Const('MBVER','String '476
31157:  Const('MBVERI','Int'(476);
31158:  Const('MBVERIALL','Int'(47620);
31159:  Const('EXENAME','String 'maXbox4.exe
31160:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
31161:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
31162:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
31163:  Const('MXINTERNETCHECK','String 'www.ask.com
31164:  Const('MXMAIL','String 'max@kleiner.com
31165:  Const('TAB',Char #$09);
31166:  Const('CODECOMPLETION','String 'bds_delphi.dci
31167:  SIRegister_TMaxForm1(CL);
31168: end;
31169:   with FindClass('TForm'),'TMaxForm1') do begin
31170:     memo2', 'TMemo', iptrw);
31171:     memo1', 'TSynMemo', iptrw);
31172:     CB1SCList', 'TComboBox', iptrw);
31173:     mxNavigator', 'TComboBox', iptrw);
31174:     IPHost', 'string', iptrw);
31175:     IPPort', 'Int', iptrw);
31176:     COMPort', 'Int', iptrw);  //3.9.6.4
31177:     Splitter1', 'TSplitter', iptrw);
31178:     PSScript', 'TPSScript', iptrw);
31179:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
31180:     MainMenu1', 'TMainMenu', iptrw);
31181:     Program1', 'TMenuItem', iptrw);
31182:     Compile1', 'TMenuItem', iptrw);
31183:     Files1', 'TMenuItem', iptrw);
31184:     open1', 'TMenuItem', iptrw);
31185:     Save2', 'TMenuItem', iptrw);
31186:     Options1', 'TMenuItem', iptrw);
31187:     Savebefore1', 'TMenuItem', iptrw);
31188:     Largefont1', 'TMenuItem', iptrw);
31189:     sBytecode1', 'TMenuItem', iptrw);
31190:     Saveas3', 'TMenuItem', iptrw);
31191:     Clear1', 'TMenuItem', iptrw);
31192:     Slinenumbers1', 'TMenuItem', iptrw);
31193:     About1', 'TMenuItem', iptrw);
31194:     Search1', 'TMenuItem', iptrw);
31195:     SynPasSyn1', 'TSynPasSyn', iptrw);
```

```
31196:     memo1', 'TSynMemo', iptrw);
31197:     SynEditSearch1', 'TSynEditSearch', iptrw);
31198:     WordWrap1', 'TMenuItem', iptrw);
31199:     XPManifest1', 'TXPManifest', iptrw);
31200:     SearchNext1', 'TMenuItem', iptrw);
31201:     Replace1', 'TMenuItem', iptrw);
31202:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
31203:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
31204:     ShowInclude1', 'TMenuItem', iptrw);
31205:     SynEditPrint1', 'TSynEditPrint', iptrw);
31206:     Printout1', 'TMenuItem', iptrw);
31207:     mnPrintColors1', 'TMenuItem', iptrw);
31208:     dlgFilePrint', 'TPrintDialog', iptrw);
31209:     dlgPrintFont1', 'TFontDialog', iptrw);
31210:     mnuPrintFont1', 'TMenuItem', iptrw);
31211:     Include1', 'TMenuItem', iptrw);
31212:     CodeCompletionList1', 'TMenuItem', iptrw);
31213:     IncludeList1', 'TMenuItem', iptrw);
31214:     ImageList1', 'TImageList', iptrw);
31215:     ImageList2', 'TImageList', iptrw);
31216:     CoolBar1', 'TCoolBar', iptrw);
31217:     ToolBar1', 'TToolBar', iptrw);
31218:     tbtnLoad', 'TToolButton', iptrw);
31219:     ToolButton2', 'TToolButton', iptrw);
31220:     tbtnFind', 'TToolButton', iptrw);
31221:     tbtnCompile', 'TToolButton', iptrw);
31222:     tbtnTrans', 'TToolButton', iptrw);
31223:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
31224:     toolbtnTutorial', 'TToolButton', iptrw);
31225:     tbtn6res', 'TToolButton', iptrw);
31226:     ToolButton5', 'TToolButton', iptrw); 'ToolButton1', 'TToolButton', iptrw);
31227:     ToolButton3', 'TToolButton', iptrw); 'statusBar1', 'TStatusBar', iptrw);
31228:     SaveOutput1', 'TMenuItem', iptrw); 'ExportClipboard1', 'TMenuItem', iptrw);
31229:     Close1', 'TMenuItem', iptrw); 'Manual1', 'TMenuItem', iptrw);
31230:     About2', 'TMenuItem', iptrw); 'loadLastfile1', 'TMenuItem', iptrw);
31231:     imglogo', 'TImage', iptrw); 'cedebug', 'TPSScriptDebugger', iptrw);
31232:     debugPopupMenu1', 'TPopupMenu', iptrw);
31233:     BreakPointMenu', 'TMenuItem', iptrw);
31234:     Decompile1', 'TMenuItem', iptrw);
31235:     StepInto1', 'TMenuItem', iptrw);
31236:     StepOut1', 'TMenuItem', iptrw);
31237:     Reset1', 'TMenuItem', iptrw);
31238:     DebugRun1', 'TMenuItem', iptrw);
31239:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
31240:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
31241:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
31242:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
31243:     tutorial4', 'TMenuItem', iptrw);
31244:     ExporttoClipboard1', 'TMenuItem', iptrw);
31245:     ImportfromClipboard1', 'TMenuItem', iptrw);
31246:     N4','TMenuItem', iptrw); N5', 'TMenuItem', iptrw);N6', 'TMenuItem', iptrw);
31247:     ImportfromClipboard2', 'TMenuItem', iptrw);
31248:     tutorial1', 'TMenuItem', iptrw);
31249:     N7', 'TMenuItem', iptrw);
31250:     ShowSpecChars1', 'TMenuItem', iptrw);
31251:     OpenDirectory1', 'TMenuItem', iptrw);
31252:     procMess', 'TMenuItem', iptrw);
31253:     tbtnUseCase', 'TToolButton', iptrw);
31254:     ToolButton7', 'TToolButton', iptrw);
31255:     EditFont1', 'TMenuItem', iptrw);
31256:     UseCase1', 'TMenuItem', iptrw);
31257:     tutorial21', 'TMenuItem', iptrw);
31258:     OpenUseCase1', 'TMenuItem', iptrw);
31259:     PSImport_DB1', 'TPSImport_DB', iptrw);
31260:     tutorial31', 'TMenuItem', iptrw);
31261:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
31262:     HTMLSyntax1', 'TMenuItem', iptrw);
31263:     ShowInterfaces1', 'TMenuItem', iptrw);
31264:     Tutorial5', 'TMenuItem', iptrw);
31265:     AllFunctionsList1', 'TMenuItem', iptrw);
31266:     ShowLastException1', 'TMenuItem', iptrw);
31267:     PlayMP31', 'TMenuItem', iptrw);
31268:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
31269:     texSyntax1', 'TMenuItem', iptrw);
31270:     N8', 'TMenuItem', iptrw);
31271:     GetEMails1', 'TMenuItem', iptrw);
31272:     SynCppSyn1', 'TSynCppSyn', iptrw);
31273:     CSyntax1', 'TMenuItem', iptrw);
31274:     Tutorial6', 'TMenuItem', iptrw);
31275:     New1', 'TMenuItem', iptrw);
31276:     AllObjectsList1', 'TMenuItem', iptrw);
31277:     LoadBytecode1', 'TMenuItem', iptrw);
31278:     CipherFile1', 'TMenuItem', iptrw);
31279:     N9', 'TMenuItem', iptrw); 'N10', 'TMenuItem', iptrw);
31280:     Tutorial11', 'TMenuItem', iptrw);
31281:     Tutorial71', 'TMenuItem', iptrw);
31282:     UpdateService1', 'TMenuItem', iptrw);
31283:     PascalSchool1', 'TMenuItem', iptrw);
31284:     Tutorial81', 'TMenuItem', iptrw);
```

```
31285:     DelphiSite1', 'TMenuItem', iptrw);
31286:     Output1', 'TMenuItem', iptrw);
31287:     TerminalStyle1', 'TMenuItem', iptrw);
31288:     ReadOnly1', 'TMenuItem', iptrw);
31289:     ShellStyle1', 'TMenuItem', iptrw);
31290:     BigScreen1', 'TMenuItem', iptrw);
31291:     Tutorial91', 'TMenuItem', iptrw);
31292:     SaveOutput2', 'TMenuItem', iptrw);
31293:     N11', 'TMenuItem', iptrw);
31294:     SaveScreenshot', 'TMenuItem', iptrw);
31295:     Tutorial101', 'TMenuItem', iptrw);
31296:     SQLSyntax1', 'TMenuItem', iptrw);
31297:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
31298:     Console1', 'TMenuItem', iptrw);
31299:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
31300:     XMLSyntax1', 'TMenuItem', iptrw);
31301:     ComponentCount1', 'TMenuItem', iptrw);
31302:     NewInstance1', 'TMenuItem', iptrw);
31303:     toolbtnTutorial', 'TToolButton', iptrw);
31304:     Memory1', 'TMenuItem', iptrw);
31305:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
31306:     JavaSyntax1', 'TMenuItem', iptrw);
31307:     SyntaxCheck1', 'TMenuItem', iptrw);
31308:     Tutorial10Statistics1', 'TMenuItem', iptrw);
31309:     ScriptExplorer1', 'TMenuItem', iptrw);
31310:     FormOutput1', 'TMenuItem', iptrw);
31311:     ArduinoDump1', 'TMenuItem', iptrw);
31312:     AndroidDump1', 'TMenuItem', iptrw);
31313:     GotoEnd1', 'TMenuItem', iptrw);
31314:     AllResourceList1', 'TMenuItem', iptrw);
31315:     ToolButton4', 'TToolButton', iptrw);
31316:     tbtn6res', 'TToolButton', iptrw);
31317:     Tutorial11Forms1', 'TMenuItem', iptrw);
31318:     Tutorial12SQL1', 'TMenuItem', iptrw);
31319:     ResourceExplore1', 'TMenuItem', iptrw);
31320:     Info1', 'TMenuItem', iptrw);
31321:     N12', 'TMenuItem', iptrw);
31322:     CryptoBox1', 'TMenuItem', iptrw);
31323:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
31324:     CipherFile2', 'TMenuItem', iptrw);
31325:     N13', 'TMenuItem', iptrw);
31326:     ModulesCount1', 'TMenuItem', iptrw);
31327:     AddOns2', 'TMenuItem', iptrw);
31328:     N4GewinntGame1', 'TMenuItem', iptrw);
31329:     DocuforAddOns1', 'TMenuItem', iptrw);
31330:     Tutorial14Async1', 'TMenuItem', iptrw);
31331:     Lessons15Review1', 'TMenuItem', iptrw);
31332:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
31333:     PHPSyntax1', 'TMenuItem', iptrw);
31334:     Breakpoint1', 'TMenuItem', iptrw);
31335:     SerialRS2321', 'TMenuItem', iptrw);
31336:     N14', 'TMenuItem', iptrw);
31337:     SynCSSyn1', 'TSynCSSyn', iptrw);
31338:     CSyntax2', 'TMenuItem', iptrw);
31339:     Calculator1', 'TMenuItem', iptrw);
31340:     tbtnSerial', 'TToolButton', iptrw);
31341:     ToolButton8', 'TToolButton', iptrw);
31342:     Tutorial151', 'TMenuItem', iptrw);
31343:     N15', 'TMenuItem', iptrw);
31344:     N16', 'TMenuItem', iptrw);
31345:     ControlBar1', 'TControlBar', iptrw);
31346:     ToolBar2', 'TToolBar', iptrw);
31347:     BtnOpen', 'TToolButton', iptrw);
31348:     BtnSave', 'TToolButton', iptrw);
31349:     BtnPrint', 'TToolButton', iptrw);
31350:     BtnColors', 'TToolButton', iptrw);
31351:     btnClassReport', 'TToolButton', iptrw);
31352:     BtnRotateRight', 'TToolButton', iptrw);
31353:     BtnFullSize', 'TToolButton', iptrw);
31354:     BtnFitToWindowSize', 'TToolButton', iptrw);
31355:     BtnZoomMinus', 'TToolButton', iptrw);
31356:     BtnZoomPlus', 'TToolButton', iptrw);
31357:     Panel1', 'TPanel', iptrw);
31358:     LabelBrettgroesse', 'TLabel', iptrw);
31359:     CB1SCList', 'TComboBox', iptrw);
31360:     ImageListNormal', 'TImageList', iptrw);
31361:     spbtnexplore', 'TSpeedButton', iptrw);
31362:     spbtnexample', 'TSpeedButton', iptrw);
31363:     spbsaveas', 'TSpeedButton', iptrw);
31364:     imglogobox', 'TImage', iptrw);
31365:     EnlargeFont1', 'TMenuItem', iptrw);
31366:     EnlargeFont2', 'TMenuItem', iptrw);
31367:     ShrinkFont1', 'TMenuItem', iptrw);
31368:     ThreadDemo1', 'TMenuItem', iptrw);
31369:     HEXEditor1', 'TMenuItem', iptrw);
31370:     HEXView1', 'TMenuItem', iptrw);
31371:     HEXInspect1', 'TMenuItem', iptrw);
31372:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
31373:     ExporttoHTML1', 'TMenuItem', iptrw);
```

```
31374:     ClassCount1', 'TMenuItem', iptrw);
31375:     HTMLOutput1', 'TMenuItem', iptrw);
31376:     HEXEditor2', 'TMenuItem', iptrw);
31377:     Minesweeper1', 'TMenuItem', iptrw);
31378:     N17', 'TMenuItem', iptrw);
31379:     PicturePuzzle1', 'TMenuItem', iptrw);
31380:     sbvclhelp', 'TSpeedButton', iptrw);
31381:     DependencyWalker1', 'TMenuItem', iptrw);
31382:     WebScanner1', 'TMenuItem', iptrw);
31383:     View1', 'TMenuItem', iptrw);
31384:     mnToolbar1', 'TMenuItem', iptrw);
31385:     mnStatusbar2', 'TMenuItem', iptrw);
31386:     mnConsole2', 'TMenuItem', iptrw);
31387:     mnCoolbar2', 'TMenuItem', iptrw);
31388:     mnSplitter2', 'TMenuItem', iptrw);
31389:     WebServer1', 'TMenuItem', iptrw);
31390:     Tutorial17Server1', 'TMenuItem', iptrw);
31391:     Tutorial18Arduino1', 'TMenuItem', iptrw);
31392:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
31393:     PerlSyntax1', 'TMenuItem', iptrw);
31394:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
31395:     PythonSyntax1', 'TMenuItem', iptrw);
31396:     DMathLibrary1', 'TMenuItem', iptrw);
31397:     IntfNavigator1', 'TMenuItem', iptrw);
31398:     EnlargeFontConsole1', 'TMenuItem', iptrw);
31399:     ShrinkFontConsole1', 'TMenuItem', iptrw);
31400:     SetInterfaceList1', 'TMenuItem', iptrw);
31401:     popintfList', 'TPopupMenu', iptrw);
31402:     intfAdd1', 'TMenuItem', iptrw);
31403:     intfDelete1', 'TMenuItem', iptrw);
31404:     intfRefactor1', 'TMenuItem', iptrw);
31405:     Defactor1', 'TMenuItem', iptrw);
31406:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
31407:     Tutorial20Regex', 'TMenuItem', iptrw);
31408:     N18', 'TMenuItem', iptrw);
31409:     ManualE1', 'TMenuItem', iptrw);
31410:     FullTextFinder1', 'TMenuItem', iptrw);
31411:     Move1', 'TMenuItem', iptrw);
31412:     FractalDemo1', 'TMenuItem', iptrw);
31413:     Tutorial21Android1', 'TMenuItem', iptrw);
31414:     Tutorial0Function1', 'TMenuItem', iptrw);
31415:     SimuLogBox1', 'TMenuItem', iptrw);
31416:     OpenExamples1', 'TMenuItem', iptrw);
31417:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
31418:     JavaScriptSyntax1', 'TMenuItem', iptrw);
31419:     Halt1', 'TMenuItem', iptrw);
31420:     CodeSearch1', 'TMenuItem', iptrw);
31421:     SynRubySyn1', 'TSynRubySyn', iptrw);
31422:     RubySyntax1', 'TMenuItem', iptrw);
31423:     Undo1', 'TMenuItem', iptrw);
31424:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
31425:     LinuxShellScript1', 'TMenuItem', iptrw);
31426:     Rename1', 'TMenuItem', iptrw);
31427:     spdcodesearch', 'TSpeedButton', iptrw);
31428:     Preview1', 'TMenuItem', iptrw);
31429:     Tutorial22Services1', 'TMenuItem', iptrw);
31430:     Tutorial23RealTime1', 'TMenuItem', iptrw);
31431:     Configuration1', 'TMenuItem', iptrw);
31432:     MP3Player1', 'TMenuItem', iptrw);
31433:     DLLSpy1', 'TMenuItem', iptrw);
31434:     SynURIOpener1', 'TSynURIOpener', iptrw);
31435:     SynURISyn1', 'TSynURISyn', iptrw);
31436:     URILinksClicks1', 'TMenuItem', iptrw);
31437:     EditReplace1', 'TMenuItem', iptrw);
31438:     GotoLine1', 'TMenuItem', iptrw);
31439:     ActiveLineColor1', 'TMenuItem', iptrw);
31440:     ConfigFile1', 'TMenuItem', iptrw);
31441:     Sort1Intflist', 'TMenuItem', iptrw);
31442:     Redo1', 'TMenuItem', iptrw);
31443:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
31444:     Tutorial25Configuration1', 'TMenuItem', iptrw);
31445:     IndentSelection1', 'TMenuItem', iptrw);
31446:     UnindentSection1', 'TMenuItem', iptrw);
31447:     SkyStyle1', 'TMenuItem', iptrw);
31448:     N19', 'TMenuItem', iptrw);
31449:     CountWords1', 'TMenuItem', iptrw);
31450:     imbookmarkimages', 'TImageList', iptrw);
31451:     Bookmark11', 'TMenuItem', iptrw);
31452:     N20', 'TMenuItem', iptrw);
31453:     Bookmark21', 'TMenuItem', iptrw);
31454:     Bookmark31', 'TMenuItem', iptrw);
31455:     Bookmark41', 'TMenuItem', iptrw);
31456:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
31457:
31458: Proc IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
31459: Proc IFPS3ClassesPlugin1ExecImport(Sender:TObject;Exec:TPSExec;x:TPSRuntimeClassImporter);
31460:  Proc PSScriptCompile( Sender : TPSScript)
31461:  Proc Compile1Click( Sender : TObject)
31462:  Proc PSScriptExecute( Sender : TPSScript)
```

```
31463:  Proc open1Click( Sender : TObject)
31464:  Proc Save2Click( Sender : TObject)
31465:  Proc Savebefore1Click( Sender : TObject)
31466:  Proc Largefont1Click( Sender : TObject)
31467:  Proc FormActivate( Sender : TObject)
31468:  Proc SBytecode1Click( Sender : TObject)
31469:  Proc FormKeyPress( Sender : TObject; var Key : Char)
31470:  Proc Saveas3Click( Sender : TObject)
31471:  Proc Clear1Click( Sender : TObject)
31472:  Proc Slinenumbers1Click( Sender : TObject)
31473:  Proc About1Click( Sender : TObject)
31474:  Proc Search1Click( Sender : TObject)
31475:  Proc FormCreate( Sender : TObject)
31476:  Proc Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:str;Line,Column:Int;
31477:                                                  var Action : TSynReplaceAction)
31478:  Proc Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
31479:  Proc WordWrap1Click( Sender : TObject)
31480:  Proc SearchNext1Click( Sender : TObject)
31481:  Proc Replace1Click( Sender : TObject)
31482:  Func PSScriptNeedFile(Sdr:TObject;const OrginFileName:Str;var FName,Output:Str):Bool;
31483:  Proc ShowInclude1Click( Sender : TObject)
31484:  Proc Printout1Click( Sender : TObject)
31485:  Proc mnuPrintFont1Click( Sender : TObject)
31486:  Proc Include1Click( Sender : TObject)
31487:  Proc FormDestroy( Sender : TObject)
31488:  Proc FormClose( Sender : TObject; var Action : TCloseAction)
31489:  Proc UpdateView1Click( Sender : TObject)
31490:  Proc CodeCompletionList1Click( Sender : TObject)
31491:  Proc SaveOutput1Click( Sender : TObject)
31492:  Proc ExportClipboard1Click( Sender : TObject)
31493:  Proc Close1Click( Sender : TObject)
31494:  Proc Manual1Click( Sender : TObject)
31495:  Proc LoadLastFile1Click( Sender : TObject)
31496:  Proc Memo1Change( Sender : TObject)
31497:  Proc Decompile1Click( Sender : TObject)
31498:  Proc StepInto1Click( Sender : TObject)
31499:  Proc StepOut1Click( Sender : TObject)
31500:  Proc Reset1Click( Sender : TObject)
31501:  Proc cedebugAfterExecute( Sender : TPSScript)
31502:  Proc cedebugBreakpoint(Sender:TObject;const FileName:str;Position,Row,Col:Card)
31503:  Proc cedebugCompile( Sender : TPSScript)
31504:  Proc cedebugExecute( Sender : TPSScript)
31505:  Proc cedebugIdle( Sender : TObject)
31506:  Proc cedebugLineInfo(Sender:TObject;const FileName:str;Position,Row,Col:Card)
31507:  Proc Memo1SpecialLineColors(Sender:TObject;Line:Int;var Speci:Bool;var FG,BG:TColor);
31508:  Proc BreakPointMenuClick( Sender : TObject)
31509:  Proc DebugRun1Click( Sender : TObject)
31510:  Proc tutorial4Click( Sender : TObject)
31511:  Proc ImportfromClipboard1Click( Sender : TObject)
31512:  Proc ImportfromClipboard2Click( Sender : TObject)
31513:  Proc tutorial1Click( Sender : TObject)
31514:  Proc ShowSpecChars1Click( Sender : TObject)
31515:  Proc StatusBar1Dbl1Click( Sender : TObject)
31516:  Proc PSScriptLine( Sender : TObject)
31517:  Proc OpenDirectory1Click( Sender : TObject)
31518:  Proc procMessClick( Sender : TObject)
31519:  Proc tbtnUseCaseClick( Sender : TObject)
31520:  Proc EditFont1Click( Sender : TObject)
31521:  Proc tutorial21Click( Sender : TObject)
31522:  Proc tutorial31Click( Sender : TObject)
31523:  Proc HTMLSyntax1Click( Sender : TObject)
31524:  Proc ShowInterfaces1Click( Sender : TObject)
31525:  Proc Tutorial5Click( Sender : TObject)
31526:  Proc ShowLastException1Click( Sender : TObject)
31527:  Proc PlayMP31Click( Sender : TObject)
31528:  Proc AllFunctionsList1Click( Sender : TObject)
31529:  Proc texSyntax1Click( Sender : TObject)
31530:  Proc GetEMails1Click( Sender : TObject)
31531:  Proc DelphiSite1Click(Sender: TObject);
31532:  Proc TerminalStyle1Click(Sender: TObject);
31533:  Proc ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
31534:  Proc ShellStyle1Click(Sender: TObject);
31535:  Proc Console1Click(Sender: TObject);     //3.2
31536:  Proc BigScreen1Click(Sender: TObject);
31537:  Proc Tutorial91Click(Sender: TObject);
31538:  Proc SaveScreenshotClick(Sender: TObject);
31539:  Proc Tutorial101Click(Sender: TObject);
31540:  Proc SQLSyntax1Click(Sender: TObject);
31541:  Proc XMLSyntax1Click(Sender: TObject);
31542:  Proc ComponentCount1Click(Sender: TObject);
31543:  Proc NewInstance1Click(Sender: TObject);
31544:  Proc CSyntax1Click(Sender: TObject);
31545:  Proc Tutorial6Click(Sender: TObject);
31546:  Proc New1Click(Sender: TObject);
31547:  Proc AllObjectsList1Click(Sender: TObject);
31548:  Proc LoadBytecode1Click(Sender: TObject);
31549:  Proc CipherFile1Click(Sender: TObject);  //V3.5
31550:  Proc NewInstance1Click(Sender: TObject);
31551:  Proc toolbtnTutorialClick(Sender: TObject);
```

```
31552:  Proc Memory1Click(Sender: TObject);
31553:  Proc JavaSyntax1Click(Sender: TObject);
31554:  Proc SyntaxCheck1Click(Sender: TObject);
31555:  Proc ScriptExplorer1Click(Sender: TObject);
31556:  Proc FormOutput1Click(Sender: TObject);  //V3.6
31557:  Proc GotoEnd1Click(Sender: TObject);
31558:  Proc AllResourceList1Click(Sender: TObject);
31559:  Proc tbtn6resClick(Sender: TObject);   //V3.7
31560:  Proc Info1Click(Sender: TObject);
31561:  Proc Tutorial10Statistics1Click(Sender: TObject);
31562:  Proc Tutorial11Forms1Click(Sender: TObject);
31563:  Proc Tutorial12SQL1Click(Sender: TObject);  //V3.8
31564:  Proc ResourceExplore1Click(Sender: TObject);
31565:  Proc Info1Click(Sender: TObject);
31566:  Proc CryptoBox1Click(Sender: TObject);
31567:  Proc ModulesCount1Click(Sender: TObject);
31568:  Proc N4GewinntGame1Click(Sender: TObject);
31569:  Proc PHPSyntax1Click(Sender: TObject);
31570:  Proc SerialRS2321Click(Sender: TObject);
31571:  Proc CSyntax2Click(Sender: TObject);
31572:  Proc Calculator1Click(Sender: TObject);
31573:  Proc Tutorial13Ciphering1Click(Sender: TObject);
31574:  Proc Tutorial14Async1Click(Sender: TObject);
31575:  Proc PHPSyntax1Click(Sender: TObject);
31576:  Proc BtnZoomPlusClick(Sender: TObject);
31577:  Proc BtnZoomMinusClick(Sender: TObject);
31578:  Proc btnClassReportClick(Sender: TObject);
31579:  Proc ThreadDemo1Click(Sender: TObject);
31580:  Proc HEXView1Click(Sender: TObject);
31581:  Proc ExporttoHTML1Click(Sender: TObject);
31582:  Proc Minesweeper1Click(Sender: TObject);
31583:  Proc PicturePuzzle1Click(Sender: TObject); //V3.9
31584:  Proc sbvclhelpClick(Sender: TObject);
31585:  Proc DependencyWalker1Click(Sender: TObject);
31586:  Proc CB1SCListDrawItem(Control:TWinCtrl;Index:Int;aRect:TRect;State:TOwnerDrawState);
31587:  Proc WebScanner1Click(Sender: TObject);
31588:  Proc mnToolbar1Click(Sender: TObject);
31589:  Proc mnStatusbar2Click(Sender: TObject);
31590:  Proc mnConsole2Click(Sender: TObject);
31591:  Proc mnCoolbar2Click(Sender: TObject);
31592:  Proc mnSplitter2Click(Sender: TObject);
31593:  Proc WebServer1Click(Sender: TObject);
31594:  Proc PerlSyntax1Click(Sender: TObject);
31595:  Proc PythonSyntax1Click(Sender: TObject);
31596:  Proc DMathLibrary1Click(Sender: TObject);
31597:  Proc IntfNavigator1Click(Sender: TObject);
31598:  Proc FullTextFinder1Click(Sender: TObject);
31599:  Func AppName:Str;
31600:  Func ScriptName:Str;
31601:  Func LastName:Str;
31602:  Proc FractalDemo1Click(Sender: TObject);
31603:  Proc SimuLogBox1Click(Sender: TObject);
31604:  Proc OpenExamples1Click(Sender: TObject);
31605:  Proc Halt1Click(Sender: TObject);
31606:  Proc Stop;
31607:  Proc CodeSearch1Click(Sender: TObject);
31608:  Proc RubySyntax1Click(Sender: TObject);
31609:  Proc Undo1Click(Sender: TObject);
31610:  Proc LinuxShellScript1Click(Sender: TObject);
31611:  Proc WebScannerDirect(urls:Str);
31612:  Proc WebScanner(urls:Str);
31613:  Proc LoadInterfaceList2;
31614:  Proc DLLSpy1Click(Sender: TObject);
31615:  Proc Memo1DblClick(Sender: TObject);
31616:  Proc URILinksClicks1Click(Sender: TObject);
31617:  Proc GotoLine1Click(Sender: TObject);
31618:  Proc ConfigFile1Click(Sender: TObject);
31619:  Proc Sort1Intflist1Click( Sender : TObject)
31620:  Proc Redo1Click( Sender : TObject)
31621:  Proc Tutorial24CleanCode1Click( Sender : TObject)
31622:  Proc IndentSelection1Click( Sender : TObject)
31623:  Proc UnindentSection1Click( Sender : TObject)
31624:  Proc SkyStyle1Click( Sender : TObject)
31625:  Proc CountWords1Click( Sender : TObject)
31626:  Proc Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
31627:  Proc Memo1GutterClick(Send:TObject;Button:TMouseButton;X,Y,Line:Int;Mark:TSynEditMark);
31628:  Proc Bookmark11Click( Sender : TObject)
31629:  Proc Bookmark21Click( Sender : TObject)
31630:  Proc Bookmark31Click( Sender : TObject)
31631:  Proc Bookmark41Click( Sender : TObject)
31632: Proc SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
31633:     'STATMemoryReport', 'boolean', iptrw);
31634:     'IPPort', 'Int', iptrw);
31635:     'COMPort', 'Int', iptrw);
31636:     'lbintflist', 'TListBox', iptrw);
31637:  Func GetStatChange :Bool
31638:  Proc SetStatChange( vstat :Bool)
31639:  Func GetActFileName :Str
31640:  Proc SetActFileName( vname :Str)
```

```
31641:  Func GetLastFileName :Str
31642:  Proc SetLastFileName( vname :Str)
31643:  Proc WebScannerDirect( urls :Str)
31644:  Proc LoadInterfaceList2
31645:  Func GetStatExecuteShell :Bool
31646:  Proc DoEditorExecuteCommand( EditorCommand : word)
31647:  Func GetActiveLineColor: TColor
31648:  Proc SetActiveLineColor(acolor: TColor)
31649:  Proc ScriptListbox1Click(Sender: TObject);
31650:  Proc Memo2KeyPress(Sender: TObject; var Key: Char);
31651:  Proc EnlargeGutter1Click(Sender: TObject);
31652:  Proc Tetris1Click(Sender: TObject);
31653:  Proc ToDoList1Click(Sender: TObject);
31654:  Proc ProcessList1Click(Sender: TObject);
31655:  Proc MetricReport1Click(Sender: TObject);
31656:  Proc ProcessList1Click(Sender: TObject);
31657:  Proc TCPSockets1Click(Sender: TObject);
31658:  Proc ConfigUpdate1Click(Sender: TObject);
31659:  Proc ADOWorkbench1Click(Sender: TObject);
31660:  Proc SocketServer1Click(Sender: TObject);
31661:  Proc FormDemo1Click(Sender: TObject);
31662:  Proc Richedit1Click(Sender: TObject);
31663:  Proc SimpleBrowser1Click(Sender: TObject);
31664:  Proc DOSShell1Click(Sender: TObject);
31665:  Proc SynExport1Click(Sender: TObject);
31666:  Proc ExporttoRTF1Click(Sender: TObject);
31667:  Proc FormCloseQuery(Sender: TObject; var CanClose:Bool);
31668:  Proc SOAPTester1Click(Sender: TObject);
31669:  Proc Sniffer1Click(Sender: TObject);
31670:  Proc AutoDetectSyntax1Click(Sender: TObject);
31671:  Proc FPlot1Click(Sender: TObject);
31672:  Proc PasStyle1Click(Sender: TObject);
31673:  Proc Tutorial183RGBLED1Click(Sender: TObject);
31674:  Proc Reversi1Click(Sender: TObject);
31675:  Proc ManualmaXbox1Click(Sender: TObject);
31676:  Proc BlaisePascalMagazine1Click(Sender: TObject);
31677:  Proc AddToDo1Click(Sender: TObject);
31678:  Proc CreateGUID1Click(Sender: TObject);
31679:  Proc Tutorial27XML1Click(Sender: TObject);
31680:  Proc CreateDLLStub1Click(Sender: TObject);
31681:  Proc Tutorial28DLL1Click(Sender: TObject);;
31682:  Proc ResetKeyPressed;;
31683:  Proc SetKeyPressed;;
31684:  Proc KeyPressedFalse;
31685:  Proc FileChanges1Click(Sender: TObject);;
31686:  Proc OpenGLTry1Click(Sender: TObject);;
31687:  Proc AllUnitList1Click(Sender: TObject);;
31688:  Proc Tutorial29UMLClick(Sender: TObject);
31689:  Proc CreateHeader1Click(Sender: TObject);
31690:  Proc Oscilloscope1Click(Sender: TObject);;
31691:  Proc Tutorial30WOT1Click(Sender: TObject);;
31692:  Proc GetWebScript1Click(Sender: TObject);;
31693:  Proc Checkers1Click(Sender: TObject);;
31694:  Proc TaskMgr1Click(Sender: TObject);;
31695:  Proc WebCam1Click(Sender: TObject);;
31696:  Proc Tutorial31Closure1Click(Sender: TObject);;
31697:  Proc GEOMapView1Click(Sender: TObject);;
31698:  Proc Run1Click(Sender: TObject);
31699:  MaxForm1.GPSSatView1Click, 'GPSSatView1Click;
31700:  MaxForm1.N3DLab1Click, 'N3DLab1Click;
31701:  Proc ExternalApp1Click(Sender: TObject);;
31702:  Proc PANView1Click(Sender: TObject);
31703:  Proc Tutorial39GEOMaps1Click(Sender: TObject);
31704:  Proc UnitConverter1Click(Sender: TObject);
31705:  maxform1.myscript1click(self)
31706:  Proc Terminal1Click(Sender: TObject);
31707:  Proc Tutorial361Click(Sender: TObject);
31708:  Proc TrainingArduino1Click(Sender: TObject);
31709:  Proc Chess41Click(Sender: TObject);
31710:  Proc OrangeStyle1Click(Sender: TObject);
31711:  Proc Darkcolor1Click(Sender: TObject);
31712: //-----------------------------------------------------------------------------
31713: //**************mX4 Editor SynEdit Tools API ************************************
31714: //-----------------------------------------------------------------------------
31715: Proc SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
31716: begin  //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
31717:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
31718:     Constructor Create( AOwner : TComponent)
31719:     SelStart', 'Int', iptrw);
31720:     SelEnd', 'Int', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
31721:     Proc UpdateCaret
31722: Proc AddKey(Command:TSynEditorCommand;Key1:word;SS1:TShiftState;Key2:word;SS2:TShiftState);
31723: Proc AddKey(Command:TSynEditorCommand;Key1:word;SS1:TShiftState;Key2:word;SS2:TShiftState);
31724:   Proc BeginUndoBlock
31725:   Proc BeginUpdate
31726:   Func CaretInView :Bool
31727:   Func CharIndexToRowCol( Index : Int) : TBufferCoord
31728:   Proc Clear
31729:   Proc ClearAll
```

```
31730:   Proc ClearBookMark( BookMark : Int)
31731:   Proc ClearSelection
31732:   Proc CommandProcessor(Command:TSynEditorCommand; AChar : char; Data : pointer)
31733:   Proc ClearUndo
31734:   Proc CopyToClipboard
31735:   Proc CutToClipboard
31736:   Proc DoCopyToClipboard( const SText :Str)
31737:   Proc EndUndoBlock
31738:   Proc EndUpdate
31739:   Proc EnsureCursorPosVisible
31740:   Proc EnsureCursorPosVisibleEx( ForceToMiddle :Bool)
31741:   Proc FindMatchingBracket
31742:   Func GetMatchingBracket : TBufferCoord
31743:   Func GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
31744:   Proc ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
31745:   Func GetBookMark( BookMark : Int; var X, Y : Int) :Bool
31746:   Func GetHighlighterAttriAtRowCol(const XY: TBufferCoord; var Token:str; var Attri
31747:            : TSynHighlighterAttributes) :Bool
31748:   Func GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token :Str;
31749:            var TokenType, Start : Int; var Attri:TSynHighlighterAttributes):boolean
31750:   Func GetPositionOfMouse( out aPos : TBufferCoord) :Bool
31751:   Func GetWordAtRowCol( const XY : TBufferCoord) :Str
31752:   Proc GotoBookMark( BookMark : Int)
31753:   Proc GotoLineAndCenter( ALine : Int)
31754:   Func IdentChars : TSynIdentChars
31755:   Proc InvalidateGutter
31756:   Proc InvalidateGutterLine( aLine : Int)
31757:   Proc InvalidateGutterLines( FirstLine, LastLine : Int)
31758:   Proc InvalidateLine( Line : Int)
31759:   Proc InvalidateLines( FirstLine, LastLine : Int)
31760:   Proc InvalidateSelection
31761:   Func IsBookmark( BookMark : Int) :Bool
31762:   Func IsPointInSelection( const Value : TBufferCoord) :Bool
31763:   Proc LockUndo
31764:   Func BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
31765:   Func DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
31766:   Func LineToRow( aLine : Int) : Int
31767:   Func RowToLine( aRow : Int) : Int
31768:   Func NextWordPos : TBufferCoord
31769:   Func NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
31770:   Proc PasteFromClipboard
31771:   Func WordStart : TBufferCoord
31772:   Func WordStartEx( const XY : TBufferCoord) : TBufferCoord
31773:   Func WordEnd : TBufferCoord
31774:   Func WordEndEx( const XY : TBufferCoord) : TBufferCoord
31775:   Func PrevWordPos : TBufferCoord
31776:   Func PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
31777:   Func PixelsToRowColumn( aX, aY : Int) : TDisplayCoord
31778:   Func PixelsToNearestRowColumn( aX, aY : Int) : TDisplayCoord
31779:   Proc Redo
31780:   Proc RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerDat:pointr);
31781:   Func RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
31782:   Func RowColToCharIndex( RowCol : TBufferCoord) : Int
31783:   Func SearchReplace( const ASearch,AReplace:str; AOptions:TSynSearchOptions): int
31784:   Proc SelectAll
31785:   Proc SetBookMark( BookMark : Int; X : Int; Y : Int)
31786:   Proc SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
31787:   Proc SetDefaultKeystrokes
31788:   Proc SetSelWord
31789:   Proc SetWordBlock( Value : TBufferCoord)
31790:   Proc Undo Proc UnlockUndo
31791:   Proc UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
31792:   Proc AddKeyUpHandler( aHandler : TKeyEvent)
31793:   Proc RemoveKeyUpHandler( aHandler : TKeyEvent)
31794:   Proc AddKeyDownHandler( aHandler : TKeyEvent)
31795:   Proc RemoveKeyDownHandler( aHandler : TKeyEvent)
31796:   Proc AddKeyPressHandler( aHandler : TKeyPressEvent)
31797:   Proc RemoveKeyPressHandler( aHandler : TKeyPressEvent)
31798:   Proc AddFocusControl( aControl : TWinControl)
31799:   Proc RemoveFocusControl( aControl : TWinControl)
31800:   Proc AddMouseDownHandler( aHandler : TMouseEvent)
31801:   Proc RemoveMouseDownHandler( aHandler : TMouseEvent)
31802:   Proc AddMouseUpHandler( aHandler : TMouseEvent)
31803:   Proc RemoveMouseUpHandler( aHandler : TMouseEvent)
31804:   Proc AddMouseCursorHandler( aHandler : TMouseCursorEvent)
31805:   Proc RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
31806:   Proc SetLinesPointer( ASynEdit : TCustomSynEdit)
31807:   Proc RemoveLinesPointer
31808:   Proc HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
31809:   Proc UnHookTextBuffer
31810:   BlockBegin', 'TBufferCoord', iptrw);
31811:   BlockEnd', 'TBufferCoord', iptrw);
31812:   CanPaste', 'Boolean', iptr); 'CanRedo', 'boolean', iptr);
31813:   CanUndo', 'boolean', iptr);  'CaretX', 'Int', iptrw);
31814:   CaretY', 'Int', iptrw); 'CaretXY', 'TBufferCoord', iptrw);
31815:   ActiveLineColor', 'TColor', iptrw);
31816:   DisplayX', 'Int', iptr); 'DisplayY', 'Int', iptr);
31817:   DisplayXY', 'TDisplayCoord', iptr); 'DisplayLineCount', 'Int', iptr);
31818:   CharsInWindow', 'Int', iptr);
```

```
31819:   CharWidth', 'Int', iptr);
31820:   Font', 'TFont', iptrw);
31821:   GutterWidth', 'Int', iptr);
31822:   Highlighter', 'TSynCustomHighlighter', iptrw);
31823:   LeftChar', 'Int', iptrw);
31824:   LineHeight', 'Int', iptr);
31825:   LinesInWindow', 'Int', iptr);
31826:   LineText', 'string', iptrw); Lines', 'TStrings', iptrw);
31827:   Marks', 'TSynEditMarkList', iptr);
31828:   MaxScrollWidth', 'Int', iptrw);
31829:   Modified', 'Boolean', iptrw);
31830:   PaintLock', 'Int', iptr);
31831:   ReadOnly', 'Boolean', iptrw);
31832:   SearchEngine', 'TSynEditSearchCustom', iptrw);
31833:   SelAvail', 'Boolean', iptr); SelLength', 'Int', iptrw);
31834:   SelTabBlock', 'Boolean', iptr);
31835:   SelTabLine', 'Boolean', iptr); SelText', 'string', iptrw);
31836:   StateFlags', 'TSynStateFlags', iptr);
31837:   Text', 'string', iptrw); TopLine', 'Int', iptrw);
31838:   WordAtCursor', 'string', iptr); 'WordAtMouse', 'string', iptr);
31839:   UndoList', 'TSynEditUndoList', iptr);
31840:   RedoList', 'TSynEditUndoList', iptr);
31841:   OnProcessCommand', 'TProcessCommandEvent', iptrw);
31842:   BookMarkOptions', 'TSynBookMarkOpt', iptrw);
31843:   BorderStyle', 'TSynBorderStyle', iptrw);
31844:   ExtraLineSpacing', 'Int', iptrw);
31845:   Gutter', 'TSynGutter', iptrw);
31846:   HideSelection', 'boolean', iptrw);
31847:   InsertCaret', 'TSynEditCaretType', iptrw);
31848:   InsertMode', 'boolean', iptrw); 'IsScrolling', 'Boolean', iptr);
31849:   Keystrokes', 'TSynEditKeyStrokes', iptrw);
31850:   MaxUndo', 'Int', iptrw); 'Options', 'TSynEditorOptions', iptrw);
31851:   OverwriteCaret', 'TSynEditCaretType', iptrw);
31852:   RightEdge', 'Int', iptrw); RightEdgeColor', 'TColor', iptrw);
31853:   ScrollHintColor', 'TColor', iptrw);
31854:   ScrollHintFormat', 'TScrollHintFormat', iptrw);
31855:   ScrollBars', 'TScrollStyle', iptrw);
31856:   SelectedColor', 'TSynSelectedColor', iptrw);
31857:   SelectionMode', 'TSynSelectionMode', iptrw);
31858:   ActiveSelectionMode', 'TSynSelectionMode', iptrw);
31859:   TabWidth', 'Int', iptrw);  WantReturns', 'boolean', iptrw);
31860:   WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
31861:   WordWrapGlyph', 'TSynGlyph', iptrw);
31862:   OnChange', 'TNotifyEvent', iptrw);
31863:   OnClearBookmark', 'TPlaceMarkEvent', iptrw);
31864:   OnCommandProcessed', 'TProcessCommandEvent', iptrw);
31865:   OnContextHelp', 'TContextHelpEvent', iptrw);
31866:   OnDropFiles', 'TDropFilesEvent', iptrw);
31867:   OnGutterClick', 'TGutterClickEvent', iptrw);
31868:   OnGutterGetText', 'TGutterGetTextEvent', iptrw);
31869:   OnGutterPaint', 'TGutterPaintEvent', iptrw);
31870:   OnMouseCursor', 'TMouseCursorEvent', iptrw);
31871:   OnPaint', 'TPaintEvent', iptrw);
31872:   OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
31873:   OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
31874:   OnReplaceText', 'TReplaceTextEvent', iptrw);
31875:   OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
31876:   OnStatusChange', 'TStatusChangeEvent', iptrw);
31877:   OnPaintTransient', 'TPaintTransient', iptrw);
31878:   OnScroll', 'TScrollEvent', iptrw);
31879:  end;
31880:  Proc RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
31881:  Func GetPlaceableHighlighters : TSynHighlighterList
31882:  Func EditorCommandToDescrString( Cmd : TSynEditorCommand) :Str
31883:  Func EditorCommandToCodeString( Cmd : TSynEditorCommand) :Str
31884:  Proc GetEditorCommandValues( Proc : TGetStrProc)
31885:  Proc GetEditorCommandExtended( Proc : TGetStrProc)
31886:  Func IdentToEditorCommand( const Ident :Str; var Cmd : longint) :Bool
31887:  Func EditorCommandToIdent( Cmd : longint; var Ident :Str) :Bool
31888:  Func ConvertCodeStringToExtended( AString :Str) :Str
31889:  Func ConvertExtendedToCodeString( AString :Str) :Str
31890:  Func ConvertExtendedToCommand( AString :Str) : TSynEditorCommand
31891:  Func ConvertCodeStringToCommand( AString :Str) : TSynEditorCommand
31892:  Func IndexToEditorCommand( const AIndex : Int) : Int
31893: TSynEditorOption = (
31894:  eoAltSetsColumnMode,       //Holding down Alt Key will put select mode into columnar format
31895:  eoAutoIndent,        //Will indent caret onnewlines same amountof leading whitespace as
31896:                            // preceding line
31897:  eoAutoSizeMaxScrollWidth,  //Automatically resizes MaxScrollWidth property when insert text
31898:  eoDisableScrollArrows,     //Disables scroll bar arrow buttons when you cant scroll that
31899:                            //direction any more
31900:  eoDragDropEditing,  //Allows to select a textblock and drag in doc to another location
31901:  eoDropFiles,               //Allows the editor accept OLE file drops
31902:  eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
31903:  eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
31904:  eoGroupUndo, //When undoing/redoing actions,handle all cont.changes same kind in onecall
31905:                            //instead undoing/redoing each command separately
31906:  eoHalfPageScroll,   //By scrolling with page-up/page-down commands, scroll half page attime
31907:  eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
```

```
31908:  If you have ScrollPastEOL,then it horizontal bar will always there (uses MaxLength instead)
31909:  eoKeepCaretX,         //When moving through lines w/o cursor Past EOL, keeps X posof cursor
31910:  eoNoCaret,                //Makes it so the caret is never visible
31911:  eoNoSelection,            //Disables selecting text
31912:  eoRightMouseMovesCursor,//When clicking right mouse for popup menu,moves cursor to location
31913:  eoScrollByOneLess,        //Forces scrolling to be one less
31914:  eoScrollHintFollows,      //The scroll hint follows the mouse when scrolling vertically
31915:  eoScrollPastEof,          //Allows the cursor to go past the end of file marker
31916:  eoScrollPastEol,          //Allows cursor go past lastchar into white space at end ofline
31917:  eoShowScrollHint,  //Shows a hint of the visible line numbers when scrolling vertically
31918:  eoShowSpecialChars,       //Shows the special Characters
31919:  eoSmartTabDelete,         //similar to Smart Tabs, but when you delete characters
31920:  eoSmartTabs,              //When tabbing, cursor will goto non-whitespace charof prev line
31921:  eoSpecialLineDefaultFg,//disables foreground textcolor override  OnSpecialLineColor event
31922:  eoTabIndent,     //If active <Tab>and<Shift><Tab> act block indent,unindent iftext select
31923:  eoTabsToSpaces, //Converts a tab character to a specified number of space characters
31924:  eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
31925:
31926:  Change above Options in Editor (for bootscript also):
31927:  for eg. to indent blocks with tab:
31928:
31929:  memo1.options:= memo1.options + [eoTabIndent]
31930:  or to control something:
31931:  memo1.wanttabs:= true;
31932:  if (memo1.wanttabs) and (GetKeyState(VK_CONTROL)) then writeln('wanttabs0^');
31933:
31934: *******************Important Editor Short Cut Cuts shorcut****************);
31935:  Double click to select a word and count words with highlightning.
31936:  Triple click to select a line.
31937:  CTRL+SHIFT+click to extend a selection.
31938:  Drag with the ALT key down to select columns of text !!!
31939:  Drag and drop is supported.
31940:  With CTRL 3 to jump to the last change  (change tracker)
31941:  Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
31942:  Type CTRL+A to select all.
31943:  Type CTRL+N to set a new line.
31944:  Type CTRL+T to delete a line or token. //Tokenizer
31945:  Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
31946:  Type CTRL+Shift+S to Save as....
31947:  Type CTRL+Alt+M to jump to Memo Terminal.
31948:  Type CTRL+Shift+T to add ToDo in line and list.
31949:  Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
31950:  Type CTRL[0..9] to jump or get to bookmarks.
31951:  Type Home to position cursor at beginning of curr line and End to position at end of line.
31952:  Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of docu.
31953:     Page Up and Page Down work as expected.
31954:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
31955:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
31956:
31957: {$ Short Key Positions Ctrl<A-Z>: }
31958: def
31959:    <A> Select All
31960:    <B> Count Words
31961:    <C> Copy
31962:    <D> Internet Start
31963:    <E> Script List
31964:    <F> Find
31965:    <G> Goto
31966:    <H> Mark Line
31967:    <I> Interface List
31968:    <J> Code Completion
31969:    <K> Console
31970:    <L> Interface List Box
31971:    <M> Font Smaller -
31972:    <N> New Line
31973:    <O> Open File
31974:    <P> Font Larger +
31975:    <Q> Quit
31976:    <R> Replace
31977:    <S> Save!
31978:    <T> Delete Line
31979:    <U> Use Case Editor
31980:    <V> Paste
31981:    <W> URI Links
31982:    <X> Reserved for coding use internal
31983:    <Y> Delete Line
31984:    <Z> Undo
31985:
31986: ref F1 Help to Func List
31987:     F2 Syntax Check
31988:     F3 Search Next
31989:     F4 New Instance
31990:     F5 Line Mark /Breakpoint
31991:     F6 Goto End
31992:     F7 Debug Step Into
31993:     F8 Debug Step Out
31994:     F9 Compile
31995:     F10 Menu
31996:     F11 Word Count Highlight
```

```
31997:     F12 Reserved for coding use internal
31998:
31999:    AddRegisteredVariable('Application', 'TApplication;
32000:    AddRegisteredVariable('Screen', 'TScreen;
32001:    AddRegisteredVariable('Self', 'TForm;
32002:    AddRegisteredVariable('Memo1', 'TSynMemo;
32003:    AddRegisteredVariable('memo2', 'TMemo;
32004:    AddRegisteredVariable('maxForm1', 'TMaxform1;  //!!
32005:    AddRegisteredVariable('debugout', 'Tdebugoutput;  //!!
32006:    AddRegisteredVariable('hlog','THotlog;  //!!
32007:    AddRegisteredVariable('mouse','TMouse;  //!!
32008:    AddRegisteredVariable( it ,Int;  //for closure and each loop!!
32009:    AddRegisteredVariable( sr ,string;  //for closure
32010:    AddRegisteredVariable( bt ,boolean;  //for closure
32011:    AddRegisteredVariable( ft ,double;  //for closure
32012:    AddRegisteredVariable( srlist ,TStringlist;  //for closures
32013:
32014:  def ReservedWords: array[0..86] of string =
32015:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
32016:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
32017:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
32018:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
32019:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
32020:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
32021:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
32022:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
32023:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
32024:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected', xraise
32025:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
32026:    AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3:Bool;
32027: //-----------------------------------------------------------------------------
32028: //**************End of mX4 Public  Tools API ***********************************
32029: //-----------------------------------------------------------------------------
32030:
32031: maXbox 4 Internal Now Inventory INI Nov 2017
32032: Amount of Functions: 18547
32033: Amount of Procedures: 11143
32034: Amount of Constructors: 1730
32035: Totals of Calls: 31420
32036: SHA1: of 4.5.8.10 FFF8300142CE582DFCC8E15489AD7D5EE2E08062
32037: CRC32: A013C5BC:  28465616 bytes
32038: ▯▯▯ mX4 executed: 03/01/2018 17:10:09  Runtime: 0:8:5.816  Memload: 29% use
32039: maXbox 4 Internal Now Inventory INI Dez 2019
32040: Amount of Functions: 18929
32041: Amount of Procedures: 11351
32042: Amount of Constructors: 1771
32043: Amount of Destructors: 12
32044: Totals of Calls: 32063
32045: SHA1: of 4.7.1.80 E372D00CE31BDFC30A29F6C1F77600ABD25AFDB8
32046: CRC32: 3C620310:  28952992 bytes
32047: ▯▯▯ mX4 executed: 03/12/2019 23:15:23  Runtime: 0:9:29.13  Memload: 48% use
32048: ▯▯▯ V 4.7.4.64
32049: Totals of Func Calls: 32633, 75 Tutorials
32050: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
32051: CRC32: 3EB27A87: 28.2 MB (29,608,248) bytes
32052:
32053: ************************************************************
32054:  Doc Short Manual with 70 Tips! Troublshooter - Errors
32055: ************************************************************
32056: - Editor: dont ever load and save a BINARY file in the editor, it saves as CRLF text !
32057: - Install: just save your maxboxdef.ini before and then extract the zip file!
32058: - Toolbar: Click on the red maXbox Sign (right on top) opens your work dir or jump to <Help>
32059: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
32060: - Menu: With <Crtl><F3> you can search for code on examples
32061: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
32062: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
32063: - Menu: Switch or toogle between last 2 scripts in menu File/LoadLast(History is set to 9 files)
32064:
32065: - Inifile: Set memory report in ini:MEMORYREPORT=Y :report on memoryleaks on shutdown by dialog
32066: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
32067: - Inifile: Set VERSIONCHECK=N and MEMORYREPORT=N to speedup boot and shutdown of box
32068:
32069: - Context Menu: You can printout your scripts as a pdf-file or html-export
32070: - Context: You do have a context menu with the right mouse click
32071:
32072: - Menu: With the UseCase Editor you can convert graphic formats too.
32073: - Menu: On menu Options you find Addons as compiled scripts
32074: - IDE: Menu Program: Run Only is faster, after F2 - You dont need a mouse use shortcuts
32075: - Menu: Check Options/ProcessMessages! if something is wrong or you cant see graphics in a time
32076: - IDE: Dragndrop scripts in box or model in use case editor(Cut,Copy,Paste always available)
32077: - 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>
32078:
32079: - Menu: In menu output (console) set output menu in edit mode by unchecking <read only output>
32080: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
32081: - Code: If you code a loop till key-pressed use function: isKeyPressed;
32082: - Code: Macro set the macros #name,dmMaxMaxinistratorth, #file,s.txtmaxbox_functions.txtist5.pdf
32083: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
32084: - 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
```

```
- Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
- IDE: A file info with system and script information you find in menu Program/Information
- IDE: After change the config file in help you can update changes in menu Help/Config Update
- IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
- IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time start it.
- IDE: With escape or <Ctrl> Q you can also leave box or stopscript in menu program - stop program
- Editor: Set Bookmarks to check your work in app or code
- Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
- Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
- Editor: With <Ctrl W> you set active URL links in code to test availability in Context Menu
- Editor  shortcut Ctrl+Alt+M to jump to output (memo2) console
- IDE with  menu /Options/ADO SQL Workbench you can manage your Database
- Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
- Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
- Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
- Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
- Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
- Code: if you cant run a Func try the second one, for ex. Voice()-Voice2(),inc() - inc1()
- IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
- IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
- IDE menu /Help/Tools/ open the Task Manager
- Editor: set doc section after end. end. in units and interface with begin main end!
- new Style: menu//Output/Darkcolor
  http://www.softwareschule.ch/images/maXbox4_darkcolor.png
  http://www.softwareschule.ch/images/maXbox4_darkcolor3.png

- Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
- Add on when no browser is available start /Options/Add ons/Easy Browser
- Add on SOAP Tester with SOP POST File
- Add on IP Protocol Sniffer with List View
- Add on OpenGL mX Robot Demo for android
- Add on Checkers Game, Add on Oscilloscope /View GEO Map View3

- Menu: Help/Tools as a Tool Section with DOS Opener
- Menu Editor: export the code as RTF File
- Menu: Help/Tools/Syn Export your code as available in HTML or RTF
- Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
- Context: Auto Detect of Syntax depending on file extension
- Code: some Windows API Func start with w in the name like wGetAtomName();
- IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
- IDE File Check with menu ..View/File Changes/...
- Context: Create a Header with Create Header in Navigator List at right window
- Code: use  SysErrorMessage to get a real Error Description, Ex.
  RemoveDir('c:\NoSuchFold;writeln('System Error Message:'+SysErrorMessage(GetLastError));
- IDE getWebScript from a URL with menu ..Help/Get Web Script/...
- Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click on link
- Editor: with <Ctrl+Alt+R> you can write in RTF format with RichEdit link
- Editor: with <Ctrl+Alt+T> you can fill your To Do List.
- Editor: with <Ctrl+Alt+M> you jump to output window below (memo2).
- Menu: Check Help/Tools! you can use richedit, DOS Shell or Explorer
- Menu: Start View/MyScript of ini file maxboxdef.ini [MYSCRIPT]= path of script
- Indent: close you doc lines with //}    for {...}
- Menu: Help/ToDo List add prio and user with {/1.......//}
            Name#Prio# example:
  {/1 TODO (maxname#7#): do the new task one34 then mX4' //}

- With this code I get an EIdHTTPProtocolException with the message "HTTP/1.1 403 Forbidden".
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.

   with TIdHTTP.Create(self)  do begin
      Request.UserAgent:= 'maXbox4 compatible';
      get('http://www.softwareschule.ch/examples/robo.txt')
      free
   end;

use for TRegistry.create constructor TRegistry.Create1
in maxbox_functions.txt add TRegistry.Create1

 //In the ServiceAfterUninstall method add these lines:
 // begin
  // Delete registry entries for event viewer.
  Key:= '\SYSTEM\CurrentControlSet\Services\Eventlog\Application\' + Self.Name;
  Reg:= TRegistry.Create1(KEY_READ or KEY_WRITE);
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.KeyExists(Key) then
      Reg.DeleteKey(Key);
  finally
    Reg.Free;
  end;

Func AnsiByteArrayStringLen2(Data: TBytes): Integer;
Func StringToAnsiByteArray2(const S:Str): TBytes;
Func AnsiByteArrayToString2(const Data: TBytes; Count: Integer):Str;
Func BytesOf(const Value: Ansistr): TBytes;
Func BytesOfChar(const Value: AnsiChar): TBytes;
Func StringOf(const Bytes: array of Byte): Ansistr;
```

```
32173: - using DLL example in maXbox: //function: {**********************}
32174:  Func GetProcessMemoryInfo(Process:THandle;var
       MemoryCounters:TProcessMemoryCounters;cb:DWORD):BOOL;//stdcall;
32175:     External 'GetProcessMemoryInfo@psapi.dll stdcall';
32176:  Func OpenProcess(dwDesiredAccess:DWORD;bInheritHandle:BOOL;dwProcessId: DWORD):THandle;
32177:     External  'OpenProcess@kernel32.dll stdcall';
32178:
32179: Win API direct DLL with array of char result:
32180:
32181: //https://docs.microsoft.com/en-us/windows/win32/api/shlobj_core/nf-shlobj_core-shgetfolderpatha
32182:
32183:   function  SHGetFolderPathA(hwnd: HWND; csidl: integer; htoken: THANDLE;
32184:                              dwflags: DWORD; pszPath: string): HRESULT;
32185:    external 'SHGetFolderPathA@Shell32.dll stdcall';
32186:
32187: function GetMyPictures: string;
32188: var pStr: string; //array[0..260] of Char;
32189: begin
32190:   setLength(pstr, 260-1);
32191:   if SHGetFolderPathA(0, CSIDL_MYPICTURES, 0,0, pStr) = S_OK then
32192:     Result:= pStr;
32193: end;
32194:
32195: >>> GetMyPictures path C:\Users\Max\OneDrive\Pictures
32196:
32197: GCC Compile Ex Script
32198: Proc TFormMain_btnCompileClick(Sender: TObject);
32199: begin
32200:   AProcess:= TProcess.Create(Nil);
32201:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
32202:       + ' -o "' + OpenDialog2.FileName + '"';
32203:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
32204:     AProcess.Execute;
32205:     Memo2.Lines.BeginUpdate;
32206:     Memo2.Lines.Clear;
32207:     Memo2.Lines.LoadFromStream(AProcess.Output);
32208:     Memo2.Lines.EndUpdate;
32209:   finally
32210:     AProcess.Free;
32211:   end;
32212: end;
32213:
32214: ref Stopwatch pattern snip
32215: Time1:= Time;
32216:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
32217:   if initAndStartBoard then
32218:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
32219:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
32220:   PrintF('%d %s',[Trunc((Time-Time1)*24),FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
32221:   POST git-receive-pack (chunked)
32222: Pushing to https://github.com/maxkleiner/maXbox3.git
32223: To https://github.com/maxkleiner/maXbox3.git f127d21..c6a98da  masterbox2 -> masterbox2
32224: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
32225:
32226: ---------------------------------------------------------------------------
32227: History Shell Hell - Walk the Talk - CoolCode
32228: PCT Precompile Technology , mX4 ScriptStudio
32229: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
32230: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
32231: emax layers: system-package-component-unit-class-function-block
32232: new keywords def ref using maXCalcF UML xraise: use case act class state seq pac comp dep - lib lab
32233: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
32234: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
32235: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
32236: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
32237: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
32238: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
32239: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
32240: SendBuffer, Color+Caption hack,ComboSet,SetPrivilege,WakeOnLAN,ParaDice 3DCube Polygraph,OCR
32241: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
32242: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
32243: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
32244: Inno Install and Setup Routines Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
32245: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
32246: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
32247: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
32248: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
32249: Func Menu/View/GEO Map View, DownloadFile, wgetX, sensors
32250: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
32251: ByteCode2, IPUtils2, GEOCode, CGI-Powtils, GPS_2, External App, Unit Converter
32252: Add 16 Units, 1 Slide,Tutor, Big Numbers (Decimals, TInt), ModBusTCP, TGEOInfo
32253: Add 36 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, ACM, CDS, XMLDoc, DDE
32254: Add 77 Units, 12 Tutors, ChangeTracker, CPP+,OLEUtils2,xmldom,Chess4,3DFrame,XMLRPC,X509
32255: Add 15 Units, 1 Tutor, Pipe Libraray2, KLog, FPlot42, KDialogs, NumEdit, KControls, Kronos
32256: add 12 units and 125 functions - Class Helper - KMemo RTF
32257: add 16 units and 365 functions- WMI Script Type Library - webbox -restbox
32258: V4.6.2.10 Jan 2018:Tutor 56 Neural Network-Python Checker- 3 more Units PascalCoin
32259: V4.7.1.10 Sep 2019:Tutor 57-70, VState Machine,CGI,MachineLearning +20 Units +Tensorflow dll
32260: V4.7.1.82 Dec 2019:Tutor-72,EKON23Fixing,WebSocks/Spider,PHP_CGI,JS,OpenOffice,7+overbyte Units
```

```
32261: V4.7.4.64 June 2020: EKON24 Fixing, uPSI_SimpleRSS, Json Base Prometheus, neural CAI, Dendron
32262: V4.7.5.20 Jan 2021: few Fundamentals 5.00, JCL fixes, GraphMathLibrary, StringBuilder
32263: V4.7.5.80 July 2021: RSS+, WDCC, P4D_Beta (Python4Delphi), Wininet Threads<br>
32264: V4.7.5.90 October 2021: FLC Vectors, CAI Neuralnet, SingleListClass<br>
32265: V4.7.6.20 October 2022: Threadslist,CAI Neuralnet3,P4D+,EKON25 fixing,klib,HTTPClient2, -c CLI, PSProc<br>
32266: Ref:
32267: https://unibe-ch.academia.edu/MaxKleiner
32268: http://www.slideshare.net/maxkleiner1
32269: http://www.scribd.com/max_kleiner
32270: http://www.delphiforfun.org/Programs/Utilities/index.htm
32271: http://www.slideshare.net/maxkleiner1
32272: http://s3.amazonaws.com/PreviewLinks/22959.html
32273: http://www.softwareschule.ch/arduino_training.pdf
32274: http://www.jrsoftware.org/isinfo.php
32275: http://www.be-precision.com/products/precision-builder/express/
32276: http://www.blaisepascal.eu/
32277: http://www.delphibasics.co.uk/
32278: http://www.youtube.com/watch?v=av89HAbqAsI
32279: http://www.angelfire.com/hi5/delphizeus/modal.html
32280: http://www.retroarchive.org/garbo/pc/turbopas/index.html
32281: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
32282: http://delphi.org/2014/01/every-android-api-for-delphi/
32283: https://en.wikipedia.org/wiki/User:Maxkleiner
32284: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
32285: https://bitbucket.org/max_kleiner/maxbox3
32286: https://bitbucket.org/max_kleiner/maxbox3/downloads
32287: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
32288: http://www.slideshare.net/maxkleiner1/codereview-topics
32289: http://www.abaecker.biz/images/abakus/images/TAbCompass.gif
32290: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
32291: https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
32292: https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
32293: http://max.kleiner.com/boxart.htm
32294: https://maxbox4.wordpress.com
32295:
32296: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
32297: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
       =renderBasicSearchNarrative&q=%s';
32298: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
32299: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
32300:
32301: SAMPLES Snippets Blocks and Error or Errors or Bugs or Bug List or Trouble Bug Fix Errorlist bugfix codes
32302: ------------------------------------------------  Troubleshooter
32303:
32304: [Error] C:\maXbox\TestApp2\sonysavefeb2014\TestApp\tetris\175_pas_anti_tetris2.txt(1:1): Variable Expected
32305: dont inc an longint in plus Inc(Score+5); ---> solution: Inc(Score);
32306:
32307: Editor: index out of bounds error exception in editor after open or modified
32308: 03/12/2019 20:10:50 V:4.7.1.20 [Max] MAXBOX10 List index out of bounds (147).
32309:
32310: This is due of Editor Indent  ../Options/Show Indent
32311: 4 Workarounds:
32312: 1 simply move cursor to right or left and save (Ctrl+S)
32313: 2 Goto to end or line (147) or what it says and return new line or tab
32314: 3 Disable <Options Show Indent> in menu
32315: 4 Set INDENT=N in maxboxdef.ini for runtime systems
32316:
32317: can not see the Form , Form not visible ---> add Show;
32318: frm:= TmyForm.create(self)
32319: with frm do begin
32320:   Left := 245
32321:   Top := 108
32322:   BorderStyle := bsDialog
32323:   Caption := 'Suggestions'
32324:   ClientHeight := 219
32325:   ClientWidth := 297
32326:   Color := clBtnFace
32327:   //TextHeight := 20
32328:   Show; ------------------------->
32329:
32330: Exception Handling Raise and XRaise
32331: type EPowerException = Exception;
32332:   ...end else
32333:        Xraise( EPowerException.Create('Infinite Result'));
32334:      end;
32335:      if (X > 0) then
32336:      try
32337:        result := exp(N * ln(X));
32338:        exit;
32339:      except
32340:        raise EPowerException.Create('Overflow/Underflow Result');
32341:      end;
32342:
32343: - Object with an Record inside get first a record instance and second you get the attribute
32344:
32345: Proc testprimesPerformance;   // Performance Test
32346: var count, beg, j, r: integer;
32347: var f :boolean;
32348: begin
```

```
32349:    processmessagesOFF
32350:    Println(' prime time performance check ?????: ');
32351:    count:= 0;
32352:    beg:= Random(1000000000)+2;
32353:    for it:=beg to beg+5000 do begin
32354:      f:= True;
32355:      j:= 2;
32356:      r:= Round(Sqrt(it));
32357:      while f and (j<=r) do
32358:      if it mod j = 0 then f:= False
32359:      else inc(j);
32360:      if f then begin
32361:        Print(itoa(it)+' ');
32362:        count:= count+ 1;
32363:        if count mod 8 = 0 then
32364:        Println('');
32365:      end;
32366:    end;
32367:    processmessagesON;
32368: end;
32369:
32370: RSS-FEED Rss Feed
32371: IIndyHTTP:=TIdHTTP.create(self);
32372:     with TSimpleRSS.create(self) do begin
32373:       XMLType:= xtRDFrss;
32374:       IndyHTTP:= iindyhttp;
32375:       LoadFromHTTP(RSS_NewsFeed)
32376:       for it:= 0 to items.count-1 do begin
32377:          //Items[it].description
32378:          writeln(itoa(it)+': '+Items[it].title+'  :'{+items[it].link});
32379:          println(GetSentimentStream7(URLSentimentAPI2,Items[it].title));
32380:       end;
32381:       free;
32382:       IIndyHTTP.Free;
32383:     end; //}
32384:
32385: Ver: 4.7.4.64 (474). Workdir: C:\maXbox\mX47464\maxbox4
32386: ▯▯▯ mX4 executed: 05/11/2020 17:57:05  Runtime: 0:0:39.224  Memload: 60% use
32387:
32388: Func OpenMap(const Data:Str):Bool;
32389: var encURL:Str;
32390: begin
32391:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
32392:   try  //HttpGet(EncodedURL, mapStream);   //WinInet
32393:   Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
32394:    //OpenDoc(Exepath+'openmapx.html;
32395:    S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
32396:   finally encURL:= '';
32397:   end;
32398: end;
32399:
32400: Proc GetGEOMap(C_form,apath:Str; const Data:Str);
32401: var encodedURL:Str; mapStream: TMemoryStream;
32402: begin //encodedURL:= Format(UrlGoogleQrCode,[Width,Height,C_Level, HTTPEncode(Data)]);
32403:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
32404:   mapStream:= TMemoryStream.create;
32405:   try
32406:     Wininet_HttpGet(EncodedURL, mapStream);   //WinInet
32407:     mapStream.Position:= 0;
32408:     mapStream.Savetofile(apath); // OpenDoc(apath);
32409:     S_ShellExecute(apath,'',seCmdOpen);
32410:   finally
32411:     mapStream.Free;
32412:   end;
32413: end;
32414:
32415: Web Request API HTTPSend:
32416:
32417: //http://synapse.ararat.cz/doc/help/httpsend.THTTPSend.html#@Description
32418:
32419: Const aURL='https://api.metadefender.com/v4/file/bzIwMDYxNi1TSW42NDBPVlprTWw3YjRBMQ';
32420:
32421: Begin  //@main
32422:     srlist:= TStringlist.create;
32423:     with THTTPSend.create() do begin
32424:       Headers.Add('apikey: 6b337c92c792174a54acd715ab1aae64');
32425:       writeln(botostr(HTTPMethod('GET',aURl)));
32426:       with TJson.create() do begin
32427:         Parse(StreamtoString3(Document));
32428:         Split(Stringify,'{',srlist)
32429:       end;
32430:       writeln(itoa(resultCode)+':'+resultString+CRLF+srlist.text);
32431:       clear;free;
32432:     end;
32433:     srlist.Free;
32434:
32435:
32436: Proc BtnFactory(a,b,c,d:smallint; title,apic:str;
32437:                              var abtn:TBitBtn; anEvent:TNotifyEvent; afrm:TForm);
```

```
32438: begin
32439:   abtn:= TBitBtn.create(afrm);
32440:   with abtn do begin
32441:     parent:= afrm;
32442:     setBounds(a,b,c,d)
32443:     font.size:= 12;
32444:     glyph.LoadFromResourceName(HINSTANCE, apic);
32445:     mXButton(5,5,width, height,12,12,handle);
32446:     caption:= title;
32447:     onClick:= anEvent As TNotifyEvent;
32448:   end;
32449: end;
32450:
32451: Func GetSystemDirectory2(var S:Str):Bool;
32452: var Len: Int;
32453: begin
32454:   Len:= {Windows.}GetSystemDirectory('Nil', 0);
32455:   if Len > 0 then begin
32456:     SetLength(S, Len);  //writeln(itoa(len))
32457:     Len:= {Windows.}GetSystemDirectory(PChar(S), Len);
32458:     SetLength(S, Len);  //writeln(itoa(len))
32459:     Result:= Len > 0;
32460:   end else
32461:     Result:= False;
32462: end;
32463:
32464: Func RecordsetToXML2(const Recordset: variant):Str;
32465: var
32466:   RS: Variant;
32467:   Stream: TStringStream;
32468: begin
32469:   Result := '';
32470:   //if Recordset = unassigned then Exit;
32471:   Stream:= TStringStream.Create(';
32472:   try
32473:     RS:= CreateOleObject('ADODB.Recordset;
32474:     RS:= Recordset;
32475:     RS.Save(TStreamAdapter.Create(stream,soReference) as IUnknown, adPersistXML);
32476:     Stream.Position := 0;
32477:     Result:= Stream.DataString;
32478:   finally
32479:     Stream.Free;
32480:   end;
32481: end;
32482:
32483: convert JPEG to Bitmap
32484: if version = '4.7.6.20' then begin
32485:   JPGImage:= TJPEGImage.Create;
32486:   try
32487:     JPGImage.LoadFromStream(TResourceStream.Create(hInstance,'Everest','JPEG'));
32488:     with TImage.create(form1) do begin
32489:       setbounds(50,500,500,200)
32490:       parent:= form1;
32491:       Picture.Graphic:= JPGImage;
32492:       stretch:= true;
32493:     end;
32494:   finally
32495:     JPGImage.Free;
32496:   end; //}
32497: end;
32498:
32499: Func BigFactSum(n: integer):Str;
32500:   var tbig1, tbig2, tbig3: TInteger;
32501:   begin
32502:     result:= '0'
32503:     tbig1:= TInteger.create(1);  //facultry
32504:     tbig2:= TInteger.create(0);  //assign
32505:     tbig3:= Tinteger.create(0);
32506:     for it:= 1 to n do begin
32507:       tbig2.assign1(it)
32508:      tbig1.mult(tbig2)
32509:     tbig3.add(tbig1);
32510:   end;
32511:     result:= tbig3.tostring(false)
32512:     tbig3.free;
32513:     tbig2.free;
32514:     tbig1.free;
32515:   end;
32516:
32517: No helper class possible or predict:
32518: USES Classes,StrUtils;
32519:
32520: TYPE
32521: TStringListHelper = CLASS HELPER FOR TStrings
32522: Func ToSQL :Str;
32523: END;
32524: Func TStringListHelper.ToSQL :Str;
32525: VAR S :Str;
32526:
```

```
32527: Func QuotedStr(CONST S :Str) :Str;
32528: BEGIN
32529: Result:=''''+ReplaceStr(S,'''','''''')+''''
32530: END;
32531:
32532: BEGIN
32533: Result:='';
32534: FOR S IN Self DO BEGIN
32535: IF Result='' THEN Result:='(' ELSE Result:=Result+',';
32536: Result:=Result+QuotedStr(S)
32537: END;
32538: IF Result<>'' THEN Result:=Result+')'
32539: END;
32540:
32541: SL:=TStringList.Create;
32542: SL.Add('One');
32543: SL.Add('Two');
32544: SL.Add('Number Three');
32545: SL.Add('It''s number 4');
32546: WRITELN('SELECT * FROM TABLE WHERE FIELD IN '+SL.ToSQL);
32547: SELECT * FROM TABLE WHERE FIELD IN ('One','Two','Number Three','It''s number 4'}
32548:
32549: RSA Crypto Demo
32550: 965_RSA_IBZ_PrivatePublicKey_Form.pas
32551: const _n = '95163118457906561534997167608470014334413 57'; //# p*q = modulus
32552:       _e = '65537';
32553:       _d = '56178431878449531703084636222302833762986 85';
32554:   abig:=TInteger.create(1);
32555:   abig.Assignhex(ASCIItohex(message));
32556:   writeln('big plain text integer: '+abig.Tostring(normal))
32557:   s_encrypt:= BigPowMod(abig.Tostring(normal),_e, _n)     //bigmulu(ap,aq));
32558:   writeln('encrypted text integer '+s_encrypt)
32559:   s_decrypt:= BigPowMod(s_encrypt,_d,_n)                  //bigmulu(ap,aq))
32560:   writeln('decrypted text integer '+s_decrypt)
32561:
32562: Func ConvertHex(HexString :Str; Len : integer) :Str;
32563: begin
32564: if trim(HexString) = '' then Result := '0' else
32565:   Result := IntToStr(StrToInt('$' + trim(HexString)));
32566:    while length(Result) < Len do Result := '0' + Result;
32567: end;
32568:  Proc SetBit(const a,b:Card);
32569:   var d: Int64; // absolute dest;
32570:  begin
32571:     d:= $F000000000+(Int64(a) shl 21)+b shl 6;
32572:  end;
32573:
32574: var alabel: TComponent;   //Exception: A component named lblCell0 already exists.
32575:     lc: integer;
32576: begin
32577: //object frMain: TfrMain
32578:     if assigned(TLabel(FindComponent('lblCell'+intToStr(lc)))) then begin
32579:       write('exist&Nil ')
32580:      TLabel(FindComponent('lblCell'+ IntToStr(lc))).Name:= '';
32581:      alabel:= TLabel(FindComponent('lblCell'+ IntToStr(lc)));
32582:      alabel:= Nil;
32583:      end;
32584:    end;
32585:
32586: //Discuss the Recursion of findfile!
32587: Proc FindFilePattern3(root:str; pattern:str);
32588: var SR: TFindRec;
32589: begin
32590:   root:=IncludeTrailingPathDelimiter(root);
32591:   if FindFirst3(root+'*.*',SR) = TRue then begin
32592:   repeat
32593:    Application.ProcessMessages;
32594:      if ((sr.attributes and faDirectory)= SR.Attributes) and
32595:        (pos('.',SR.name)=0) then
32596:         FindFilePattern3(root+SR.Name,pattern)
32597:      else begin
32598:       if pos(pattern,SR.Name)>0 then
32599:         //Form1.ListBox1.Items.Add(Root+SR.Name);
32600:         writeln(Root+SR.Name);
32601:      end;
32602:   until FindNext3(SR) = False;
32603:     FindClose3(SR);
32604:   end;
32605: end;
32606:
32607: Encoding Decoding
32608: const tname= 'Subject: =?iso-8859-1?Q?Die_besten_W=FCnsche_zum_Neuen_Jahr_2021?=';
32609:
32610:   writeln('mimecharset '+CharsetConversion(tn,ISO_8859_1, UTF_8));
32611:   writeln('mimecharset '+CharsetConversion(tname,ISO_8859_1, win_1252));
32612:   writeln( MimeCharsetToCharsetString(ISO_8859_1) );
32613:
32614:   solution: writeln(DecodeHeader(tname)); or writeln(ForceDecodeHeader(tname));
32615:   -----> Subject: Die besten Wünsche zum Neuen Jahr 2021
```

```
32616:   writeln(UTF8toAnsi(ALUTF8HTMLdecode('Kaufbest&auml;tigung')))
32617:   -----> Kaufbestätigung
32618:
32619: RegEx by example:
32620: The Regexlibrary is inbuilt so it works independent of Operating System:
32621:
32622: Proc RegEX_ExtractDemo;
32623: var regEx: TRegExpr; //or OleVariant if HISUtils;
32624: InStr, ResStr:Str;
32625: begin
32626:    ResStr:='';
32627:    InStr:= 'Please e-mail us at support@mycompany.com or to sales@mycompany.com';
32628:    // Create a regular expression
32629:    regEx:= TRegExpr.create; //HISUtils.RegExpr;
32630:    // Set regular expression pattern that specifies an e-mail address
32631:    regEx.Expression:='\w+(\.[\w-]+)*@([\w-]+\.)+[a-zA-Z]{2,6}';
32632:    // Execute search
32633:    if regEx.Exec(InStr) then Repeat
32634:      ResStr:= ResStr + regEx.Match[0] + '; ';
32635:    Until not regEx.ExecNext;
32636:
32637:    //Log.Message('These e-mails were extracted: '+ResStr);
32638:    writeln('These e-mails were extracted: '+ResStr);
32639:    // Posts the following to the log:
32640:    //'These e-mails were extracted: support@mycompany.com; sales@mycompany.com;'
32641: end;
32642:
32643: Proc LoadFilesByMask(lst: TStrings; const SpecDir, WildCard:Str);
32644: var SearchRec: TFindRec;
32645: begin lst.Clear;
32646:   if FindFirst3(SpecDir + WildCard, SearchRec) then
32647:   repeat
32648:     lst.Add(SpecDir + SearchRec.Name);
32649:   until FindNext3(SearchRec) = false;
32650:   FindClose3(SearchRec);
32651: end;
32652:
32653: Proc regExtest;     //SynRegExpr unit     RegEx Snippet
32654:  var RegEx: TRegExpr; // This is a class, need to be freed!
32655:    //Match: TMatch;
32656:    i: Integer;
32657:  begin
32658:    with TRegExpr.Create do begin
32659:      Expression:= ('^v(ol\.?|olume)?\s*([0-9]+)$');
32660:      if Exec('vol.3456') then
32661:        for i:=0 to SubExprMatchCount do
32662:           PrintF('Group %d  : %s', [i, Match[i]]);
32663:       //Until not regex.ExecNext;
32664:      Free;
32665:    end
32666:  end;
32667:
32668:  //(?i) ---> case insensitive
32669:  //(?i) case-insensitive mode ON
32670:  //(?-i) case-insensitive mode OFF
32671:  writeln(getMatchString('(?i)G[a-b].*', 'this GB string'));
32672:  PrintF('CountPos: %d',[CountPos(uppercase('Selftest'),uppercase(sr))])
32673:
32674:  Set Mapping
32675:  There are some differences in the use of set operators in PascalScript and Delphi.
32676:  In DelphiScript the [...] set operator is not supported or the in operator not always works.
32677:  Set mapping with ord(CS) works:
32678:  type  TMimeCharset = (
32679:          CS_DEFAULT1,
32680:          CS_NOTMAPPED,
32681:          UTF_8,
32682:          WIN_1250,
32683:          WIN_1254,
32684:          WIN_1255,
32685:          WIN_1257,
32686:          WIN_1258,
32687:          ISO_8859_1,
32688:          ISO_8859_2)
32689:
32690:     Cs: TMimeCharSet;
32691:     CS:= CS_NOTMAPPED;
32692:     I := Ord(CS);
32693:     CharsetInfos[I].MimeCharset    := CS_NOTMAPPED;
32694:     //CharsetInfos[I].CodePage     := ERR_CP_NOTMAPPED;
32695:     CharsetInfos[I].MimeName       := ('');
32696:     CharsetInfos[I].FriendlyName   := 'sUnicodeUTF844';
32697:
32698:   EnvironmentString:
32699:   envlist:= Tstringlist.create;
32700:   StrtoList(GetEnvironmentString, envlist, CR)
32701:      writeln(envlist.Text);
32702:      writeln('envlist.count: '+itoa(envlist.count));
32703:   envlist.Free;
32704:
```

```
32705: Note arrays in PascalScript cannot be initialized statically. For instance, the following code raises an
       error:
32706: var I : array[0..1] of Integer = (3,4) // <-- Error!!!
32707:
32708: To solve problem, initialize array elements in your code. There are several ways to do this:
32709:     Initializing each element individually:
32710:     DelphiScript
32711:     var a : array [0..3] of string;
32712:     begin
32713:       a[0] := 'Element 0';
32714:       a[1] := 'Element 1';
32715:       a[2] := 'Element 2';
32716:       a[3] := 'Element 3';
32717:       ...
32718:     end;
32719:
32720: Initializing elements in a loop:
32721: DelphiScript
32722: var a : array [0..3] of string;
32723: begin
32724:   for i := 0 to 3 do
32725:     a[i] := 'Element ' + IntToStr(i);
32726:   ...
32727: end;
32728:
32729: var array with 2 dimensions:
32730: Var frMain     : TfrMain;
32731:     iXPos      : array [1..3] of array [ 1..3] of Integer;
32732:     iOPos      : array [1..3] of array [ 1..3] of Integer;
32733:
32734: Constructor with override do most have a second one:
32735:
32736: var Images : TObjectList;
32737:     //Create TObject list with AOwnsObjects set to True means that destroying
32738:     //the object list will also destroy all of the objects it contains
32739:     //NOTE: On ARC compiler destroying TObjectList will only remove the reference
32740:     //to the objects and they will be destroyed only if thir reference count
32741:     //drops to 0
32742:
32743:     Images := TObjectList.Create;
32744:     Images := TObjectList.Create1(true);   //override
32745:     a1 := TStatArray.Create1(0); // Growing array.
32746:
32747: Return Values
32748: Use ArcTan3 instead of ArcTan
32749: Func ArcTan3(const X : Extended) : Extended');  //bugfix of ArcTan()
32750:
32751: To return a value from a function, use Result instead of Func name. Using Func name will cause an error:
32752:
32753: Func Calc(x, y) : Integer;
32754: begin
32755:   Calc := (x - y) * (x + y); // Error !!!
32756:   Result := (x - y) * (x + y); // Correct variant
32757: end;
32758:
32759: Proc TStringsExchange(Index1, Index2: Integer; stlist: TStringlist);
32760: var
32761:   TempObject: TObject;
32762:   TempString:Str;
32763: begin
32764:   stlist.BeginUpdate;
32765:   try
32766:     TempString := stlist.Strings[Index1];
32767:     TempObject := stlist.Objects[Index1];
32768:     stlist.Strings[Index1] := stlist.Strings[Index2];
32769:     stlist.Objects[Index1] := stlist.Objects[Index2];
32770:     stlist.Strings[Index2] := TempString;
32771:     stlist.Objects[Index2] := TempObject;
32772:   finally
32773:     stlist.EndUpdate;
32774:   end;
32775: end;    //*)
32776:
32777: Integer Overflow of a DLL import:
32778: https://forum.lazarus.freepascal.org/index.php?topic=26116.0
32779: usually you set stdcall convention for a dll
32780: Func  csoundgetversion: Longint;
32781:      external 'csoundGetVersion@csound64.dll stdcall';
32782:
32783: but for procedures or void try with cdecl:
32784:   Proc  csoundSleep(duration: word);
32785:      external 'csoundSleep@csound64.dll cdecl';  //no stdcall (int overflow)
32786:
32787: [Error]..\maXbox4\examples\558_hirestimer2test2.txt(12:1): program is not allowed at this position cause:
       strange typecasts like: TStream(Nil) or TString(null)
32788:
32789: set a DLL Template: dll+Ctrl J
32790:
32791: //https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-internetsetoptiona
```

```
32792:
32793: Func  InternetSetOption(hinternet:HINTERNET; dwoption: DWord;
32794:                              buffer: Ansistr; bufflen: DWord):Bool;
32795:     external 'InternetSetOptionA@wininet.dll stdcall';
32796:
32797:  { BOOLAPI InternetSetOptionA(
32798:     HINTERNET hInternet,
32799:     DWORD     dwOption,
32800:     LPVOID    lpBuffer,
32801:     DWORD     dwBufferLength
32802:    );  }
32803:
32804: Proc testReverse_Proxy;
32805:  var  ConnectionHandle: HINTERNET;
32806:   UserName: Ansistr;
32807:   res:Bool;
32808:   EmptyParam: OleVariant;
32809: begin
32810:   // Init ConnectionHandle here
32811:   ... res:= InternetSetOption(ConnectionHandle,
32812:                            INTERNET_OPTION_PROXY_USERNAME,
32813:                            UserName, Length(UserName) + 1);
32814:
32815: Proc Str() doesnt return a value : use intToStr()
32816:
32817: TBytes = array of byte : call of the library problem
32818: Proc SaveBytesToFile(const Data: TBytes; const FileName:Str);  //const OK!
32819: Func ShiftByteArray(var A: TBytes): Byte;      // var Not OK!
32820: in script
32821: Func ByteToString(var Value: TBytes):Str;  // var and const OK!
32822: var I: integer;
32823:     S :Str;
32824:     Letra: char;
32825: begin
32826:     S := '';
32827:     for I := Length(Value)-1 Downto 0 do begin
32828:         letra := Chr(Value[I] + 48);
32829:         S := letra + S;
32830:     end;
32831:     Result := S;
32832: end;
32833:
32834: ------------------------------ Func wrapper --------------------------
32835: Its better to call a Func wrapper than the Func itself:
32836:
32837: Proc QuickSortX(StringList : TStringList; L, R: Integer);
32838: Var
32839: I, J, P : Integer;
32840: Begin
32841:   P := StringList.Count;
32842:   If (L >= P) or (R >= P) Then Exit;
32843: {……................................}
32844:
32845: Proc SortList(StringList : TStringList); //Func wrapper hides paras
32846: Begin
32847:   QuickSortX(StringList, 0, StringList.Count - 1)
32848: End;
32849: ------------------
32850: tst:= TStringlist.create;
32851: tst.add('7')
32852: tst.add('9')
32853: writeln(objtostr(tst))
32854: SortList(tst) //call to Func wrapper
32855:
32856: Hashmap or Dictionary or stringhash or map
32857:
32858: with TStringHashMap.create(CaseSensitiveTraits, 50) do begin
32859:       //Add( const s :Str; const p);
32860:      data['this is']:= '345'
32861:      data['this is2']:= '3456'
32862:      data['this is25']:= '3456677'
32863:      writeln(botostr(Has('this is2')))
32864:      setlength(sr,120)
32865:      stri:= pchar(data['this is'][10])
32866:      writeln(itoa(length(stri)))
32867:      writeln(stri)
32868:      writeln(itoa(Count));
32869:      writeln(itoa(HashSize))
32870:      clear;
32871:      free;
32872:     end;
32873:
32874: Proc stringHashtest;
32875:  var hash : TStringHash2;
32876:   //Here's unit with those TIntegerHash2 and TStringHash2, TObjectHash2 described @
32877:  basehash: THash2;
32878:  begin
32879:    hash:= TStringHash2.Create;
32880:    try
```

```
32881:      hash['one']:= 'viens value';
32882:      hash['two']:= 'divi value';
32883:      //ShowMessage(hash['one']);
32884:      //ShowMessage(hash['two']);
32885:      if hash.exists('one') then
32886:        writeln(hash['one']);
32887:      writeln(hash['two']);
32888:      hash.Rename('one', 'onerename');
32889:      writeln(hash['onerename']);
32890:    finally
32891:      hash.Free;
32892:    end;
32893:  end;
32894:
32895:  // word frequency snippet
32896:  for it:= 1 to wordlist.count-1 do begin
32897:    //writeln(itoa(it)+' '+Utf8ToAnsi(wordlist[it]));
32898:    aword:= lowercase(Utf8ToAnsi(wordlist[it]))
32899:    regex.RegEx:= '\w+';
32900:    regex.Subject:= aword;
32901:    if regex.match then aword:= regex.groups[0];  //word filter
32902:    if aword <> '' then
32903:    if dict.exists(aword) then
32904:      dict[aword]:= inttostr(strtoint(dict[aword])+1) else
32905:         dict[aword]:= '1';
32906:  end;
32907:   // iterate through all values
32908:   dict.Restart
32909:   for it:= 1 to (dict.itemcount-1) do
32910:     if dict.next then
32911:       writeln(dict.currentkey+':  '+dict.items[dict.currentkey]) ;  //}
32912:
32913: for each method - for in metheod
32914: theres no for each operator but we can map it:
32915:
32916: //for c := 'a' to 'z' do
32917:    for c:= ord('a') to ord('z') do
32918:
32919: //for ch in adata do begin
32920:   for chi:= low(adata) to high(adata) do begin
32921:      ch:= adata[chi]
32922:
32923:   c: AnsiChar; Dec: Ansistr
32924:   //for c in Dec do
32925:   for it:= 1 to length(dec) do begin
32926:     c:= dec[it]
32927:
32928:   for in and set or include charset
32929:   var
32930:     include: charset; ex:str;
32931:   //if ex[i] in ['a'..'z'] then include:=include+[ex[i]]; -type mismatch
32932:   alternatives:
32933:   1. if (ex[i] > 'a') and (ex[i] < 'z') then //in ['a'..'z']
32934:   2. if ex[i] in StrToCharSet(lowercase(LETTERSET))
32935:                then include:= include+strtoCharset(ex[i]);
32936:
32937:   use strtocharset and chartostrset as helper Func with sets
32938:   stest[2]:= '^';
32939:     if stest[2] in
32940:        StrToCharSet(lowercase(LETTERSET)+'_'+'?'+' '+'-'+'*'+'/'+'ö'+'^') then
32941:               writeln('inside') else writeln('outside');
32942:   >>> inside
32943:   writeln(charsettostr(strtocharset('*'+'?')))
32944:
32945:   function IsSTRLegal2(c: string): boolean;
32946:   begin
32947:     //result:= (Length(c)=1) and (c[1] in StrToCharSet('+'+ '-'+ '.'+ DIGISET));
32948:     result:= (Length(c)=1) and (c[1] in StrToCharSet('+-.,:'+DIGISET));
32949:   end;
32950:
32951: Func IsInvalid(s:Str):Bool;
32952: var c: char;
32953:   //leters: set of char;
32954:   leters: Charset;
32955:   firstE:Bool;
32956: begin
32957:   //writeln(itoa(IndexStr(s,['k'])));
32958:   //Result := (Length(s) < 3) or (s.IndexOf('k') = -1) or (s.Length > 9);
32959:   Result:= (Length(s) < 3) or (pos('k',s) = 0) or (Length(s) > 9);
32960:   if not Result then begin
32961:     leters:= ['d', 'e', 'g', 'k', 'l', 'n', 'o', 'w'];
32962:     firstE:= true;
32963:     //for c in s do
32964:     for it:= 1 to length(s) do begin
32965:       c:= s[it];
32966:       if c in leters then
32967:         if (c = 'e') and (firstE) then
32968:           firstE:= false
32969:         else
```

```
32970:          //Exclude(leters, AnsiChar(c))
32971:          leters:= leters-strtoChars(c)
32972:      else begin
32973:        result:= true
32974:        exit;
32975:      end;
32976:    end;
32977:  end;
32978: end;
32979:
32980:   Func TrimRight7(Str:Str):Str;
32981:    var i: Integer;
32982:    begin
32983:        i:= Length(Str);
32984:        //while (i > 0) and (Str[i] in [' ', #9]) do
32985:        while (i > 0) and (Str[i] in strtocharset(' '+#9)) do
32986:            i:= i - 1;
32987:        Result:= Copy(Str,1,i);
32988:    end;
32989:
32990:   if MatchesMask('abc.xyz', 'a*.*',true) then
32991:     writeln('File name is OK.')
32992:   else
32993:     writeln('Invalid file name.');
32994:
32995:   Find all files of a defined date:
32996:   8 C:\>dir /s /o N *.dcu | find "16/06/2020" > C:\maXbox\maxboxunitalldisk_sort.txt
32997:
32998:  Operator Overloading Surrogate:
32999:  type complexreal = record
33000:                re : real;
33001:                im : real;
33002:             end;
33003:   { TFR }
33004:   { frequency response }
33005:   TFR = record
33006:     M, phi: extended; { magnitude and phase }
33007:     F: complexreal;   { complex FR (F = M(omega) * exp(i * phi(omega)) }
33008:   end;
33009:
33010:  Func cexp (z : complexreal) : complexreal;
33011:     { exponantial : r := exp(z) }
33012:     { exp(x + iy) = exp(x).exp(iy) = exp(x).[cos(y) + i sin(y)] }
33013:     var expz : real;
33014:     begin
33015:        expz := exp(z.re);
33016:        cexp.re := expz * cos(z.im);
33017:        cexp.im := expz * sin(z.im);
33018:     end;
33019:  Func complexmulu(z1 : complexreal; r : real): complexreal;
33020:     { multiplication : z := z1 * r }
33021:     begin
33022:        result.re := z1.re * r;
33023:        result.im := z1.im * r;
33024:     end;
33025:   FFr.F:= FFr.M * cexp(i * FFr.phi); { M and phi encoded in polar coordinates }
33026:   //complextemp:= complexmulu(i,FFr.phi);
33027:   FFr.F:= complexmulu(cexp(complexmulu(i,FFr.phi)), FFr.M); { M and phi encoded in polar }
33028:
33029: replace move with iteration to move()
33030:
33031: // big note to move the move to an iteration
33032:    For i:= BListLenght downto 1 do BList[i]:= BList[i-1]; //or
33033:
33034:    move(Blist[0],Blist[1],BListLenght*SizeOf(Blist[0]));
33035:
33036:  Array of const
33037:  you cant set an array like :
33038:  {CONST} VAR t : TA;
33039: (* =
33040:                 ((0,3,1,7,5,9,8,6,4,2),
33041:                  (7,0,9,2,1,5,4,8,6,3),
33042:                  (4,2,0,6,8,7,1,3,5,9),
33043:                  (1,7,5,0,9,8,3,4,2,6),
33044:                  (6,1,2,3,0,4,5,9,7,8),
33045:                  (3,6,7,4,2,0,9,5,8,1),
33046:                  (5,8,6,9,7,2,0,1,3,4),
33047:                  (8,9,4,5,3,6,2,0,1,7),
33048:                  (9,4,3,8,6,1,7,2,0,5),
33049:                  (2,5,8,1,4,3,6,7,9,0));  *)
33050:
33051:   but you can convert:
33052:
33053:   const arrstr=  '0,3,1,7,5,9,8,6,4,2'+
33054:                  '7,0,9,2,1,5,4,8,6,3'+
33055:                  '4,2,0,6,8,7,1,3,5,9'+
33056:                  '1,7,5,0,9,8,3,4,2,6'+
33057:                  '6,1,2,3,0,4,5,9,7,8'+
33058:                  '3,6,7,4,2,0,9,5,8,1'+
```

```
33059:                  '5,8,6,9,7,2,0,1,3,4'+
33060:                  '8,9,4,5,3,6,2,0,1,7'+
33061:                  '9,4,3,8,6,1,7,2,0,5'+
33062:                  '2,5,8,1,4,3,6,7,9,0';
33063:
33064: procedure InitTable4;
33065: var i,j,k: integer; repl: string;
33066: begin
33067:  repl:= StringReplace(arrstr,',','',[rfReplaceAll]);
33068:  k:= 0
33069:  for i:= 0 to 9 do
33070:    for j:= 0 to 9 do begin
33071:        inc(k)
33072:        t[i][j]:= strtoint(repl[k])
33073:    end;
33074: end;
33075:
33076: Convert Static Array to Dynamic Array
33077:
33078: procedure TForm1Button1Click(Sender: TObject);
33079: var
33080:   ADate: TDateTime;
33081:   days: array[1..7] of string;
33082: begin
33083:   days[1]:= 'Sunday';
33084:   days[2]:= 'Monday';
33085:   days[3]:= 'Tuesday';
33086:   days[4]:= 'Wednesday';
33087:   days[5]:= 'Thursday';
33088:   days[6]:= 'Friday';
33089:   days[7]:= 'Saturday';
33090:   ADate := StrToDate('31/01/1981');
33091:   ShowMessage('31/01/1981' + ' was a ' + days[{SysUtils.}DayOfWeek(ADate)]);
33092: end;
33093:
33094: //Incompatible types: 'Dynamic array' and 'array of string' to open array
33095:
33096: procedure TForm1Button1ClickDynArray(Sender: TObject);
33097: var aDate: TDateTime;
33098:     days: array of string;
33099: begin
33100:   setLength(days,7);
33101:   days:=['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'];
33102:   aDate:= StrToDate('31/01/1981');
33103:   ShowMessage('31/01/1981'+' was a '+ days[{SysUtils.}DayOfWeek(aDate)-1]);
33104: end;
33105: // with SplitStr
33106: begin
33107:   aDate:= StrToDate('31/01/1981');
33108:   days:= SplitStr('Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday',',');
33109:   writeln('31/01/1981'+' was a '+ days[{SysUtils.}DayOfWeek(aDate)-1]);
33110: end;
33111:
33112:  AssignFile versus TStringlist
33113:  ---------------------------------
33114:  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.
33115:  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
33116:
33117:     filelist:= TStringlist.create;
33118:     // assignfile(f,opendialog1.filename);
33119:     filelist.loadfromfile( filename)
33120:     // reset(f);
33121:     // while not eof(f) do
33122:     for it:= 0 to filelist.count-1 do begin
33123:       //readln(f,line);
33124:       line:= filelist.strings[it];
33125:       writeln(line)
33126:       if (length(line)>2) and (copy(line,1,2)='ID') then {skip Id records}
33127:       else begin {it's a data record}
33128:         if totcount=0 then begin
33129:           savedecimalseparator:=decimalseparator;
33130:           if pos(',',line)>0 then setdecimalseparator(',')
33131:           else setdecimalseparator('.');
33132:         end;
33133:          errcode:=0;
33134:         try
33135:           x:=strtofloat(trim(line));
33136:           except errcode:=1;
33137:         end;
33138:         if (errcode=0) and (x>0)and(x<=amaxtime)
33139:         then begin
33140:          inc(samps[round(maxindex*x)]); {increment the count for the bar for x}
33141:          inc(totcount);
33142:         end;
33143:       end;
33144:     end;
```

```
33145:     // closefile(f);
33146:     filelist.Free;
33147:
33148: Random Generator:
33149: https://github.com/TurboPack/SysTools/blob/master/source/StRandom.pas
33150:
33151:  strand:= TStRandomSystem.create(42); //if 0 then randomseed, else randomstate
33152:   sleep(200)
33153:   writeln(floattostr(strand.asFloat))
33154:   writeln(itoa(strand.asInt(10000)))
33155:   writeln(itoa(strand.asInt($E15)))
33156:   UniformityTest2(strand, ChiSquare, DegsFreedom)
33157:   writeln('ChiSquare: '+floattostr(ChiSquare)+' ,degsfreedom:'+itoa(degsfreedom));
33158:   UniformityTest3(strand, ChiSquare, DegsFreedom)
33159:   writeln('ChiSquare: '+floattostr(ChiSquare)+' ,degsfreedom:'+itoa(degsfreedom));
33160:  strand.Free;
33161:
33162: Https Post API: http://www.softwareschule.ch/download/maxbox_starter94.pdf
33163: translator for different countries:
33164: var httpReq,hr: Olevariant;
33165: try
33166:   hr:= httpReq.Open('POST','https://libretranslate.pussthecat.org/translate', false);
33167:   httpReq.setRequestheader('user-agent',
33168: 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0');
33169:   httpReq.setRequestheader('content-type','application/x-www-form-urlencoded');
33170: //httpReq.setRequestheader('X-RapidAPI-Host','nlp-translation.p.rapidapi.com');
33171:   if hr= S_OK then HttpReq.Send('q='+HTTPEncode(feedstream)+
33172:                                 '&source=es'+'&target=en');
33173:   If HttpReq.Status = 200 Then
33174:        result:= HttpReq.responseText
33175:     Else result:= 'Failed at getting response:'+itoa(HttpReq.Status)+HttpReq.responseText;
33176:      //writeln('debug response '+HttpReq.GetAllResponseHeaders);
33177:   finally
33178:     httpreq:= unassigned;
33179:
33180: File Size Reference:  filegetsize()
33181: Func FileGetSizeX(const FileName:Str): Int64;
33182: {$ifndef fpc}
33183: var
33184:   FileInfo: TSearchRec;
33185: {$endif}
33186: begin
33187: {$ifdef fpc}
33188:   Result:= FileUtil.FileSize(FileName);
33189: {$else}
33190:   // from LCL FileUtil code
33191:   FileInfo.Name:= Filename;
33192:   FileInfo.FindHandle:= {Windows.}FindFirstFile({Windows.}{LPTSTR}(FileInfo.Name)
33193:                                                 ,FileInfo.FindData);
33194:   if FileInfo.FindHandle={Windows.}Invalid_Handle_value then begin
33195:     Result:=-1;
33196:     Exit;
33197:   end;
33198:   Result:= (int64(FileInfo.FindData.nFileSizeHigh)
33199:                     Shl 32)+ FileInfo.FindData.nFileSizeLow;
33200:   {Windows.}FindCloseW(FileInfo.FindHandle);
33201: {$endif}
33202: end;
33203: { this func is a wrapper for platform-specific file fns
33204:   IE a way to get rid of those portability warnings
33205:   and a place to put the equivalent linux fns }
33206:
33207: RTL OSLibrary+
33208: Func MySoundcard: Longint; external 'waveOutGetNumDevs@winmm.dll stdcall';
33209: Func isSound:Bool; begin result:= mySoundcard > 0 end;
33210: Func StringtoHex(Data:Str):Str;
33211: Func GetAnsistrRefcount(const S:Str):Card;
33212: Func WideStringToString(const ws: WideString; codePage: Word): Ansistr;
33213: Func StringToWideString(const s: Ansistr; codePage: Word): WideString;
33214: Proc FreeObjectList(List: TObjectList);
33215: Func SecondToTime(const Seconds:Card): Double;
33216: Func CopyDir2(const fromDir, toDir:Str):Bool;;
33217: Func MoveDir(const fromDir, toDir:Str):Bool;;
33218: Func DelDir(dir:Str):Bool;;
33219: Proc DeleteScansRect(Src, Dest: TBitmap; rs, rd: TRect);
33220: Proc FadeIn(ImageFileName: TFileName; aForm1: TForm);
33221: Proc FadeOut(ImageFileName: TFileName);
33222: Proc FadeOut32(const Bmp: TImage; Pause: Int);
33223: Func CheckBDEInstalled:Bool;   //IsBDE
33224: Proc SaveStringToStream( const Str :Str; Stream : TStream);
33225: Func LoadStringFromStream( Stream : TStream) :Str;
33226: Func WaitForSyncObject(SyncObject:THandle;Timeout:Card;BlockInput:Bool):Card;
33227: Func ProcessMessage :Bool;
33228: Proc ProcessMessages( Timeout : DWORD);  //without application!
33229: Func GetNumberOfEventLogRecords(hEventLog:THandle;var NumberOfRecords:DWORD): BOOL;
33230: Func GetOldestEventLogRecord(hEventLog:THandle; var OldestRecord : DWORD): BOOL;
33231: ex.:EventLog:=RegisterEventSource('0',('maxbox4.exe'));if GetNumberOfEventLogRecords(eventlog,recs)
33232: Proc InitializeLocaleSupport); Proc TerminateLocaleSupport;
33233: Func StrReplaceRegEx(const Subject,Pattern:Ansistr; Args:array of const : Ansistr;
```

```
33234: Func TickCountToDateTime( Ticks :Card) : TDateTime;
33235: Proc OutputDebugStringmax( const S :Str);
33236: Proc OutputDebugFormat( const FmtStr :Str; Args : array of const);
33237: Func IsAppRunningInDelphi :Bool;
33238: Func RunExecutable( const AFileName :Str; AWaitForIt :Bool) : DWORD;
33239: Func SystemCodePage : Int;
33240: Func CPUSpd:Str;
33241: Func CPUSpeed:Str;
33242: Func BigFib(n: Int):Str;  //BigFibo
33243: Func BigFac(n: Int):Str;  //BigFact
33244: Func UpTime:Str;
33245: Func NetLogon(const Server,User,Password:WideString; out ErrorMessage:str) :Bool;
33246: Func NetLogoff( const Server, User, Password : WideString) :Bool;
33247: Proc ErrorNamedPipe( const Message :Str);
33248: Proc BroadcastChange; //method that broadcasts the necessary message WM_SETTINGCHANGE
33249: Func WriteFile2(hFile:THandle;const Buffer,nNumberOfBytesToWrite:DWORD; var lpNumberOfBytesWritten:DWORD;
       lpOverlapped : Tobject):BOOL;
33250: Func ReadFile2(hFile:THandle;var Buffer,nNumberOfBytesToRead:DWORD;var
       lpNumberOfBytesRead:DWORD;lpOverlapped : Tobject):BOOL;
33251: // Security helper functions
33252: Proc InitializeSecurity(var SA: TSecurityAttributes);
33253: Proc FinalizeSecurity(var SA: TSecurityAttributes);
33254: // Pipe helper functions
33255: Proc CheckPipeName(Value:Str);
33256: Proc ClearOverlapped(var Overlapped: TOverlapped; ClearEvent:Bool);
33257: Proc CloseHandleClear(var Handle: THandle);
33258: Func ComputerName2:Str;
33259: Proc DisconnectAndClose(Pipe: HPIPE; IsServer:Bool = True);
33260: Func EnumConsoleWindows(Window: HWND; lParam: Int): BOOL; stdcall;
33261: Proc FlushMessages;
33262: Func IsHandle(Handle: THandle):Bool;
33263: Proc RaiseWindowsError;
33264: Func UpTime:Str; //OS uptime
33265: Func CharSetToCP(ACharSet: TFontCharSet): Int;
33266: Func CPToCharSet(ACodePage: Int): TFontCharSet;
33267: Func TwipsToPoints(AValue: Int): Int;
33268: Func PointsToTwips(AValue: Int): Int;
33269: Proc LoadGraphicFromResource(Graphic:TGraphic; const ResName:str; ResType:PChar);
33270: Func SplitStr(sInput:str; Delimiter:str): TStringArray;
33271: Func GetDataFromFile2(sFileName: Ansistr): Ansistr;
33272: Func ExtractFileNameWithoutExt(const FileName:Str):Str;
33273: Func SubstringCount (Substring :Str; Str :Str): Int;
33274: Func loadForm(vx, vy: smallint): TForm;   //alias getForm()
33275: Func getForm(vx, vy: smallint): TForm;   //alias getForm()
33276: Func getForm2(vx,vy:smallint;acolor:TColor;aname:str):TForm; //sizeable!
33277: Proc paintProcessingstar2(ppform: TForm);
33278: Func ContentTypeGetExtn(Const Content:Str; var CLSID:Str):Str;
33279: Func ContentTypeFromExtn(Const Extension:Str):Str;
33280: Func DateTimeDiff2(Start, Stop : TDateTime) : int64;
33281: Func checkSystem:Str;;
33282: Func getSystemReport:Str;;
33283: Func SystemCheck:Str;;
33284: Func SetThreadDesktop( hDesktop : HDESK) : BOOL;
33285: Func RSAEncrypt(aval, apow, amod:Str):Str;;
33286: Func RSADecrypt(aval, apow, amod:Str):Str;;
33287: Func EncryptRSA(aval, apow, amod:Str):Str;;
33288: Func DecryptRSA(aval, apow, amod:Str):Str;;
33289: Func SwitchToThread : BOOL;
33290: Func CloseDesktop( hDesktop : HDESK) : BOOL;
33291: Func GetThreadDesktop( dwThreadId : DWORD) : HDESK;
33292: Func SetSyscallHook():Bool;;
33293: Func SetSwapcontextHook():Bool;;
33294: Func UnhookAll():Bool;;
33295: Func getWorld:Str;;
33296: Func getIPConfigAll:Str;;
33297: Func getIPConfig:Str;;
33298: Func WinsockEnabled:Bool;
33299: Func HTTPEncode2(const AStr:Str):Str;
33300: RSAEncryptStr(const EncryptionType:TRSAEncryptionType;const PublicKey:TRSAPublicKey;const Plain:Ansistr):
       Ansistr;
33301:
33302: ////////////////////////////////////////////////////////////////////////////
33303: // Object instance functions
33304: ////////////////////////////////////////////////////////////////////////////
33305: Func AllocateHWnd(Method: TWndMethod): HWND;
33306: Proc DeallocateHWnd(Wnd: HWND);
33307: Func DeleteDC( DC : HDC) : BOOL;
33308: Func DeleteMetaFile( p1 : HMETAFILE) : BOOL;
33309: Func DeleteObject( p1 : HGDIOBJ) : BOOL;
33310: Func FormInstanceCount2(AFormClass: TFormClass): Int;;
33311: Func FormInstanceCount(const AFormClassName:Str): Int;;
33312:  ClassN(CL.FindClass('Class of TForm'),'TFormClass;
33313: Func ObjToStr( const O : TObject) :Str
33314: MessageBox(0,PChar('CPU speedis '+CPUSpd+' MHz'),'CPU Speed Check',MB_IconInformation+MB_OK);
33315: TKLogEvent', 'Proc (Sender : TObject; Code : TKLogType; const Text :Str);
33316: AddRegisteredVariable('FormatSettings','TFormatSettings;  //at sysutils!
33317: AddRegisteredVariable('mouse','TMouse;  //at controls HPIPE', 'THandle;
33318:
33319: Func ListIdentical2(l1,l2:TStringList):Bool;
```

```
33320: begin  Result:= False;
33321:   if l1.count = l2.count then begin
33322:     for it:= 0 to l1.count-1 do
33323:       if (l1[it] <> l2[it]) then Exit;
33324:     Result:= True;
33325:   end;
33326: end;
33327:
33328: // Converts String To Hexadecimal Maybe usefull for a hex-editor
33329: // For example: Input = 'ABCD' Output = '41 42 43 44'
33330: Func StringtoHex(Data:Str):Str;
33331: var i, i2: Int; s:Str;
33332: begin
33333:   i2:= 1;
33334:   for i:= 1 to Length(Data) do begin
33335:     Inc(i2);
33336:     if i2 = 2 then
33337:       s:= s + ' ';  i2 := 1;
33338:       s:= s + IntToHex(Ord(Data[i]), 2);
33339:   end;
33340:   Result:= s;
33341: end;
33342:
33343: Func XRTLIsInMainThread:Bool;
33344: begin
33345:   Result:= GetCurrentThreadID = MainThreadID;
33346: end;
33347:
33348: computers:= TStringList.Create;
33349:   FindComputers(computers)
33350:   for it:= 0 to computers.count-1 do
33351:           writeln(computers.strings[it]); computers.free;
33352:
33353:   s:= (Sender AS TStringGrid).Cells[col,row];
33354:   Sender AS TStringGrid).Canvas.Font.Color := clBlack;
33355:   twidth:=  (Sender AS TStringGrid).Canvas.TextWidth('X;
33356:   (Sender AS TStringGrid).Canvas.TextRect(Rect, Rect.Left + twidth div 2,
33357:                 YCenter(Rect, (Sender AS TStringGrid).Canvas,s),s)
33358:
33359: Proc | EXceptionsOnmaXbox;  Exception Handling EXceptions On maXbox
33360: var filename,emsg:str;
33361: begin
33362:  filename:= '';
33363:  try
33364:    if filename = '' then
33365:      RaiseException(erCustomError, 'Exception: File name cannot be blank;
33366:  except
33367:    emsg:= ExceptionToString(ExceptionType, ExceptionParam);
33368:      //do act with the exception message i.e. email it or save to a log etc
33369:    writeln(emsg)
33370:  end;
33371: end;
33372:
33373: XRaise raise an exception without except block:
33374: if (DLLHandle < HINSTANCE_ERROR) then
33375:    Xraise(Exception.Create(DLLName + ' library can not be loaded or not found.'+
33376:                              SysErrorMessage(GetLastError)));
33377:
33378: Proc CallTicker(msecs: Int);
33379: begin
33380:  with TStopWatch.create do begin
33381:    start
33382:      | //Delay2StopWatch(msecs)
33383:    stop
33384:    writeln('Stop Watch CPU Tester: '+getValueStr)
33385:  end;
33386: end;
33387:
33388: MessageDlg Example:
33389: mr:=messagedlg('Save clock first?',mtconfirmation,[mbYes,mbNo,mbCancel],0);
33390:     if mr=mryes then writeln('yes')
33391:
33392: Func getFilefrom
33393: if not (Fileexists('1004unixdict.txt') or fileexists('unixdict.txt')) then
33394:  if IsInternet then begin
33395:    writeln(dictURL)
33396:    wGetX2(dictUrl,Exepath+'1004unixdict.txt');
33397:    openFile(Exepath+'1004unixdict.txt')
33398:  end; //}
33399:
33400: Func IsASCIIDigit(const Ch: Char):Bool;
33401: begin //Result := Ord(Ch) in [Ord('0')..Ord('9')];
33402:   Result:= (Ord(Ch) >= Ord('0')) And (ord(ch) <= Ord('9'));
33403: end;
33404:
33405: Func RemainingBatteryPercent: Int;
33406: var Stat: {Windows.}TSystemPowerStatus;
33407: begin
33408:   {Windows.}GetSystemPowerStatus(Stat);
```

```
33409:   Result:= Stat.BatteryLifePercent;
33410:   if (Result < 0) Or (Result > 100) then
33411:     Result:= -1;
33412: end;
33413: // ************************************************************
33414: // TIdBaseComponent is the base class for all Indy components.
33415: // ************************************************************
33416: type
33417:   TIdBaseComponent = class(TComponent)
33418:   public
33419:     Func GetVersion:Str;
33420:     property Version:Str read GetVersion;
33421:   published
33422:   end;
33423:
33424:  if IsInternet then
33425:    wGetX('http://www.softwareschule.ch/download/maxbox_starter4.pdf','mywgettestpdf.pdf;
33426:     RegEx tester sr := '123.456.789-00$p'
33427:     writeln(ReplaceRegExpr('\D',sr,'',true))
33428:     >>> 12345678900  //\D is a non-digit, and \W is a non-word character, both should work)
33429:     ref: lambda: a+b --> a.+(b) in preparations
33430:     namedpipe1:= namedpipe as TNamedpipe;
33431:     printF('this is %.18f ',[maxCalc('ln(2)^e')]); this is 0.369248502937272178
33432:     printF('this is Area of r=1  %.18f ',[maxCalc('PI*(1^2)')]);
33433:     printF('this is Area of d=2r %.18f ',[maxCalc('PI/4*(2^2)')]);
33434:     println(ReplaceRegExpr('\d\d{4,12}[}],',
33435:              GetSentimentStream7(URLSentimentAPI2,Items[it].title),' ',false))
33436:     {"probability": {"neg": 0.5180  "neutral": 0.5201  "pos": 0.4819 , "label": "neutral"}
33437:  begin
33438:   initCRCarray2;
33439:   aStringStream := TStringStream.Create(filetoString(exepath+testFile));
33440:   try
33441:    //aStringStream.loadfromfile
33442:     writeln('CRC32_: '+IntToHex(CalculateCRC32X(aStringStream), 8));
33443:   finally
33444:     aStringStream.Free;
33445:   end;
33446:  end;
33447:
33448: Proc TForm1Findwindow_Remote_Notepad;
33449: var wnd: HWND; i: Int; s:Str;
33450: begin
33451:   wnd := FindWindow('notepad', ';
33452:   if wnd <> 0 then begin
33453:     wnd := FindWindowEx(wnd, 0, 'Edit', ';
33454:     // Write Text in Notepad.
33455:     s := 'Hello maXPad 3 ecotronics'+#13#10+'second line:';
33456:     for i := 1 to Length(s) do
33457:       SendMessage(wnd, WM_CHAR, Ord(s[i]), 0);
33458:     // Simulate Return Key.
33459:     PostMessage(wnd, WM_KEYDOWN, VK_RETURN, 0);
33460:     // Simulate Space.
33461:     PostMessage(wnd, WM_KEYDOWN, VK_SPACE, 0);
33462:     SendMessage(wnd, WM_CHAR, 5, 0);
33463:     //send time of F5 command in notepad!
33464:     PostMessage(wnd, WM_KEYDOWN, VK_F5, 0);
33465:   end;
33466: end;
33467:
33468: Proc XPATHTest; var XML: OLEVariant;
33469: begin
33470:   XML:= CreateOleObject('MSXML2.DOMDocument.3.0') ;
33471:   XML.async:= false;
33472:   XML.SetProperty('SelectionLanguage','XPath');
33473:   XML:= unassigned;
33474: end;
33475:
33476: with adoQuery do begin //recordset hack
33477:   cachesize:= 500;
33478:   //commandtext:= 'INSERT INTO Table1 (FirstName, LastName, Phone)'+
33479:                     'VALUES (''MaXML'', ''Box42810'', ''031-3337788'')';
33480:   commandText:= 'SELECT * FROM Table1';
33481:   connectionString:='Provider=MSDASQL.1;Persist Security Info=False;Data Source=mX4base';
33482:   Open;
33483:   Recordset20(recordset).save(Exepath+'recordset6.xml',adPersistXML)
33484:   writeln(GetISODateString(1995,12,24))
33485:   writeln(botostr(IsValidISODateTimeString(GetISODateString(1995,12,12), true)))
33486:   writeln(MicrosoftCodePageToMIMECharset(1252))writeln(GetISOTimeString(12,45,33, true));
33487:   writeln(botostr(IsValidISOTimeString(GetISOTimeString(12,45,33, true), true)))
33488:   IDoc:= LoadXMLDocument3(BOOKSXML); //('.\books.xml');
33489:   INode:= SelectXMLNode(IDoc.getDocumentElement,'/bookstore/book[2]/title');
33490:   writeln(FormatXMLFile2String(BOOKSXML));
33491:   writeln(DataSetToXML(TDataset(adoquery)));
33492:
33493: Proc EnableDEP2;     // DEP
33494: const PROCESS_DEP_ENABLE: DWORD=$00000001;
33495: var SetProcessDEPPolicy: function(dwFlags: DWORD): BOOL; stdcall;
33496: begin  SetProcessDEPPolicy := GetProcAddress(GetModuleHandle(kernel32),'SetProcessDEPPolicy');
33497:   if Assigned(SetProcessDEPPolicy) then begin
```

```
33498:     //don't bother checking for errors since we don't need to know if it fails
33499:     SetProcessDEPPolicy(PROCESS_DEP_ENABLE);
33500:   end;
33501: end;
33502:
33503: unit uPSI_UTGraphSearch; chNGE tnode to tnode2
33504:
33505: procedure SIRegister_TNode2(CL: TPSPascalCompiler);
33506: begin
33507:   //with RegClassS(CL,'tObject', 'TNode2') do
33508:   with CL.AddClassN(CL.FindClass('tObject'),'TNode') do begin
33509:     RegisterProperty('adjacents', 'array of TEdge', iptrw);
33510:     RegisterProperty('nbradjacents', 'integer', iptrw);
33511:     RegisterProperty('index', 'integer', iptrw);
33512:     RegisterProperty('x', 'integer', iptrw);
33513:     RegisterProperty('y', 'integer', iptrw);
33514:     RegisterMethod('Constructor create');
33515:   end;
33516: end;
33517:
33518: TList.count and capacity!
33519: RegisterProperty('Q', 'TList', iptrw);
33520: type  __Pointer; as TObject cause of NIL
33521: FreeAndNilStream (Output); Output : TStream;
33522: FreeAndNilBmp (Output); Output : TBitmap;
33523: sign: About!
33524:
33525: Convert a dfm to string: var abt:Bool;
33526: srlist:= TStringlist.create;
33527: if LoadDFMFile2Strings('C:\maXbox\mX42810\backgroundworker\Demo\Primes\Main.dfm',srlist,abt)= 0 then
       writeln(srlist.text);  srlist.Free;    *) Assert Test: _outd( a1.StandardDeviation );
33528:    Assert(trunc(a1.StandardDeviation)=46, 'must = 46 ');
33529:  isloc:= WMIStart;  iserv:= WMIConnect(isloc,'localhost','','');
33530:  wmi_dom:= 'Win32_NetworkAdapterConfiguration';
33531:
33532: V 4.7.5.90 VI  +  4.7.6.10
33533: Amount of Functions: 20549 Amount of Procedures: 12307
33534: Amount of Constructors: 1937 Amount of Destructors: 14
33535: Totals of Calls: 34807 , Size: 31,974,216 bytes
33536: SHA1: of 4.7.5.90 96DCDE2028125E00B67E42A801721AC513A5EAFC CRC32: BBC3A7E5
33537: ▯▯▯ mX4 executed: 03/11/2021 15:25:47  Runtime: 0:12:33.807 Memload: 39% use
33538: ▯▯▯ 4.7.6.10 III Totals of Calls: 35218
33539: SHA1: of 4.7.6.10 6647E6200E46702944A6B8175A53EDC1ABEF8E14
33540: CRC32: 1A2922DA:  32,221,000 bytes
33541: ▯▯▯ mX4 executed: 12/12/2021 10:03:11  Runtime: 0:12:50.630  Memload: 45% use
33542: ▯▯▯ 4.7.6.10 IV Total of Function Calls: 35351
33543: SHA1: of 4.7.6.10  A9C0D35723F9D2E72E9270207640F938FF58555B
33544: CRC32: 8230EA82 31.3 MB (32,917,832 bytes)
33545: ▯▯▯ 4.7.6.10 VII
33546: Amount of Functions: 20994 Amount of Procedures: 12492
33547: Amount of Constructors: 1984  Amount of Destructors: 14
33548: Totals of Calls:35484 SHA1:of 4.7.6.10 EC20D0A371A57AA078762D6D45F7229A370549A1
33549: CRC32: 82E481A4:  33273160 bytes
33550: ▯▯▯ mX4 executed: 14/03/2022 15:21:38  Runtime: 0:13:41.253  Memload: 42% use
33551: ▯▯▯
33552: 17/03/2022  14:08      194,307 IFSI_WinForm1puzzle.dcu
33553: 17/03/2022  20:30       46,854 uPSC_classes.dcu
33554: 17/03/2022  19:48        5,226 uPSC_std.dcu
33555: 17/03/2022  18:49       53,416 uPSR_classes.dcu
33556: 17/03/2022  20:30       19,528 uPSI_SynEditHighlighter.dcu
33557: 17/03/2022  16:03       11,784 uPSI_SynEditMiscProcs.dcu
33558: 17/03/2022  21:15      173,341 uPSI_TeEngine.dcu
33559: 17/03/2022  21:16       30,854 MathsLib.dcu
33560: 17/03/2022  20:13       77,627 uPSI_neuralvolume.dcu
33561: 17/03/2022  13:10       13,188 uPSI_UcomboV2.dcu
33562: 17/03/2022  13:26       14,813 uPSI_UTGraphSearch.dcu
33563: 17/03/2022  13:22       21,041 UTGraphSearch.dcu
33564:
33565: Release Notes maXbox 4.7.6.10 VIII August 2022 mX476
33566: Amount of Functions: 21179
33567: Amount of Procedures: 12591
33568: Amount of Constructors: 1987
33569: Amount of Destructors: 14
33570: Totals of Calls: 35771
33571:
33572: Release Notes maXbox 4.7.6.20 Oct 2022 mX476
33573: Amount of Functions: 21309
33574: Amount of Procedures: 12640
33575: Amount of Constructors: 1989
33576: Amount of Destructors: 14
33577: Totals of Calls: 35952
33578: SHA1: of 4.7.6.20 C0459A5DD0CCC12CE9F3CDEC82E3E0C920D38D9C
33579: CRC32: 498B62AF:  33690392 bytes
33580: ▯▯▯ mX4 executed: 09/10/2022 20:59:18  Runtime: 0:14:21.810  Memload: 39% use
33581: ▯▯▯ maXbox4.exe now available as a NFT
33582: sha256: 09eb95c4cc3b7fe60772405b170af4705085da89d190213228c65313c8733499
33583: https://lawrencebarsanti.wordpress.com/2009/11/28/introduction-to-pascal-script/
33584:
33585: Total of Function Calls: 35952
```

```
33586: SHA1: 4.7.6.20 B2D3DB10079C63D9C4E3C3350844174E259F4FEB
33587: CRC32: 0C3FCCE9 32.2 MB (33,776,920 bytes)
33588: Compilation Timestamp 2022-10-13 18:54:15 UTC
33589: Entry Point 24770844 - Contained Sections 10
33590: ▯▯▯
33591: sha1: b2d3db10079c63d9c4e3c3350844174e259f4feb
33592: sha256: 5d229931d7b91a8e1d79ee2b1f19279a0dd2685c7c99974b0da9df749347fc58
33593:
33594: Amount of Functions: 21428
33595: Amount of Procedures: 12692
33596: Amount of Constructors: 2007
33597: Amount of Destructors: 14
33598: Totals of Calls: 36141
33599: SHA1: of 4.7.6.20 B2B44FC20DA22111EF81553B94B0B903B9D7FA7A
33600: CRC32: 1511CC43:  34014488 bytes
33601: ▯▯▯ mX4 executed: 08/12/2022 11:52:51  Runtime: 0:15:7.352  Memload: 42% use
33602: Total of Function Calls: 36141
33603: SHA1: 4.7.6.20 B2B44FC20DA22111EF81553B94B0B903B9D7FA7A
33604:                b2b44fc20da22111ef81553b94b0b903b9d7fa7a
33605: CRC32: 1511CC43 32.4 MB (34,014,488 bytes)
33606: Compilation Timestamp 2022-12-08 10:01:05 UTC Signing time 08 Dec 2022 11:12:55
33607: Entry Point 24902184 - Contained Sections 10
33608: sha1: b2b44fc20da22111ef81553b94b0b903b9d7fa7a
33609: sha256: 66e02bfc560c5082c2bfcb5e4b8e2a7b19fa1f60386474097625d55c7cc6b472
33610: https://www.hybrid-analysis.com/
33611: Suspicious Indicators
33612: General Found a potential E-Mail address in binary/memory
33613: Network Related Found potential IP address in binary/memory
33614: Remote Access Related Contains indicators of bot communication commands
33615: Contains references to WMI/WMIC Reads terminal service related keys (often RDP related)
33616: Spyware/Information Retrieval Touches files in program files directory
33617: Unusual Characteristics Input file contains API references not part of its Import Address Table (IAT)
33618: Installs hooks/patches the running process
33619: https://www.virustotal.
       com/gui/file/212f8fda437e877c7f89f771ce40e57814cfb4380ee010c090df78d8dbbc4d9f/behavior
33620: This report is generated from a file or URL submitted to this webservice on December 15th 2022 18:16:53
       (UTC)
33621: Guest System: Windows 10 64 bit, Professional, 10.0 (build 16299),
33622: Report generated by Falcon Sandbox v9.5.2 © Hybrid Analysis
33623: 1 match for rule Change PowerShell Policies to an Insecure Level by frack113 from Sigma Integrated Rule
       Set (GitHub) Detects use of executionpolicy option to set insecure policies
33624: DetectItEasy PE32  Compiler: Borland Delphi (2007) Linker: Turbo Linker (2.25*,Delphi) [GUI32,signed]
33625: 5.4% (.EXE) Win64 Executable (generic)- 5.1% (.EXE) DOS Borland compiled Executable (generic)
33626:
33627:
33628: ************************************************************
33629:  unit List asm internal end units  all unitlist unitslist allunits
33630: ************************************************************
33631: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
33632: 02 unit SIRegister_IdStrings                   //Indy Sockets
33633: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
33634: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
33635: 05 unit IFSI_WinForm1puzzle;                   //maXbox
33636: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
33637: 07 unit RegisterDateTimeLibrary_R(exec);       //Delphi
33638: 08 unit RIRegister_MathMax_Routines(exec);     //Jedi & Delphi
33639: 09 unit RIRegister_IdGlobal_Routines(exec);    //Indy Sockets
33640: 10 unit RIRegister_SysUtils_Routines(Exec);    //Delphi
33641: 11 unit uPSI_IdTCPConnection;                  //Indy some functions
33642: 12 unit uPSCompiler.pas;                       //PS kernel functions
33643: 13 unit uPSI_DBCommon;                         //DB Common_Routines and Types
33644: 14 unit uPSI_Printers.pas                      //Delphi VCL
33645: 15 unit uPSI_MPlayer.pas                       //Delphi VCL
33646: 16 unit uPSC_comobj;                           //COM Functions
33647: 17 unit uPSI_Clipbrd;                          //Delphi VCL
33648: 18 unit Filectrl in IFSI_SysUtils_max;         //VCL Runtime
33649: 19 unit uPSI_SqlExpr;                          //DBX3
33650: 20 unit uPSI_ADODB;                            //ADODB
33651: 21 unit uPSI_StrHlpr;                          //String Helper Routines
33652: 22 unit uPSI_DateUtils;                        //Expansion to DateTimeLib
33653: 23 unit uPSI_FileUtils;                        //Expansion to Sys/File Utils
33654: 24 unit JUtils / gsUtils;                      //Jedi / Metabase
33655: 25 unit JvFunctions_max;                       //Jedi Functions
33656: 26 unit HTTPParser;                            //Delphi VCL
33657: 27 unit HTTPUtil;                              //Delphi VCL
33658: 28 unit uPSI_XMLUtil;                          //Delphi VCL
33659: 29 unit uPSI_SOAPHTTPClient;                   //Delphi VCL SOAP WebService V3.5
33660: 30 unit uPSI_Contnrs;                          //Delphi RTL Container of Classes
33661: 31 unit uPSI_MaskUtils;                        //RTL Edit and Mask functions
33662: 32 unit uPSI_MyBigInt;                         //big Int class with Math
33663: 33 unit uPSI_ConvUtils;                        //Delphi VCL Conversions engine
33664: 34 unit Types_Variants;                        //Delphi\Win32\rtl\sys
33665: 35 unit uPSI_IdHashSHA1;                       //Indy Crypto Lib
33666: 36 unit uPSI_IdHashMessageDigest               //Indy Crypto;
33667: 37 unit uPSI_IdASN1Util;                       //Indy ASN1Utility Routines;
33668: 38 unit uPSI_IdLogFile;                        //Indy Logger from LogBase
33669: 39 unit uPSI_IdIcmpClient;                     //Indy Ping ICMP
33670: 40 unit uPSI_IdHashMessageDigest_max           //Indy Crypto &OpenSSL;
33671: 41 unit uPSI_FileCtrl;                         //Delphi RTL
```

```
33672: 42 unit uPSI_Outline;                          //Delphi VCL
33673: 43 unit uPSI_ScktComp;                         //Delphi RTL
33674: 44 unit uPSI_Calendar;                         //Delphi VCL
33675: 45 unit uPSI_VListView                         //VListView;
33676: 46 unit uPSI_DBGrids;                          //Delphi VCL
33677: 47 unit uPSI_DBCtrls;                          //Delphi VCL
33678: 48 unit ide_debugoutput;                       //maXbox
33679: 49 unit uPSI_ComCtrls;                         //Delphi VCL
33680: 50 unit uPSC_stdctrls+;                        //Delphi VCL
33681: 51 unit uPSI_Dialogs;                          //Delphi VCL
33682: 52 unit uPSI_StdConvs;                         //Delphi RTL
33683: 53 unit uPSI_DBClient;                         //Delphi RTL
33684: 54 unit uPSI_DBPlatform;                       //Delphi RTL
33685: 55 unit uPSI_Provider;                         //Delphi RTL
33686: 56 unit uPSI_FMTBcd;                           //Delphi RTL
33687: 57 unit uPSI_DBCGrids;                         //Delphi VCL
33688: 58 unit uPSI_CDSUtil;                          //MIDAS
33689: 59 unit uPSI_VarHlpr;                          //Delphi RTL
33690: 60 unit uPSI_ExtDlgs;                          //Delphi VCL
33691: 61 unit sdpStopwatch;                          //maXbox
33692: 62 unit uPSI_JclStatistics;                    //JCL
33693: 63 unit uPSI_JclLogic;                         //JCL
33694: 64 unit uPSI_JclMiscel;                        //JCL
33695: 65 unit uPSI_JclMath_max;                      //JCL RTL
33696: 66 unit uPSI_uTPLb_StreamUtils;                //LockBox 3
33697: 67 unit uPSI_MathUtils;                        //BCB
33698: 68 unit uPSI_JclMultimedia;                    //JCL
33699: 69 unit uPSI_WideStrUtils;                     //Delphi API/RTL
33700: 70 unit uPSI_GraphUtil;                        //Delphi RTL
33701: 71 unit uPSI_TypeTrans;                        //Delphi RTL
33702: 72 unit uPSI_HTTPApp;                          //Delphi VCL
33703: 73 unit uPSI_DBWeb;                            //Delphi VCL
33704: 74 unit uPSI_DBBdeWeb;                         //Delphi VCL
33705: 75 unit uPSI_DBXpressWeb;                      //Delphi VCL
33706: 76 unit uPSI_ShadowWnd;                        //Delphi VCL
33707: 77 unit uPSI_ToolWin;                          //Delphi VCL
33708: 78 unit uPSI_Tabs;                             //Delphi VCL
33709: 79 unit uPSI_JclGraphUtils;                    //JCL
33710: 80 unit uPSI_JclCounter;                       //JCL
33711: 81 unit uPSI_JclSysInfo;                       //JCL
33712: 82 unit uPSI_JclSecurity;                      //JCL
33713: 83 unit uPSI_JclFileUtils;                     //JCL
33714: 84 unit uPSI_IdUserAccounts;                   //Indy
33715: 85 unit uPSI_IdAuthentication;                 //Indy
33716: 86 unit uPSI_uTPLb_AES;                        //LockBox 3
33717: 87 unit uPSI_IdHashSHA1;                       //LockBox 3
33718: 88 unit uTPLb_BlockCipher;                     //LockBox 3
33719: 89 unit uPSI_ValEdit.pas;                      //Delphi VCL
33720: 90 unit uPSI_JvVCLUtils;                       //JCL
33721: 91 unit uPSI_JvDBUtil;                         //JCL
33722: 92 unit uPSI_JvDBUtils;                        //JCL
33723: 93 unit uPSI_JvAppUtils;                       //JCL
33724: 94 unit uPSI_JvCtrlUtils;                      //JCL
33725: 95 unit uPSI_JvFormToHtml;                     //JCL
33726: 96 unit uPSI_JvParsing;                        //JCL
33727: 97 unit uPSI_SerDlgs;                          //Toolbox
33728: 98 unit uPSI_Serial;                           //Toolbox
33729: 99 unit uPSI_JvComponent;                      //JCL
33730: 100 unit uPSI_JvCalc;                          //JCL
33731: 101 unit uPSI_JvBdeUtils;                      //JCL
33732: 102 unit uPSI_JvDateUtil;                      //JCL
33733: 103 unit uPSI_JvGenetic;                       //JCL
33734: 104 unit uPSI_JclBase;                         //JCL
33735: 105 unit uPSI_JvUtils;                         //JCL
33736: 106 unit uPSI_JvStrUtil;                       //JCL
33737: 107 unit uPSI_JvStrUtils;                      //JCL
33738: 108 unit uPSI_JvFileUtil;                      //JCL
33739: 109 unit uPSI_JvMemoryInfos;                   //JCL
33740: 110 unit uPSI_JvComputerInfo;                  //JCL
33741: 111 unit uPSI_JvgCommClasses;                  //JCL
33742: 112 unit uPSI_JvgLogics;                       //JCL
33743: 113 unit uPSI_JvLED;                           //JCL
33744: 114 unit uPSI_JvTurtle;                        //JCL
33745: 115 unit uPSI_SortThds; unit uPSI_ThSort;      //maXbox
33746: 116 unit uPSI_JvgUtils;                        //JCL
33747: 117 unit uPSI_JvExprParser;                    //JCL
33748: 118 unit uPSI_HexDump;                         //Borland
33749: 119 unit uPSI_DBLogDlg;                        //VCL
33750: 120 unit uPSI_SqlTimSt;                        //RTL
33751: 121 unit uPSI_JvHtmlParser;                    //JCL
33752: 122 unit uPSI_JvgXMLSerializer;                //JCL
33753: 123 unit uPSI_JvJCLUtils;                      //JCL
33754: 124 unit uPSI_JvStrings;                       //JCL
33755: 125 unit uPSI_uTPLb_IntUtils;                  //TurboPower
33756: 126 unit uPSI_uTPLb_HugeCardinal;              //TurboPower
33757: 127 unit uPSI_uTPLb_HugeCardinalUtils;         //TurboPower
33758: 128 unit uPSI_SynRegExpr;                      //SynEdit
33759: 129 unit uPSI_StUtils;                         //SysTools4
33760: 130 unit uPSI_StToHTML;                        //SysTools4
```

```
33761: 131 unit uPSI_StStrms;                          //SysTools4
33762: 132 unit uPSI_StFIN;                            //SysTools4
33763: 133 unit uPSI_StAstroP;                         //SysTools4
33764: 134 unit uPSI_StStat;                           //SysTools4
33765: 135 unit uPSI_StNetCon;                         //SysTools4
33766: 136 unit uPSI_StDecMth;                         //SysTools4
33767: 137 unit uPSI_StOStr;                           //SysTools4
33768: 138 unit uPSI_StPtrns;                          //SysTools4
33769: 139 unit uPSI_StNetMsg;                         //SysTools4
33770: 140 unit uPSI_StMath;                           //SysTools4
33771: 141 unit uPSI_StExpEng;                         //SysTools4
33772: 142 unit uPSI_StCRC;                            //SysTools4
33773: 143 unit uPSI_StExport,                         //SysTools4
33774: 144 unit uPSI_StExpLog,                         //SysTools4
33775: 145 unit uPSI_ActnList;                         //Delphi VCL
33776: 146 unit uPSI_jpeg;                             //Borland
33777: 147 unit uPSI_StRandom;                         //SysTools4
33778: 148 unit uPSI_StDict;                           //SysTools4
33779: 149 unit uPSI_StBCD;                            //SysTools4
33780: 150 unit uPSI_StTxtDat;                         //SysTools4
33781: 151 unit uPSI_StRegEx;                          //SysTools4
33782: 152 unit uPSI_IMouse;                           //VCL
33783: 153 unit uPSI_SyncObjs;                         //VCL
33784: 154 unit uPSI_AsyncCalls;                       //Hausladen
33785: 155 unit uPSI_ParallelJobs;                     //Saraiva
33786: 156 unit uPSI_Variants;                         //VCL
33787: 157 unit uPSI_VarCmplx;                         //VCL Wolfram
33788: 158 unit uPSI_DTDSchema;                        //VCL
33789: 159 unit uPSI_ShLwApi;                          //Brakel
33790: 160 unit uPSI_IBUtils;                          //VCL
33791: 161 unit uPSI_CheckLst;                         //VCL
33792: 162 unit uPSI_JvSimpleXml;                      //JCL
33793: 163 unit uPSI_JclSimpleXml;                     //JCL
33794: 164 unit uPSI_JvXmlDatabase;                    //JCL
33795: 165 unit uPSI_JvMaxPixel;                       //JCL
33796: 166 unit uPSI_JvItemsSearchs;                   //JCL
33797: 167 unit uPSI_StExpEng2;                        //SysTools4
33798: 168 unit uPSI_StGenLog;                         //SysTools4
33799: 169 unit uPSI_JvLogFile;                        //Jcl
33800: 170 unit uPSI_CPort;                            //ComPort Lib v4.11
33801: 171 unit uPSI_CPortCtl;                         //ComPort
33802: 172 unit uPSI_CPortEsc;                         //ComPort
33803: 173 unit BarCodeScaner;                         //ComPort
33804: 174 unit uPSI_JvGraph;                          //JCL
33805: 175 unit uPSI_JvComCtrls;                       //JCL
33806: 176 unit uPSI_GUITesting;                       //D Unit
33807: 177 unit uPSI_JvFindFiles;                      //JCL
33808: 178 unit uPSI_StSystem;                         //SysTools4
33809: 179 unit uPSI_JvKeyboardStates;                 //JCL
33810: 180 unit uPSI_JvMail;                           //JCL
33811: 181 unit uPSI_JclConsole;                       //JCL
33812: 182 unit uPSI_JclLANMan;                        //JCL
33813: 183 unit uPSI_IdCustomHTTPServer;               //Indy
33814: 184 unit IdHTTPServer                           //Indy
33815: 185 unit uPSI_IdTCPServer;                      //Indy
33816: 186 unit uPSI_IdSocketHandle;                   //Indy
33817: 187 unit uPSI_IdIOHandlerSocket;                //Indy
33818: 188 unit IdIOHandler;                           //Indy
33819: 189 unit uPSI_cutils;                           //Bloodshed
33820: 190 unit uPSI_BoldUtils;                        //boldsoft
33821: 191 unit uPSI_IdSimpleServer;                   //Indy
33822: 192 unit uPSI_IdSSLOpenSSL;                     //Indy
33823: 193 unit uPSI_IdMultipartFormData;              //Indy
33824: 194 unit uPSI_SynURIOpener;                     //SynEdit
33825: 195 unit uPSI_PerlRegEx;                        //PCRE
33826: 196 unit uPSI_IdHeaderList;                     //Indy
33827: 197 unit uPSI_StFirst;                          //SysTools4
33828: 198 unit uPSI_JvCtrls;                          //JCL
33829: 199 unit uPSI_IdTrivialFTPBase;                 //Indy
33830: 200 unit uPSI_IdTrivialFTP;                     //Indy
33831: 201 unit uPSI_IdUDPBase;                        //Indy
33832: 202 unit uPSI_IdUDPClient;                      //Indy
33833: 203 unit uPSI_utypes;                           //for DMath.DLL
33834: 204 unit uPSI_ShellAPI;                         //Borland
33835: 205 unit uPSI_IdRemoteCMDClient;                //Indy
33836: 206 unit uPSI_IdRemoteCMDServer;                //Indy
33837: 207 unit IdRexecServer;                         //Indy
33838: 208 unit IdRexec; (unit uPSI_IdRexec;)          //Indy
33839: 209 unit IdUDPServer;                           //Indy
33840: 210 unit IdTimeUDPServer;                       //Indy
33841: 211 unit IdTimeServer;                          //Indy
33842: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)    //Indy
33843: 213 unit uPSI_IdIPWatch;                        //Indy
33844: 214 unit uPSI_IdIrcServer;                      //Indy
33845: 215 unit uPSI_IdMessageCollection;              //Indy
33846: 216 unit uPSI_cPEM;                             //Fundamentals 4
33847: 217 unit uPSI_cFundamentUtils;                  //Fundamentals 4
33848: 218 unit uPSI_uwinplot;                         //DMath
33849: 219 unit uPSI_xrtl_util_CPUUtils;               //ExtentedRTL
```

```
33850: 220 unit uPSI_GR32_System;                        //Graphics32
33851: 221 unit uPSI_cFileUtils;                         //Fundamentals 4
33852: 222 unit uPSI_cDateTime; (timemachine)            //Fundamentals 4
33853: 223 unit uPSI_cTimers; (high precision timer)     //Fundamentals 4
33854: 224 unit uPSI_cRandom;                            //Fundamentals 4
33855: 225 unit uPSI_ueval;                              //DMath
33856: 226 unit uPSI_xrtl_net_URIUtils;                  //ExtendedRTL
33857: 227 unit xrtl_net_URIUtils;                       //ExtendedRTL
33858: 228 unit uPSI_ufft;  (FFT)                        //DMath
33859: 229 unit uPSI_DBXChannel;                         //Delphi
33860: 230 unit uPSI_DBXIndyChannel;                     //Delphi Indy
33861: 231 unit uPSI_xrtl_util_COMCat;                   //ExtendedRTL
33862: 232 unit uPSI_xrtl_util_StrUtils;                 //ExtendedRTL
33863: 233 unit uPSI_xrtl_util_VariantUtils;             //ExtendedRTL
33864: 234 unit uPSI_xrtl_util_FileUtils;                //ExtendedRTL
33865: 235 unit xrtl_util_Compat;                        //ExtendedRTL
33866: 236 unit uPSI_OleAuto;                            //Borland
33867: 237 unit uPSI_xrtl_util_COMUtils;                 //ExtendedRTL
33868: 238 unit uPSI_CmAdmCtl;                           //Borland
33869: 239 unit uPSI_ValEdit2;                           //VCL
33870: 240 unit uPSI_GR32;  //Graphics32                 //Graphics32
33871: 241 unit uPSI_GR32_Image;                         //Graphics32
33872: 242 unit uPSI_xrtl_util_TimeUtils;                //ExtendedRTL
33873: 243 unit uPSI_xrtl_util_TimeZone;                 //ExtendedRTL
33874: 244 unit uPSI_xrtl_util_TimeStamp;                //ExtendedRTL
33875: 245 unit uPSI_xrtl_util_Map;                      //ExtendedRTL
33876: 246 unit uPSI_xrtl_util_Set;                      //ExtendedRTL
33877: 247 unit uPSI_CPortMonitor;                       //ComPort
33878: 248 unit uPSI_StIniStm;                           //SysTools4
33879: 249 unit uPSI_GR32_ExtImage;                      //Graphics32
33880: 250 unit uPSI_GR32_OrdinalMaps;                   //Graphics32
33881: 251 unit uPSI_GR32_Rasterizers;                   //Graphics32
33882: 252 unit uPSI_xrtl_util_Exception;                //ExtendedRTL
33883: 253 unit uPSI_xrtl_util_Value;                    //ExtendedRTL
33884: 254 unit uPSI_xrtl_util_Compare;                  //ExtendedRTL
33885: 255 unit uPSI_FlatSB;                             //VCL
33886: 256 unit uPSI_JvAnalogClock;                      //JCL
33887: 257 unit uPSI_JvAlarms;                           //JCL
33888: 258 unit uPSI_JvSQLS;                             //JCL
33889: 259 unit uPSI_JvDBSecur;                          //JCL
33890: 260 unit uPSI_JvDBQBE;                            //JCL
33891: 261 unit uPSI_JvStarfield;                        //JCL
33892: 262 unit uPSI_JVCLMiscal;                         //JCL
33893: 263 unit uPSI_JvProfiler32;                       //JCL
33894: 264 unit uPSI_JvDirectories,                      //JCL
33895: 265 unit uPSI_JclSchedule,                        //JCL
33896: 266 unit uPSI_JclSvcCtrl,                         //JCL
33897: 267 unit uPSI_JvSoundControl,                     //JCL
33898: 268 unit uPSI_JvBDESQLScript,                     //JCL
33899: 269 unit uPSI_JvgDigits,                          //JCL>
33900: 270 unit uPSI_ImgList;                            //TCustomImageList
33901: 271 unit uPSI_JclMIDI;                            //JCL>
33902: 272 unit uPSI_JclWinMidi;                         //JCL>
33903: 273 unit uPSI_JclNTFS;                            //JCL>
33904: 274 unit uPSI_JclAppInst;                         //JCL>
33905: 275 unit uPSI_JvRle;                              //JCL>
33906: 276 unit uPSI_JvRas32;                            //JCL>
33907: 277 unit uPSI_JvImageDrawThread,                  //JCL>
33908: 278 unit uPSI_JvImageWindow,                      //JCL>
33909: 279 unit uPSI_JvTransparentForm;                  //JCL>
33910: 280 unit uPSI_JvWinDialogs;                       //JCL>
33911: 281 unit uPSI_JvSimLogic,                         //JCL>
33912: 282 unit uPSI_JvSimIndicator,                     //JCL>
33913: 283 unit uPSI_JvSimPID,                           //JCL>
33914: 284 unit uPSI_JvSimPIDLinker,                     //JCL>
33915: 285 unit uPSI_IdRFCReply;                         //Indy
33916: 286 unit uPSI_IdIdent;                            //Indy
33917: 287 unit uPSI_IdIdentServer;                      //Indy
33918: 288 unit uPSI_JvPatchFile;                        //JCL
33919: 289 unit uPSI_StNetPfm;                           //SysTools4
33920: 290 unit uPSI_StNet;                              //SysTools4
33921: 291 unit uPSI_JclPeImage;                         //JCL
33922: 292 unit uPSI_JclPrint;                           //JCL
33923: 293 unit uPSI_JclMime;                            //JCL
33924: 294 unit uPSI_JvRichEdit;                         //JCL
33925: 295 unit uPSI_JvDBRichEd;                         //JCL
33926: 296 unit uPSI_JvDice;                             //JCL
33927: 297 unit uPSI_JvFloatEdit;                        //JCL 3.9.8
33928: 298 unit uPSI_JvDirFrm;                           //JCL
33929: 299 unit uPSI_JvDualList;                         //JCL
33930: 300 unit uPSI_JvSwitch;                           ////JCL
33931: 301 unit uPSI_JvTimerLst;                         ////JCL
33932: 302 unit uPSI_JvMemTable;                         //JCL
33933: 303 unit uPSI_JvObjStr;                           //JCL
33934: 304 unit uPSI_StLArr;                             //SysTools4
33935: 305 unit uPSI_StWmDCpy;                           //SysTools4
33936: 306 unit uPSI_StText;                             //SysTools4
33937: 307 unit uPSI_StNTLog;                            //SysTools4
33938: 308 unit uPSI_xrtl_math_Int;                      //ExtendedRTL
```

```
33939: 309 unit uPSI_JvImagPrvw;                        //JCL
33940: 310 unit uPSI_JvFormPatch;                       //JCL
33941: 311 unit uPSI_JvPicClip;                         //JCL
33942: 312 unit uPSI_JvDataConv;                        //JCL
33943: 313 unit uPSI_JvCpuUsage;                        //JCL
33944: 314 unit uPSI_JclUnitConv_mX2;                   //JCL
33945: 315 unit JvDualListForm;                         //JCL
33946: 316 unit uPSI_JvCpuUsage2;                       //JCL
33947: 317 unit uPSI_JvParserForm;                      //JCL
33948: 318 unit uPSI_JvJanTreeView;                     //JCL
33949: 319 unit uPSI_JvTransLED;                        //JCL
33950: 320 unit uPSI_JvPlaylist;                        //JCL
33951: 321 unit uPSI_JvFormAutoSize;                    //JCL
33952: 322 unit uPSI_JvYearGridEditForm;                //JCL
33953: 323 unit uPSI_JvMarkupCommon;                    //JCL
33954: 324 unit uPSI_JvChart;                           //JCL
33955: 325 unit uPSI_JvXPCore;                          //JCL
33956: 326 unit uPSI_JvXPCoreUtils;                     //JCL
33957: 327 unit uPSI_StatsClasses;                      //mX4  a1: TStatArray;
33958: 328 unit uPSI_ExtCtrls2;                         //VCL
33959: 329 unit uPSI_JvUrlGrabbers;                     //JCL
33960: 330 unit uPSI_JvXmlTree;                         //JCL
33961: 331 unit uPSI_JvWavePlayer;                      //JCL
33962: 332 unit uPSI_JvUnicodeCanvas;                   //JCL
33963: 333 unit uPSI_JvTFUtils;                         //JCL
33964: 334 unit uPSI_IdServerIOHandler;                 //Indy
33965: 335 unit uPSI_IdServerIOHandlerSocket;           //Indy
33966: 336 unit uPSI_IdMessageCoder;                    //Indy
33967: 337 unit uPSI_IdMessageCoderMIME;                //Indy
33968: 338 unit uPSI_IdMIMETypes;                       //Indy
33969: 339 unit uPSI_JvConverter;                       //JCL
33970: 340 unit uPSI_JvCsvParse;                        //JCL
33971: 341 unit uPSI_umath;  unit uPSI_ugamma;          //DMath
33972: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
33973: 343 unit uPSI_JvDBGridExport;                    //JCL
33974: 344 unit uPSI_JvgExport;                         //JCL
33975: 345 unit uPSI_JvSerialMaker;                     //JCL
33976: 346 unit uPSI_JvWin32;                           //JCL
33977: 347 unit uPSI_JvPaintFX;                         //JCL
33978: 348 unit uPSI_JvOracleDataSet; (beta)            //JCL
33979: 349 unit uPSI_JvValidators; (preview)            //JCL
33980: 350 unit uPSI_JvNTEventLog;                      //JCL
33981: 351 unit uPSI_ShellZipTool;                      //mX4
33982: 352 unit uPSI_JvJoystick;                        //JCL
33983: 353 unit uPSI_JvMailSlots;                       //JCL
33984: 354 unit uPSI_JclComplex;                        //JCL
33985: 355 unit uPSI_SynPdf;                            //Synopse
33986: 356 unit uPSI_Registry;                          //VCL
33987: 357 unit uPSI_TlHelp32;                          //VCL
33988: 358 unit uPSI_JclRegistry;                       //JCL
33989: 359 unit uPSI_JvAirBrush;                        //JCL
33990: 360 unit uPSI_mORMotReport;                      //Synopse
33991: 361 unit uPSI_JclLocales;                        //JCL
33992: 362 unit uPSI_SynEdit;                           //SynEdit
33993: 363 unit uPSI_SynEditTypes;                      //SynEdit
33994: 364 unit uPSI_SynMacroRecorder;                  //SynEdit
33995: 365 unit uPSI_LongIntList;                       //SynEdit
33996: 366 unit uPSI_devcutils;                         //DevC
33997: 367 unit uPSI_SynEditMiscClasses;                //SynEdit
33998: 368 unit uPSI_SynEditRegexSearch;                //SynEdit
33999: 369 unit uPSI_SynEditHighlighter;                //SynEdit
34000: 370 unit uPSI_SynHighlighterPas;                 //SynEdit
34001: 371 unit uPSI_JvSearchFiles;                     //JCL
34002: 372 unit uPSI_SynHighlighterAny;                 //Lazarus
34003: 373 unit uPSI_SynEditKeyCmds;                    //SynEdit
34004: 374 unit uPSI_SynEditMiscProcs,                  //SynEdit
34005: 375 unit uPSI_SynEditKbdHandler                  //SynEdit
34006: 376 unit uPSI_JvAppInst,                         //JCL
34007: 377 unit uPSI_JvAppEvent;                        //JCL
34008: 378 unit uPSI_JvAppCommand;                      //JCL
34009: 379 unit uPSI_JvAnimTitle;                       //JCL
34010: 380 unit uPSI_JvAnimatedImage;                   //JCL
34011: 381 unit uPSI_SynEditExport;                     //SynEdit
34012: 382 unit uPSI_SynExportHTML;                     //SynEdit
34013: 383 unit uPSI_SynExportRTF;                      //SynEdit
34014: 384 unit uPSI_SynEditSearch;                     //SynEdit
34015: 385 unit uPSI_fMain_back                         //maXbox;
34016: 386 unit uPSI_JvZoom;                            //JCL
34017: 387 unit uPSI_PMrand;                            //PM
34018: 388 unit uPSI_JvSticker;                         //JCL
34019: 389 unit uPSI_XmlVerySimple;                     //mX4
34020: 390 unit uPSI_Services;                          //ExtPascal
34021: 391 unit uPSI_ExtPascalUtils;                    //ExtPascal
34022: 392 unit uPSI_SocketsDelphi;                     //ExtPascal
34023: 393 unit uPSI_StBarC;                            //SysTools
34024: 394 unit uPSI_StDbBarC;                          //SysTools
34025: 395 unit uPSI_StBarPN;                           //SysTools
34026: 396 unit uPSI_StDbPNBC;                          //SysTools
34027: 397 unit uPSI_StDb2DBC;                          //SysTools
```

```
34028: 398 unit uPSI_StMoney;                          //SysTools
34029: 399 unit uPSI_JvForth;                          //JCL
34030: 400 unit uPSI_RestRequest;                      //mX4
34031: 401 unit uPSI_HttpRESTConnectionIndy;           //mX4
34032: 402 unit uPSI_JvXmlDatabase; //update           //JCL
34033: 403 unit uPSI_StAstro;                          //SysTools
34034: 404 unit uPSI_StSort;                           //SysTools
34035: 405 unit uPSI_StDate;                           //SysTools
34036: 406 unit uPSI_StDateSt;                         //SysTools
34037: 407 unit uPSI_StBase;                           //SysTools
34038: 408 unit uPSI_StVInfo;                          //SysTools
34039: 409 unit uPSI_JvBrowseFolder;                   //JCL
34040: 410 unit uPSI_JvBoxProcs;                       //JCL
34041: 411 unit uPSI_urandom; (unit uranuvag;)         //DMath
34042: 412 unit uPSI_usimann; (unit ugenalg;)          //DMath
34043: 413 unit uPSI_JvHighlighter;                    //JCL
34044: 414 unit uPSI_Diff;                             //mX4
34045: 415 unit uPSI_SpringWinAPI;                     //DSpring
34046: 416 unit uPSI_StBits;                           //SysTools
34047: 417 unit uPSI_TomDBQue;                         //mX4
34048: 418 unit uPSI_MultilangTranslator;              //mX4
34049: 419 unit uPSI_HyperLabel;                       //mX4
34050: 420 unit uPSI_Starter;                          //mX4
34051: 421 unit uPSI_FileAssocs;                       //devC
34052: 422 unit uPSI_devFileMonitorX;                  //devC
34053: 423 unit uPSI_devrun;                           //devC
34054: 424 unit uPSI_devExec;                          //devC
34055: 425 unit uPSI_oysUtils;                         //devC
34056: 426 unit uPSI_DosCommand;                       //devC
34057: 427 unit uPSI_CppTokenizer;                     //devC
34058: 428 unit uPSI_JvHLParser;                       //devC
34059: 429 unit uPSI_JclMapi;                          //JCL
34060: 430 unit uPSI_JclShell;                         //JCL
34061: 431 unit uPSI_JclCOM;                           //JCL
34062: 432 unit uPSI_GR32_Math;                        //Graphics32
34063: 433 unit uPSI_GR32_LowLevel;                    //Graphics32
34064: 434 unit uPSI_SimpleHl;                         //mX4
34065: 435 unit uPSI_GR32_Filters,                     //Graphics32
34066: 436 unit uPSI_GR32_VectorMaps;                  //Graphics32
34067: 437 unit uPSI_cXMLFunctions;                    //Fundamentals 4
34068: 438 unit uPSI_JvTimer;                          //JCL
34069: 439 unit uPSI_cHTTPUtils;                       //Fundamentals 4
34070: 440 unit uPSI_cTLSUtils;                        //Fundamentals 4
34071: 441 unit uPSI_JclGraphics;                      //JCL
34072: 442 unit uPSI_JclSynch;                         //JCL
34073: 443 unit uPSI_IdTelnet;                         //Indy
34074: 444 unit uPSI_IdTelnetServer,                   //Indy
34075: 445 unit uPSI_IdEcho,                           //Indy
34076: 446 unit uPSI_IdEchoServer,                     //Indy
34077: 447 unit uPSI_IdEchoUDP,                        //Indy
34078: 448 unit uPSI_IdEchoUDPServer,                  //Indy
34079: 449 unit uPSI_IdSocks,                          //Indy
34080: 450 unit uPSI_IdAntiFreezeBase;                 //Indy
34081: 451 unit uPSI_IdHostnameServer;                 //Indy
34082: 452 unit uPSI_IdTunnelCommon,                   //Indy
34083: 453 unit uPSI_IdTunnelMaster,                   //Indy
34084: 454 unit uPSI_IdTunnelSlave,                    //Indy
34085: 455 unit uPSI_IdRSH,                            //Indy
34086: 456 unit uPSI_IdRSHServer,                      //Indy
34087: 457 unit uPSI_Spring_Cryptography_Utils;        //Spring4Delphi
34088: 458 unit uPSI_MapReader,                        //devC
34089: 459 unit uPSI_LibTar,                           //devC
34090: 460 unit uPSI_IdStack;                          //Indy
34091: 461 unit uPSI_IdBlockCipherIntercept;           //Indy
34092: 462 unit uPSI_IdChargenServer;                  //Indy
34093: 463 unit uPSI_IdFTPServer,                      //Indy
34094: 464 unit uPSI_IdException,                      //Indy
34095: 465 unit uPSI_utexplot;                         //DMath
34096: 466 unit uPSI_uwinstr;                          //DMath
34097: 467 unit uPSI_VarRecUtils;                      //devC
34098: 468 unit uPSI_JvStringListToHtml,               //JCL
34099: 469 unit uPSI_JvStringHolder,                   //JCL
34100: 470 unit uPSI_IdCoder;                          //Indy
34101: 471 unit uPSI_SynHighlighterDfm;                //Synedit
34102: 472 unit uHighlighterProcs; in 471              //Synedit
34103: 473 unit uPSI_LazFileUtils,                     //LCL
34104: 474 unit uPSI_IDECmdLine;                       //LCL
34105: 475 unit uPSI_lazMasks;                         //LCL
34106: 476 unit uPSI_ip_misc;                          //mX4
34107: 477 unit uPSI_Barcode;                          //LCL
34108: 478 unit uPSI_SimpleXML;                        //LCL
34109: 479 unit uPSI_JclIniFiles;                      //JCL
34110: 480 unit uPSI_D2XXUnit;  {$X-}                  //FTDI
34111: 481 unit uPSI_JclDateTime;                      //JCL
34112: 482 unit uPSI_JclEDI;                           //JCL
34113: 483 unit uPSI_JclMiscel2;                       //JCL
34114: 484 unit uPSI_JclValidation;                    //JCL
34115: 485 unit uPSI_JclAnsistrs; {-PString}        //JCL
34116: 486 unit uPSI_SynEditMiscProcs2;                //Synedit
```

```
34117: 487 unit uPSI_JclStreams;                        //JCL
34118: 488 unit uPSI_QRCode;                            //mX4
34119: 489 unit uPSI_BlockSocket;                       //ExtPascal
34120: 490 unit uPSI_Masks,Utils                        //VCL
34121: 491 unit uPSI_synautil;                          //Synapse!
34122: 492 unit uPSI_JclMath_Class;                     //JCL RTL
34123: 493 unit ugamdist; //Gamma Func            //DMath
34124: 494 unit uibeta, ucorrel; //IBeta               //DMath
34125: 495 unit uPSI_SRMgr;                             //mX4
34126: 496 unit uPSI_HotLog;                            //mX4
34127: 497 unit uPSI_DebugBox;                          //mX4
34128: 498 unit uPSI_ustrings;                          //DMath
34129: 499 unit uPSI_uregtest;                          //DMath
34130: 500 unit uPSI_usimplex;                          //DMath
34131: 501 unit uPSI_uhyper;                            //DMath
34132: 502 unit uPSI_IdHL7;                             //Indy
34133: 503 unit uPSI_IdIPMCastBase,                     //Indy
34134: 504 unit uPSI_IdIPMCastServer;                   //Indy
34135: 505 unit uPSI_IdIPMCastClient;                   //Indy
34136: 506 unit uPSI_unlfit; //nlregression            //DMath
34137: 507 unit uPSI_IdRawHeaders;                      //Indy
34138: 508 unit uPSI_IdRawClient;                       //Indy
34139: 509 unit uPSI_IdRawFunctions;                    //Indy
34140: 510 unit uPSI_IdTCPStream;                       //Indy
34141: 511 unit uPSI_IdSNPP;                            //Indy
34142: 512 unit uPSI_St2DBarC;                          //SysTools
34143: 513 unit uPSI_ImageWin;  //FTL                  //VCL
34144: 514 unit uPSI_CustomDrawTreeView; //FTL         //VCL
34145: 515 unit uPSI_GraphWin;  //FTL                  //VCL
34146: 516 unit uPSI_actionMain;  //FTL                //VCL
34147: 517 unit uPSI_StSpawn;                           //SysTools
34148: 518 unit uPSI_CtlPanel;                          //VCL
34149: 519 unit uPSI_IdLPR;                             //Indy
34150: 520 unit uPSI_SockRequestInterpreter;            //Indy
34151: 521 unit uPSI_ulambert;                          //DMath
34152: 522 unit uPSI_ucholesk;                          //DMath
34153: 523 unit uPSI_SimpleDS;                          //VCL
34154: 524 unit uPSI_DBXSqlScanner;                     //VCL
34155: 525 unit uPSI_DBXMetaDataUtil;                   //VCL
34156: 526 unit uPSI_Chart;                             //TEE
34157: 527 unit uPSI_TeeProcs;                          //TEE
34158: 528 unit mXBDEUtils;                             //mX4
34159: 529 unit uPSI_MDIEdit;                           //VCL
34160: 530 unit uPSI_CopyPrsr;                          //VCL
34161: 531 unit uPSI_SockApp;                           //VCL
34162: 532 unit uPSI_AppEvnts;                          //VCL
34163: 533 unit uPSI_ExtActns;                          //VCL
34164: 534 unit uPSI_TeEngine;                          //TEE
34165: 535 unit uPSI_CoolMain; //browser               //VCL
34166: 536 unit uPSI_StCRC;                             //SysTools
34167: 537 unit uPSI_StDecMth2;                         //SysTools
34168: 538 unit uPSI_frmExportMain;                     //Synedit
34169: 539 unit uPSI_SynDBEdit;                         //Synedit
34170: 540 unit uPSI_SynEditWildcardSearch;             //Synedit
34171: 541 unit uPSI_BoldComUtils;                      //BOLD
34172: 542 unit uPSI_BoldIsoDateTime;                   //BOLD
34173: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils       //BOLD
34174: 544 unit uPSI_BoldXMLRequests;                   //BOLD
34175: 545 unit uPSI_BoldStringList;                    //BOLD
34176: 546 unit uPSI_BoldFileHandler;                   //BOLD
34177: 547 unit uPSI_BoldContainers;                    //BOLD
34178: 548 unit uPSI_BoldQueryUserDlg;                  //BOLD
34179: 549 unit uPSI_BoldWinINet;                       //BOLD
34180: 550 unit uPSI_BoldQueue;                         //BOLD
34181: 551 unit uPSI_JvPcx;                             //JCL
34182: 552 unit uPSI_IdWhois;                           //Indy
34183: 553 unit uPSI_IdWhoIsServer;                     //Indy
34184: 554 unit uPSI_IdGopher;                          //Indy
34185: 555 unit uPSI_IdDateTimeStamp;                   //Indy
34186: 556 unit uPSI_IdDayTimeServer;                   //Indy
34187: 557 unit uPSI_IdDayTimeUDP;                      //Indy
34188: 558 unit uPSI_IdDayTimeUDPServer;                //Indy
34189: 559 unit uPSI_IdDICTServer;                      //Indy
34190: 560 unit uPSI_IdDiscardServer;                   //Indy
34191: 561 unit uPSI_IdDiscardUDPServer;                //Indy
34192: 562 unit uPSI_IdMappedFTP;                       //Indy
34193: 563 unit uPSI_IdMappedPortTCP;                   //Indy
34194: 564 unit uPSI_IdGopherServer;                    //Indy
34195: 565 unit uPSI_IdQotdServer;                      //Indy
34196: 566 unit uPSI_JvRgbToHtml;                       //JCL
34197: 567 unit uPSI_JvRemLog,                          //JCL
34198: 568 unit uPSI_JvSysComp;                         //JCL
34199: 569 unit uPSI_JvTMTL;                            //JCL
34200: 570 unit uPSI_JvWinampAPI;                       //JCL
34201: 571 unit uPSI_MSysUtils;                         //mX4
34202: 572 unit uPSI_ESBMaths;                          //ESB
34203: 573 unit uPSI_ESBMaths2;                         //ESB
34204: 574 unit uPSI_uLkJSON;                           //Lk
34205: 575 unit uPSI_ZURL;  //Zeos                     //Zeos
```

```
34206: 576 unit uPSI_ZSysUtils;                          //Zeos
34207: 577 unit unaUtils internals                       //UNA
34208: 578 unit uPSI_ZMatchPattern;                      //Zeos
34209: 579 unit uPSI_ZClasses;                           //Zeos
34210: 580 unit uPSI_ZCollections;                       //Zeos
34211: 581 unit uPSI_ZEncoding;                          //Zeos
34212: 582 unit uPSI_IdRawBase;                          //Indy
34213: 583 unit uPSI_IdNTLM;                             //Indy
34214: 584 unit uPSI_IdNNTP;                             //Indy
34215: 585 unit uPSI_usniffer; //PortScanForm            //mX4
34216: 586 unit uPSI_IdCoderMIME;                        //Indy
34217: 587 unit uPSI_IdCoderUUE;                         //Indy
34218: 588 unit uPSI_IdCoderXXE;                         //Indy
34219: 589 unit uPSI_IdCoder3to4;                        //Indy
34220: 590 unit uPSI_IdCookie;                           //Indy
34221: 591 unit uPSI_IdCookieManager;                    //Indy
34222: 592 unit uPSI_WDosSocketUtils;                    //WDos
34223: 593 unit uPSI_WDosPlcUtils;                       //WDos
34224: 594 unit uPSI_WDosPorts;                          //WDos
34225: 595 unit uPSI_WDosResolvers;                      //WDos
34226: 596 unit uPSI_WDosTimers;                         //WDos
34227: 597 unit uPSI_WDosPlcs;                           //WDos
34228: 598 unit uPSI_WDosPneumatics;                     //WDos
34229: 599 unit uPSI_IdFingerServer;                     //Indy
34230: 600 unit uPSI_IdDNSResolver;                      //Indy
34231: 601 unit uPSI_IdHTTPWebBrokerBridge;              //Indy
34232: 602 unit uPSI_IdIntercept;                        //Indy
34233: 603 unit uPSI_IdIPMCastBase;                      //Indy
34234: 604 unit uPSI_IdLogBase;                          //Indy
34235: 605 unit uPSI_IdIOHandlerStream;                  //Indy
34236: 606 unit uPSI_IdMappedPortUDP;                    //Indy
34237: 607 unit uPSI_IdQOTDUDPServer;                    //Indy
34238: 608 unit uPSI_IdQOTDUDP;                          //Indy
34239: 609 unit uPSI_IdSysLog;                           //Indy
34240: 610 unit uPSI_IdSysLogServer;                     //Indy
34241: 611 unit uPSI_IdSysLogMessage;                    //Indy
34242: 612 unit uPSI_IdTimeServer;                       //Indy
34243: 613 unit uPSI_IdTimeUDP;                          //Indy
34244: 614 unit uPSI_IdTimeUDPServer;                    //Indy
34245: 615 unit uPSI_IdUserAccounts;                     //Indy
34246: 616 unit uPSI_TextUtils;                          //mX4
34247: 617 unit uPSI_MandelbrotEngine;                   //mX4
34248: 618 unit uPSI_delphi_arduino_Unit1;               //mX4
34249: 619 unit uPSI_DTDSchema2;                         //mX4
34250: 620 unit uPSI_fplotMain;                          //DMath
34251: 621 unit uPSI_FindFileIter;                       //mX4
34252: 622 unit uPSI_PppState;  (JclStrHashMap)          //PPP
34253: 623 unit uPSI_PppParser;                          //PPP
34254: 624 unit uPSI_PppLexer;                           //PPP
34255: 625 unit uPSI_PCharUtils;                         //PPP
34256: 626 unit uPSI_uJSON;                              //WU
34257: 627 unit uPSI_JclStrHashMap;                      //JCL
34258: 628 unit uPSI_JclHookExcept;                      //JCL
34259: 629 unit uPSI_EncdDecd;                           //VCL
34260: 630 unit uPSI_SockAppReg;                         //VCL
34261: 631 unit uPSI_PJFileHandle;                       //PJ
34262: 632 unit uPSI_PJEnvVars;                          //PJ
34263: 633 unit uPSI_PJPipe;                             //PJ
34264: 634 unit uPSI_PJPipeFilters;                      //PJ
34265: 635 unit uPSI_PJConsoleApp;                      //PJ
34266: 636 unit uPSI_UConsoleAppEx;                     //PJ
34267: 637 unit uPSI_DbxSocketChannelNative,            //VCL
34268: 638 unit uPSI_DbxDataGenerator,                  //VCL
34269: 639 unit uPSI_DBXClient;                         //VCL
34270: 640 unit uPSI_IdLogEvent;                        //Indy
34271: 641 unit uPSI_Reversi;                           //mX4
34272: 642 unit uPSI_Geometry;                          //mX4
34273: 643 unit uPSI_IdSMTPServer;                      //Indy
34274: 644 unit uPSI_Textures;                          //mX4
34275: 645 unit uPSI_IBX;                               //VCL
34276: 646 unit uPSI_IWDBCommon;                        //VCL
34277: 647 unit uPSI_SortGrid;                          //mX4
34278: 648 unit uPSI_IB;                                //VCL
34279: 649 unit uPSI_IBScript;                          //VCL
34280: 650 unit uPSI_JvCSVBaseControls;                 //JCL
34281: 651 unit uPSI_Jvg3DColors;                       //JCL
34282: 652 unit uPSI_JvHLEditor;  //beat                //JCL
34283: 653 unit uPSI_JvShellHook;                       //JCL
34284: 654 unit uPSI_DBCommon2                          //VCL
34285: 655 unit uPSI_JvSHFileOperation;                 //JCL
34286: 656 unit uPSI_uFilexport;                        //mX4
34287: 657 unit uPSI_JvDialogs;                         //JCL
34288: 658 unit uPSI_JvDBTreeView;                      //JCL
34289: 659 unit uPSI_JvDBUltimGrid;                     //JCL
34290: 660 unit uPSI_JvDBQueryParamsForm;               //JCL
34291: 661 unit uPSI_JvExControls;                      //JCL
34292: 662 unit uPSI_JvBDEMemTable;                     //JCL
34293: 663 unit uPSI_JvCommStatus;                      //JCL
34294: 664 unit uPSI_JvMailSlots2;                      //JCL
```

```
34295: 665 unit uPSI_JvgWinMask;                          //JCL
34296: 666 unit uPSI_StEclpse;                            //SysTools
34297: 667 unit uPSI_StMime;                              //SysTools
34298: 668 unit uPSI_StList;                              //SysTools
34299: 669 unit uPSI_StMerge;                             //SysTools
34300: 670 unit uPSI_StStrS;                              //SysTools
34301: 671 unit uPSI_StTree,                              //SysTools
34302: 672 unit uPSI_StVArr;                              //SysTools
34303: 673 unit uPSI_StRegIni;                            //SysTools
34304: 674 unit uPSI_urkf;                                //DMath
34305: 675 unit uPSI_usvd;                                //DMath
34306: 676 unit uPSI_DepWalkUtils;                        //JCL
34307: 677 unit uPSI_OptionsFrm;                          //JCL
34308: 678 unit yuvconverts;                              //mX4
34309: 679 uPSI_JvPropAutoSave;                           //JCL
34310: 680 uPSI_AclAPI;                                   //alcinoe
34311: 681 uPSI_AviCap;                                   //alcinoe
34312: 682 uPSI_ALAVLBinaryTree;                          //alcinoe
34313: 683 uPSI_ALFcnMisc;                                //alcinoe
34314: 684 uPSI_ALStringList;                             //alcinoe
34315: 685 uPSI_ALQuickSortList;                          //alcinoe
34316: 686 uPSI_ALStaticText;                             //alcinoe
34317: 687 uPSI_ALJSONDoc;                                //alcinoe
34318: 688 uPSI_ALGSMComm;                                //alcinoe
34319: 689 uPSI_ALWindows;                                //alcinoe
34320: 690 uPSI_ALMultiPartFormDataParser;                //alcinoe
34321: 691 uPSI_ALHttpCommon;                             //alcinoe
34322: 692 uPSI_ALWebSpider,                              //alcinoe
34323: 693 uPSI_ALHttpClient;                             //alcinoe
34324: 694 uPSI_ALFcnHTML;                                //alcinoe
34325: 695 uPSI_ALFTPClient;                              //alcinoe
34326: 696 uPSI_ALInternetMessageCommon;                  //alcinoe
34327: 697 uPSI_ALWininetHttpClient;                      //alcinoe
34328: 698 uPSI_ALWinInetFTPClient;                       //alcinoe
34329: 699 uPSI_ALWinHttpWrapper;                         //alcinoe
34330: 700 uPSI_ALWinHttpClient;                          //alcinoe
34331: 701 uPSI_ALFcnWinSock;                             //alcinoe
34332: 702 uPSI_ALFcnSQL;                                 //alcinoe
34333: 703 uPSI_ALFcnCGI;                                 //alcinoe
34334: 704 uPSI_ALFcnExecute;                             //alcinoe
34335: 705 uPSI_ALFcnFile;                                //alcinoe
34336: 706 uPSI_ALFcnMime;                                //alcinoe
34337: 707 uPSI_ALPhpRunner;                              //alcinoe
34338: 708 uPSI_ALGraphic;                                //alcinoe
34339: 709 uPSI_ALIniFiles;                               //alcinoe
34340: 710 uPSI_ALMemCachedClient;                        //alcinoe
34341: 711 unit uPSI_MyGrids;                             //mX4
34342: 712 uPSI_ALMultiPartMixedParser                    //alcinoe
34343: 713 uPSI_ALSMTPClient                              //alcinoe
34344: 714 uPSI_ALNNTPClient;                             //alcinoe
34345: 715 uPSI_ALHintBalloon;                            //alcinoe
34346: 716 unit uPSI_ALXmlDoc;                            //alcinoe
34347: 717 unit uPSI_IPCThrd;                             //VCL
34348: 718 unit uPSI_MonForm;                             //VCL
34349: 719 unit uPSI_TeCanvas;                            //Orpheus
34350: 720 unit uPSI_Ovcmisc;                             //Orpheus
34351: 721 unit uPSI_ovcfiler;                            //Orpheus
34352: 722 unit uPSI_ovcstate;                            //Orpheus
34353: 723 unit uPSI_ovccoco;                             //Orpheus
34354: 724 unit uPSI_ovcrvexp;                            //Orpheus
34355: 725 unit uPSI_OvcFormatSettings;                   //Orpheus
34356: 726 unit uPSI_OvcUtils;                            //Orpheus
34357: 727 unit uPSI_ovcstore;                            //Orpheus
34358: 728 unit uPSI_ovcstr;                              //Orpheus
34359: 729 unit uPSI_ovcmru;                              //Orpheus
34360: 730 unit uPSI_ovccmd;                              //Orpheus
34361: 731 unit uPSI_ovctimer;                            //Orpheus
34362: 732 unit uPSI_ovcintl;                             //Orpheus
34363: 733 uPSI_AfCircularBuffer;                         //AsyncFree
34364: 734 uPSI_AfUtils;                                  //AsyncFree
34365: 735 uPSI_AfSafeSync;                               //AsyncFree
34366: 736 uPSI_AfComPortCore;                            //AsyncFree
34367: 737 uPSI_AfComPort;                                //AsyncFree
34368: 738 uPSI_AfPortControls;                           //AsyncFree
34369: 739 uPSI_AfDataDispatcher;                         //AsyncFree
34370: 740 uPSI_AfViewers;                                //AsyncFree
34371: 741 uPSI_AfDataTerminal;                           //AsyncFree
34372: 742 uPSI_SimplePortMain;                           //AsyncFree
34373: 743 unit uPSI_ovcclock;                            //Orpheus
34374: 744 unit uPSI_o32intlst;                           //Orpheus
34375: 745 unit uPSI_o32ledlabel;                         //Orpheus
34376: 746 unit uPSI_AlMySqlClient;                       //alcinoe
34377: 747 unit uPSI_ALFBXClient;                         //alcinoe
34378: 748 unit uPSI_ALFcnSQL;                            //alcinoe
34379: 749 unit uPSI_AsyncTimer;                          //mX4
34380: 750 unit uPSI_ApplicationFileIO;                   //mX4
34381: 751 unit uPSI_PsAPI;                               //VCLé
34382: 752 uPSI_ovcuser;                                  //Orpheus
34383: 753 uPSI_ovcurl;                                   //Orpheus
```

```
34384: 754 uPSI_ovcvlb;                                  //Orpheus
34385: 755 uPSI_ovccolor;                                //Orpheus
34386: 756 uPSI_ALFBXLib,                                //alcinoe
34387: 757 uPSI_ovcmeter;                                //Orpheus
34388: 758 uPSI_ovcpeakm;                                //Orpheus
34389: 759 uPSI_O32BGSty;                                //Orpheus
34390: 760 uPSI_ovcBidi;                                 //Orpheus
34391: 761 uPSI_ovctcary;                                //Orpheus
34392: 762 uPSI_DXPUtils;                                //mX4
34393: 763 uPSI_ALMultiPartBaseParser;                   //alcinoe
34394: 764 uPSI_ALMultiPartAlternativeParser;            //alcinoe
34395: 765 uPSI_ALPOP3Client;                            //alcinoe
34396: 766 uPSI_SmallUtils;                              //mX4
34397: 767 uPSI_MakeApp;                                 //mX4
34398: 768 uPSI_O32MouseMon;                             //Orpheus
34399: 769 uPSI_OvcCache;                                //Orpheus
34400: 770 uPSI_ovccalc;                                 //Orpheus
34401: 771 uPSI_Joystick,                                //OpenGL
34402: 772 uPSI_ScreenSaver;                             //OpenGL
34403: 773 uPSI_XCollection,                             //OpenGL
34404: 774 uPSI_Polynomials,                             //OpenGL
34405: 775 uPSI_PersistentClasses, //9.86                //OpenGL
34406: 776 uPSI_VectorLists;                             //OpenGL
34407: 777 uPSI_XOpenGL,                                 //OpenGL
34408: 778 uPSI_MeshUtils;                               //OpenGL
34409: 779 unit uPSI_JclSysUtils;                        //JCL
34410: 780 unit uPSI_JclBorlandTools;                    //JCL
34411: 781 unit JclFileUtils_max;                        //JCL
34412: 782 uPSI_AfDataControls,                          //AsyncFree
34413: 783 uPSI_GLSilhouette;                            //OpenGL
34414: 784 uPSI_JclSysUtils_class;                       //JCL
34415: 785 uPSI_JclFileUtils_class;                      //JCL
34416: 786 uPSI_FileUtil;                                //JCL
34417: 787 uPSI_changefind;                              //mX4
34418: 788 uPSI_cmdIntf;                                 //mX4
34419: 789 uPSI_fservice;                                //mX4
34420: 790 uPSI_Keyboard;                                //OpenGL
34421: 791 uPSI_VRMLParser,                              //OpenGL
34422: 792 uPSI_GLFileVRML,                              //OpenGL
34423: 793 uPSI_Octree;                                  //OpenGL
34424: 794 uPSI_GLPolyhedron,                            //OpenGL
34425: 795 uPSI_GLCrossPlatform;                         //OpenGL
34426: 796 uPSI_GLParticles;                             //OpenGL
34427: 797 uPSI_GLNavigator;                             //OpenGL
34428: 798 uPSI_GLStarRecord;                            //OpenGL
34429: 799 uPSI_GLTextureCombiners;                      //OpenGL
34430: 800 uPSI_GLCanvas;                                //OpenGL
34431: 801 uPSI_GeometryBB;                              //OpenGL
34432: 802 uPSI_GeometryCoordinates;                     //OpenGL
34433: 803 uPSI_VectorGeometry;                          //OpenGL
34434: 804 uPSI_BumpMapping;                             //OpenGL
34435: 805 uPSI_TGA;                                     //OpenGL
34436: 806 uPSI_GLVectorFileObjects;                     //OpenGL
34437: 807 uPSI_IMM;                                     //VCL
34438: 808 uPSI_CategoryButtons;                        //VCL
34439: 809 uPSI_ButtonGroup;                            //VCL
34440: 810 uPSI_DbExcept;                               //VCL
34441: 811 uPSI_AxCtrls;                                //VCL
34442: 812 uPSI_GL_actorUnit1;                          //OpenGL
34443: 813 uPSI_StdVCL;                                 //VCL
34444: 814 unit CurvesAndSurfaces;                      //OpenGL
34445: 815 uPSI_DataAwareMain;                          //AsyncFree
34446: 816 uPSI_TabNotBk;                               //VCL
34447: 817 uPSI_udwsfiler;                              //mX4
34448: 818 uPSI_synaip;                                 //Synapse!
34449: 819 uPSI_synacode;                               //Synapse
34450: 820 uPSI_synachar;                               //Synapse
34451: 821 uPSI_synamisc;                               //Synapse
34452: 822 uPSI_synaser;                                //Synapse
34453: 823 uPSI_synaicnv;                               //Synapse
34454: 824 uPSI_tlntsend;                               //Synapse
34455: 825 uPSI_pingsend;                               //Synapse
34456: 826 uPSI_blcksock;                               //Synapse
34457: 827 uPSI_asn1util;                               //Synapse
34458: 828 uPSI_dnssend;                                //Synapse
34459: 829 uPSI_clamsend;                               //Synapse
34460: 830 uPSI_ldapsend;                               //Synapse
34461: 831 uPSI_mimemess;                               //Synapse
34462: 832 uPSI_slogsend;                               //Synapse
34463: 833 uPSI_mimepart;                               //Synapse
34464: 834 uPSI_mimeinln;                               //Synapse
34465: 835 uPSI_ftpsend,                                //Synapse
34466: 836 uPSI_ftptsend;                               //Synapse
34467: 837 uPSI_httpsend;                               //Synapse
34468: 838 uPSI_sntpsend;                               //Synapse
34469: 839 uPSI_smtpsend;                               //Synapse
34470: 840 uPSI_snmpsend;                               //Synapse
34471: 841 uPSI_imapsend;                               //Synapse
34472: 842 uPSI_pop3send;                               //Synapse
```

```
34473: 843 uPSI_nntpsend;                          //Synapse
34474: 844 uPSI_ssl_cryptlib;                      //Synapse
34475: 845 uPSI_ssl_openssl;                       //Synapse
34476: 846 uPSI_synhttp_daemon;                    //Synapse
34477: 847 uPSI_NetWork;                           //mX4
34478: 848 uPSI_PingThread;                        //Synapse
34479: 849 uPSI_JvThreadTimer;                     //JCL
34480: 850 unit uPSI_wwSystem;                     //InfoPower
34481: 851 unit uPSI_IdComponent;                  //Indy
34482: 852 unit uPSI_IdIOHandlerThrottle;          //Indy
34483: 853 unit uPSI_Themes;                       //VCL
34484: 854 unit uPSI_StdStyleActnCtrls;            //VCL
34485: 855 unit uPSI_UDDIHelper;                   //VCL
34486: 856 unit uPSI_IdIMAP4Server;                //Indy
34487: 857 uPSI_VariantSymbolTable,                //VCL //3.9.9.92
34488: 858 uPSI_udf_glob,                          //mX4
34489: 859 uPSI_TabGrid,                           //VCL
34490: 860 uPSI_JsDBTreeView,                      //mX4
34491: 861 uPSI_JsSendMail,                        //mX4
34492: 862 uPSI_dbTvRecordList,                    //mX4
34493: 863 uPSI_TreeVwEx,                          //mX4
34494: 864 uPSI_ECDataLink,                        //mX4
34495: 865 uPSI_dbTree,                            //mX4
34496: 866 uPSI_dbTreeCBox,                        //mX4
34497: 867 unit uPSI_Debug; //TfrmDebug            //mX4
34498: 868 uPSI_TimeFncs;                          //mX4
34499: 869 uPSI_FileIntf,                          //VCL
34500: 870 uPSI_SockTransport,                     //RTL
34501: 871 unit uPSI_WinInet;                      //RTL
34502: 872 unit uPSI_Wwstr;                        //mX4
34503: 873 uPSI_DBLookup,                          //VCL
34504: 874 uPSI_Hotspot,                           //mX4
34505: 875 uPSI_HList; //History List              //mX4
34506: 876 unit uPSI_DrTable;                      //VCL
34507: 877 uPSI_TConnect,                          //VCL
34508: 878 uPSI_DataBkr,                           //VCL
34509: 879 uPSI_HTTPIntr;                          //VCL
34510: 880 unit uPSI_Mathbox;                      //mX4
34511: 881 uPSI_cyIndy,                            //cY
34512: 882 uPSI_cySysUtils,                        //cY
34513: 883 uPSI_cyWinUtils,                        //cY
34514: 884 uPSI_cyStrUtils,                        //cY
34515: 885 uPSI_cyObjUtils,                        //cY
34516: 886 uPSI_cyDateUtils,                       //cY
34517: 887 uPSI_cyBDE,                             //cY
34518: 888 uPSI_cyClasses,                         //cY
34519: 889 uPSI_cyGraphics,  //3.9.9.94_2          //cY
34520: 890 unit uPSI_cyTypes;                      //cY
34521: 891 uPSI_JvDateTimePicker,                  //JCL
34522: 892 uPSI_JvCreateProcess,                   //JCL
34523: 893 uPSI_JvEasterEgg,                       //JCL
34524: 894 uPSI_WinSvc,  //3.9.9.94_3              //VCL
34525: 895 uPSI_SvcMgr                             //VCL
34526: 896 unit uPSI_JvPickDate;                   //JCL
34527: 897 unit uPSI_JvNotify;                     //JCL
34528: 898 uPSI_JvStrHlder                         //JCL
34529: 899 unit uPSI_JclNTFS2;                     //JCL
34530: 900 uPSI_Jcl8087 //math coprocessor         //JCL
34531: 901 uPSI_JvAddPrinter                       //JCL
34532: 902 uPSI_JvCabFile                          //JCL
34533: 903 uPSI_JvDataEmbedded;                    //JCL
34534: 904 unit uPSI_U_HexView;                    //mX4
34535: 905 uPSI_UWavein4,                          //mX4
34536: 906 uPSI_AMixer,                            //mX4
34537: 907 uPSI_JvaScrollText,                     //mX4
34538: 908 uPSI_JvArrow,                           //mX4
34539: 909 unit uPSI_UrlMon;                       //mX4
34540: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
34541: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
34542: 912 unit uPSI_DFFUtils;                     //DFF
34543: 913 unit uPSI_MathsLib;                     //DFF
34544: 914 uPSI_UIntList;                          //DFF
34545: 915 uPSI_UGetParens;                        //DFF
34546: 916 unit uPSI_UGeometry;                    //DFF
34547: 917 unit uPSI_UAstronomy;                   //DFF
34548: 918 unit uPSI_UCardComponentV2;             //DFF
34549: 919 unit uPSI_UTGraphSearch;                //DFF
34550: 920 unit uPSI_UParser10;                    //DFF
34551: 921 unit uPSI_cyIEUtils;                    //cY
34552: 922 unit uPSI_UcomboV2;                     //DFF
34553: 923 uPSI_cyBaseComm,                        //cY
34554: 924 uPSI_cyAppInstances,                    //cY
34555: 925 uPSI_cyAttract,                         //cY
34556: 926 uPSI_cyDERUtils                         //cY
34557: 927 unit uPSI_cyDocER;                      //cY
34558: 928 unit uPSI_ODBC;                         //mX
34559: 929 unit uPSI_AssocExec;                    //mX
34560: 930 uPSI_cyBaseCommRoomConnector,           //cY
34561: 931 uPSI_cyCommRoomConnector,               //cY
```

```
34562: 932 uPSI_cyCommunicate,                          //cY
34563: 933 uPSI_cyImage;                                //cY
34564: 934 uPSI_cyBaseContainer                         //cY
34565: 935 uPSI_cyModalContainer,                       //cY
34566: 936 uPSI_cyFlyingContainer;                      //cY
34567: 937 uPSI_RegStr,                                 //VCL
34568: 938 uPSI_HtmlHelpViewer;                         //VCL
34569: 939 unit uPSI_cyIniForm                          //cY
34570: 940 unit uPSI_cyVirtualGrid;                     //cY
34571: 941 uPSI_Profiler,                               //DA
34572: 942 uPSI_BackgroundWorker,                       //DA
34573: 943 uPSI_WavePlay,                               //DA
34574: 944 uPSI_WaveTimer,                              //DA
34575: 945 uPSI_WaveUtils;                              //DA
34576: 946 uPSI_NamedPipes,                             //TB
34577: 947 uPSI_NamedPipeServer,                        //TB
34578: 948 unit uPSI_process,                           //TB
34579: 949 unit uPSI_DPUtils;                           //TB
34580: 950 unit uPSI_CommonTools;                       //TB
34581: 951 uPSI_DataSendToWeb,                          //TB
34582: 952 uPSI_StarCalc,                               //TB
34583: 953 uPSI_D2_XPVistaHelperU                       //TB
34584: 954 unit uPSI_NetTools                           //TB
34585: 955 unit uPSI_Pipes                              //TB
34586: 956 uPSI_ProcessUnit,                            //mX
34587: 957 uPSI_adGSM,                                  //mX
34588: 958 unit uPSI_BetterADODataSet;                  //mX
34589: 959 unit uPSI_AdSelCom; //FTT                    //mX
34590: 960 unit unit uPSI_dwsXPlatform;                 //DWS
34591: 961 uPSI_AdSocket;                               //mX Turbo Power
34592: 962 uPSI_AdPacket;                               //mX
34593: 963 uPSI_AdPort;                                 //mX
34594: 964 uPSI_PathFunc;                               //Inno
34595: 965 uPSI_CmnFunc;                               //Inno
34596: 966 uPSI_CmnFunc2; //Inno Setup                  //Inno
34597: 967 unit uPSI_BitmapImage;                       //mX4
34598: 968 unit uPSI_ImageGrabber;                      //mX4
34599: 969 uPSI_SecurityFunc,                           //Inno
34600: 970 uPSI_RedirFunc,                              //Inno
34601: 971 uPSI_FIFO, (MemoryStream)                    //mX4
34602: 972 uPSI_Int64Em,                                //Inno
34603: 973 unit uPSI_InstFunc;                          //Inno
34604: 974 unit uPSI_LibFusion;                         //Inno
34605: 975 uPSI_SimpleExpression;                       //Inno
34606: 976 uPSI_unitResourceDetails,                    //XN
34607: 977 uPSI_unitResFile,                            //XN
34608: 978 unit uPSI_simpleComport;                     //mX4
34609: 979 unit uPSI_AfViewershelpers;                  //Async
34610: 980 unit uPSI_Console;                           //mX4
34611: 981 unit uPSI_AnalogMeter;                       //TB
34612: 982 unit uPSI_XPrinter,                          //TB
34613: 983 unit uPSI_IniFiles;                          //VCL
34614: 984 unit uPSI_lazIniFiles;                       //FP
34615: 985 uPSI_testutils;                              //FP
34616: 986 uPSI_ToolsUnit; (DBTests)                    //FP
34617: 987 uPSI_fpcunit                                 //FP
34618: 988 uPSI_testdecorator;                          //FP
34619: 989 unit uPSI_fpcunittests;                      //FP
34620: 990 unit uPSI_cTCPBuffer;                        //Fundamentals 4
34621: 991 unit uPSI_Glut,                              //Open GL
34622: 992 uPSI_LEDBitmaps,                             //mX4
34623: 993 uPSI_FileClass,                              //Inno
34624: 994 uPSI_FileUtilsClass,                         //mX4
34625: 995 uPSI_ComPortInterface; //Kit                //mX4
34626: 996 unit uPSI_SwitchLed;                         //mX4
34627: 997 unit uPSI_cyDmmCanvas;                       //cY
34628: 998 uPSI_uColorFunctions;                        //DFF
34629: 999 uPSI_uSettings;                              //DFF
34630: 1000 uPSI_cyDebug.pas                            //cY
34631: 1001 uPSI_cyColorMatrix;                         //cY
34632: 1002 unit uPSI_cyCopyFiles;                      //cY
34633: 1003 unit uPSI_cySearchFiles;                    //cY
34634: 1004 unit uPSI_cyBaseMeasure;                    //cY
34635: 1005 unit uPSI_PJIStreams;                       //DD
34636: 1006 unit uPSI_cyRunTimeResize;                  //cY
34637: 1007 unit uPSI_jcontrolutils;                    //cY
34638: 1008 unit uPSI_kcMapViewer; (+GEONames)          //kc
34639: 1009 unit uPSI_kcMapViewerDESynapse;             //kc
34640: 1010 unit uPSI_cparserutils; (+GIS_SysUtils)     //kc
34641: 1011 unit uPSI_LedNumber;                        //TurboPower
34642: 1012 unit uPSI_StStrL;                           //SysTools
34643: 1013 unit uPSI_indGnouMeter;                     //LAZ
34644: 1014 unit uPSI_Sensors;                          //LAZ
34645: 1015 unit uPSI_pwmain;  //cgi of powtils         //Pow
34646: 1016 unit uPSI_HTMLUtil;                         //Pow
34647: 1017 unit uPSI_synwrap1; //httpsend              //Pow
34648: 1018 unit StreamWrap1                            //Pow
34649: 1019 unit uPSI_pwmain;                           //Pow
34650: 1020 unit pwtypes                                //Pow
```

```
34651: 1021 uPSI_W32VersionInfo                        //LAZ
34652: 1022 unit uPSI_IpAnim;                          //LAZ
34653: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
34654: 1024 unit uPSI_LrtPoTools;                      //LAZ
34655: 1025 unit uPSI_Laz_DOM;                         //LAZ
34656: 1026 unit uPSI_hhAvComp;                        //LAZ
34657: 1027 unit uPSI_GPS2;                            //mX4
34658: 1028 unit uPSI_GPS;                             //mX4
34659: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo//mX4
34660: 1030 unit uPSI_NMEA;     // GPS                 //mX4
34661: 1031 unit uPSI_ScreenThreeDLab;                 //mX4
34662: 1032 unit uPSI_Spin;                            //VCL
34663: 1033 unit uPSI_DynaZip;                         //mX4
34664: 1034 unit uPSI_clockExpert;                     //TB
34665: 1035 unit debugLn                               //mX4
34666: 1036 uPSI_SortUtils;                            //Jcl
34667: 1037 uPSI_BitmapConversion;                     //Jcl
34668: 1038 unit uPSI_JclTD32;                         //Jcl
34669: 1039 unit uPSI_ZDbcUtils;                       //Zeos
34670: 1040 unit uPSI_ZScriptParser;                   //Zeos
34671: 1041 uPSI_JvIni,                                //JCL
34672: 1042 uPSI_JvFtpGrabber;                         //JCL
34673: 1043 unit uPSI_NeuralNetwork;                   //OCL
34674: 1044 unit uPSI_StExpr;                          //SysTools
34675: 1045 unit uPSI_GR32_Geometry;                   //GR32
34676: 1046 unit uPSI_GR32_Containers;                 //GR32
34677: 1047 unit uPSI_GR32_Backends_VCL,               //GR32
34678: 1048 unit uPSI_StSaturn;//Venus+Mercury+Mars++ //SysTools
34679: 1049 unit uPSI_JclParseUses;                    //JCL
34680: 1050 unit uPSI_JvFinalize;                      //JCL
34681: 1051 unit uPSI_panUnit1;                        //GLScene
34682: 1052 unit uPSI_DD83u1;  //Arduino Tester       //mX4
34683: 1053 unit uPSI_BigIni                           //Hinzen
34684: 1054 unit uPSI_ShellCtrls;                      //VCL
34685: 1055 unit uPSI_fmath;                           //FMath
34686: 1056 unit uPSI_fComp;                           //FMath
34687: 1057 unit uPSI_HighResTimer;                    //Lauer
34688: 1058 unit uconvMain; (Unit Converter)           //PS
34689: 1059 unit uPSI_uconvMain;                       //PS
34690: 1060 unit uPSI_ParserUtils;                     //PS
34691: 1061 unit uPSI_uPSUtils;                        //PS
34692: 1062 unit uPSI_ParserU;                         //PS
34693: 1063 unit uPSI_TypInfo;                         //VCL
34694: 1064 unit uPSI_ServiceMgr;                      //mX
34695: 1065 unit uPSI_UDict;                           //DFF
34696: 1066 unit uPSI_ubigFloatV3;                     //DFF
34697: 1067 unit uPSI_UBigIntsV4;                      //DFF
34698: 1068 unit uPSI_ServiceMgr2;                     //mX
34699: 1069 unit uPSI_UP10Build;                       //PS
34700: 1070 unit uPSI_UParser10;                       //PS
34701: 1071 unit uPSI_IdModBusServer;                  //MB
34702: 1072 unit uPSI_IdModBusClient;                  //MB
34703: 1073 unit uPSI_ColorGrd;                        //VCL
34704: 1074 unit uPSI_DirOutln;                        //VCL
34705: 1075 unit uPSI_Gauges;                          //VCL
34706: 1076 unit uPSI_CustomizeDlg;                    //VCL
34707: 1077 unit uPSI_ActnMan;                         //VCL
34708: 1078 unit uPSI_CollPanl;                        //VCL
34709: 1079 unit uPSI_Calendar2;                       //VCL
34710: 1080 unit uPSI_IBCtrls;                         //VCL
34711: 1081 unit uPSI_IdStackWindows;                  //Indy
34712: 1082 unit uPSI_CTSVendorUtils;                  //DBX
34713: 1083 unit uPSI_VendorTestFramework;             //DBX
34714: 1084 unit uPSI_TInterval;                       //mX4
34715: 1085 unit uPSI_JvAnimate                        //JCL
34716: 1086 unit uPSI_DBXCharDecoder;                  //DBX
34717: 1087 unit uPSI_JvDBLists;                       //JCL
34718: 1088 unit uPSI_JvFileInfo;                      //JCL
34719: 1089 unit uPSI_SOAPConn;                        //VCL
34720: 1090 unit uPSI_SOAPLinked;                      //VCL
34721: 1091 unit uPSI_XSBuiltIns;                      //VCL
34722: 1092 unit uPSI_JvgDigits;                       //JCL
34723: 1093 unit uPSI_JvDesignUtils;
34724: 1094 unit uPSI_JvgCrossTable;
34725: 1095 unit uPSI_JvgReport;
34726: 1096 unit uPSI_JvDBRichEdit;
34727: 1097 unit uPSI_JvWinHelp;
34728: 1098 unit uPSI_WaveConverter;
34729: 1099 unit uPSI_ACMConvertor;
34730: 1100 unit XSBuiltIns_Routines
34731: 1101 unit uPSI_ComObjOleDB_utils.pas
34732: 1102 unit uPSI_SMScript;
34733: 1103 unit uPSI_CompFileIo;
34734: 1104 unit uPSI_SynHighlighterGeneral;
34735: 1105 unit uPSI_geometry2;
34736: 1106 unit uPSI_MConnect;
34737: 1107 unit uPSI_ObjBrkr;
34738: 1108 unit uPSI_uMultiStr;
34739: 1109 unit uPSI_WinAPI.pas;
```

```
34740: 1110 unit uPSI_JvAVICapture;
34741: 1111 unit uPSI_JvExceptionForm;
34742: 1112 unit uPSI_JvConnectNetwork;
34743: 1113 unit uPSI_MTMainForm;
34744: 1114 unit uPSI_DdeMan;
34745: 1115 unit uPSI_DIUtils;
34746: 1116 unit uPSI_gnugettext;
34747: 1117 unit uPSI_Xmlxform;
34748: 1118 unit uPSI_SvrHTTPIndy;
34749: 1119 unit uPSI_CPortTrmSet;
34750: 1120 unit SvrLog; ----------------------------------------------------------------
34751: 1121 unit uPSI_IndySockTransport.pas (+IdHTTPHeaderInfo) //mX4
34752: 1122 unit uPSI_HTTPProd.pas
34753: 1123 unit uPSI_CppParser.pas
34754: 1124 unit uPSI_SynHighlighterCpp.pas
34755: 1125 unit uPSI_CodeCompletion.pas
34756: 1126 unit uPSI_U_IntList2.pas
34757: 1127 unit uPSI_SockHTTP.pas
34758: 1128 uPSI_SockAppNotify.pas
34759: 1129 uPSI_NSToIS.pas
34760: 1130 unit uPSI_DBOleCtl.pas
34761: 1131 unit uPSI_xercesxmldom;
34762: 1132 unit uPSI_xmldom;
34763: 1133 unit uPSI_Midas;
34764: 1134 unit uPSI_JclExprEval;
34765: 1135 uPSI_Gameboard;
34766: 1136 unit uPSI_ExtUtil;
34767: 1137 unit uPSI_FCGIApp;
34768: 1138 unit uPSI_ExtPascal;
34769: 1139 unit uPSI_PersistSettings;
34770: 1140 IdHTTPHeaderInfo.pas
34771: 1141 uPSI_SynEditAutoComplete;
34772: 1142 uPSI_SynEditTextBuffer.pas
34773: 1143 unit uPSI_JclPCRE;
34774: 1144 unit uPSI_ZConnection;
34775: 1145 unit uPSI_ZSequence;
34776: 1146 unit uPSI_ChessPrg;
34777: 1147 unit uPSI_ChessBrd;
34778: 1148 unit uPSI_Graph3D;
34779: 1149 uPSI_SysInfoCtrls2.pas
34780: 1150 unit uPSI_RegUtils;
34781: 1151 unit uPSI_VariantRtn;
34782: 1152 uPSI_StdFuncs,
34783: 1153 unit uPSI_SqlTxtRtns;
34784: 1154 unit uPSI_BSpectrum;
34785: 1155 unit IPAddressControl;
34786: 1156 unit uPSI_Paradox;
34787: 1157 unit uPSI_Environ;
34788: 1158 uPSI_GraphicsPrimitivesLibrary;
34789: 1159 uPSI_DrawFigures,
34790: 1160 unit uPSI_synadbg;
34791: 1161 unit uPSI_BitStream;
34792: 1162 unit uPSI_xrtl_util_FileVersion;
34793: 1163 uPSI_XmlRpcCommon,
34794: 1164 unit uPSI_XmlRpcClient;
34795: 1165 unit uPSI_XmlRpcTypes;
34796: 1166 unit uPSI_XmlRpcServer;
34797: 1167 unit uPSI_SynAutoIndent;
34798: 1168 unit uPSI_synafpc;
34799: 1169 unit uPSI_RxNotify;
34800: 1170 unit uPSI_SynAutoCorrect;
34801: 1171 unit uPSI_rxOle2Auto;
34802: 1172 unit uPSI_Spring_Utilsmx;
34803: 1173 unit uPSI_ulogifit;
34804: 1174 unit uPSI_HarmFade;
34805: 1175 unit uPSI_SynCompletionProposal;
34806: 1176 unit uPSI_rxAniFile;
34807: 1177 uPSI_ulinfit,
34808: 1178 uPSI_usvdfit;
34809: 1179 unit uPSI_JclStringLists;
34810: 1180 unit uPSI_ZLib;
34811: 1181 unit uPSI_MaxTokenizers;  //WANT
34812: 1182 unit uPSI_MaxStrUtils;
34813: 1183 unit uPSI_MaxXMLUtils;
34814: 1184 unit uPSI_MaxUtils;
34815: 1185 unit uPSI_VListBox;
34816: 1186 unit uPSI_MaxDOM;
34817: 1187 unit uPSI_MaxDOMDictionary;
34818: 1188 unit uPSI_MaxDOMDictionary_Routines;
34819: 1189 unit uPSI_cASN1;
34820: 1190 unit uPSI_cX509Certificate;
34821: 1191 unit uPSI_uCiaXml;
34822: 1192 unit uPSI_StringsW;
34823: 1193 unit uPSI_FileStreamW;  //WideString D7X
34824: 1194 unit Drawingutils;
34825: 1195 unit uPSI_InetUtilsUnified;
34826: 1196 unit uPSI_FileMask;
34827: 1197 unit uPSI_StrConv;
34828: 1198 unit uPSI_Simpat;
```

```
34829: 1199 unit uPSI_Tooltips.pas
34830: 1200 unit uPSI_StringGridLibrary.pas
34831: 1201 unit uPSI_ChronCheck.pas
34832: 1202 unit uPSI_REXX.pas
34833: 1203 uPSI_SysImg.pas
34834: 1204 unit uPSI_Tokens;
34835: 1205 unit uPSI_KFunctions;
34836: 1206 unit uPSI_KMessageBox;
34837: 1207 unit uPSI_CPUSpeed.pas
34838: 1208 uPSI_RoboTracker.pas
34839: 1209 unit uPSI_NamedPipesImpl.pas
34840: 1210 unit uPSI_KLog.pas
34841: 1211 unit uPSI_NamedPipeThreads.pas
34842: 1212 unit uPSI_MapFiles.pas //map stream of memory-mapped files
34843: 1213 unit uPSI_BKPwdGen,     //Password Generator
34844: 1214 unit uPSI_Kronos,  // big chrono date time library
34845: 1215 unit uPSI_TokenLibrary2;
34846: 1216 uPSI_KDialogs,
34847: 1217 uPSI_Numedit,
34848: 1218 unit uPSI_StSystem2;
34849: 1219 unit uPSI_KGraphics;
34850: 1220 uPSI_KGraphics_functions;
34851: 1221 uPSI_umaxPipes.pas
34852: 1222 unit uPSI_KControls;
34853: 1223 unit SysUtils_max2;
34854: 1224 uPSI_IdAntiFreeze.pas
34855: 1225 uPSI_IdLogStream.pas
34856: 1226 unit uPSI_IdThreadSafe;
34857: 1227 unit uPSI_IdThreadMgr;
34858: 1228 unit uPSI_IdAuthentication;
34859: 1229 unit uPSI_IdAuthenticationManager;
34860: 1230 uPSI_OverbyteIcsConApp
34861: 1231 unit uPSI_KMemo;
34862: 1232 unit uPSI_OverbyteIcsTicks64;
34863: 1233 unit uPSI_OverbyteIcsSha1.pas
34864: 1234 unit uPSI_KEditCommon.pas
34865: 1235 unit uPSI_UtilsMax4.pas
34866: 1236 unit uPSI_IdNNTPServer;
34867: 1237 unit uPSI_UWANTUtils;
34868: 1238 unit uPSI_UtilsMax5.pas;
34869: 1239 unit uPSI_OverbyteIcsAsn1Utils;
34870: 1240 unit uPSI_IdHTTPHeaderInfo; //mX4 response headers
34871: 1241 uPSI_wmiserv.pas {uPSI_SimpleSFTP.pas}
34872: 1242 uPSI_WbemScripting_TLB.pas
34873: 1243 unit uPSI_uJSON;
34874: 1244 uPSI_RegSvrUtils.pas
34875: 1245 unit uPSI_osFileUtil;
34876: 1246 unit uPSI_SHDocVw; //TWebbrowser
34877: 1247 unit uPSI_SHDocVw_TLB
34878: 1248 uPSC_classes.pas V2
34879: 1249 uPSR_classes.pas V2
34880: 1250 uPSI_U_Oscilloscope4_2
34881: 1251 uPSI_xutils.pas
34882: 1252 uPSI_ietf.pas
34883: 1253 uPSI_iso3166.pas
34884: 1254 uPSI_dateutil_real.pas  //Optima ISO 8601
34885: 1255 uPSI_dateext4.pas
34886: 1256 uPSI_locale.pas         //ISO Formater & Tests
34887: 1257 file charset.inc  //IANA Registered character sets
34888: 1258 unit uPSI_Strings;
34889: 1259 unit uPSI_crc_checks;  //ISO 3309 and ITU-T-V42
34890: 1260 unit uPSI_extDOS;
34891: 1261 uPSI_uBild;  //Steganography
34892: 1262 unit uPSI_SimpleTCP;
34893: 1263 unit uPSI_IdFTPList;
34894: 1264 uPSI_uTPLb_CryptographicLibrary.pas
34895: 1265 uPSI_uTPLb_RSA_Engine.pas
34896: 1266 uPSI_cHugeInt.pas
34897: 1267 unit uPSI_SimpleTCPServer;
34898: 1268 unit uPSI_xBase.pas
34899: 1269 unit uPSI_ImageHistogram.pas
34900: 1270 unit uP_PersistSettings2;
34901: 1271 uPSI_WDosDrivers.pas
34902: 1272 unit uPSI_cCipherRSA;
34903: 1273 uPSI_CromisStreams, (TStreamStorage)
34904: 1274 unit uPSI_Streams,
34905: 1275 uPSI_BitStream,
34906: 1276 uPSI_UJSONFunctions.pas
34907: 1277 uPSI_uTPLb_BinaryUtils.pas
34908: 1278 unit uPSI_USha256.pas  //PascalCoin Blockchain
34909: 1279 uPSI_uTPLb_HashDsc.pas
34910: 1280 uPSI_uTPLb_Hash.pas
34911: 1281 SIRegister_Series(X); (uPSI_Series) //4.2.6.10
34912: 1282 unit uPSI_UTime; (UTime); uPSI_mimeinln2;
34913: 1283 uPSI_uTPLb_StreamCipher.pas
34914: 1284 uPSI_uTPLb_BlockCipher.pas
34915: 1285 uPSI_uTPLb_Asymetric.pas
34916: 1286 uPSI_uTPLb_CodecIntf.pas
34917: 1287 uPSI_uTPLb_Codec.pas
```

```
34918: 1288 uPSI_ADOInt.pas
34919: 1289 uPSI_MidasCon.pas
34920: 1290 uPSI_XMLDoc.pas
34921: 1291 uPSI_XMLIntf.pas
34922: 1292 uPSI_ProxyUtils.pas
34923: 1293 unit uPSI_maxXMLUtils2;
34924: 1294 unit_StDict_Routines(S: TPSExec);
34925: 1295 unit uPSI_Hashes2
34926: 1296 unit uPSI_IdCoderHeader;
34927: 1297 unit uPSI_BackgroundWorker2
34928: 1298 uPSI_uMRU.pas
34929: 1299 unit FANN.pas
34930: 1300 unit uPSI_FannNetwork.pas
34931: 1301 unit uPSI_RTLDateTimeplus.pas
34932: 1302 uPSI_ULog.pas
34933: 1303 uPSI_UThread.pas
34934: 1304 uPSI_UTCPIP.pas
34935: 1305 Synapse_OpenSSLv11
34936: 1306 PascalCoin configuration config.inc
34937: 1307 unit uPSI_statmach,   {StateMachine}        //46310
34938: 1308 uPSI_uTPLb_RSA_Primitives,
34939: 1309 unit uPSI_UMatrix,
34940: 1310 uPSI_DXUtil,
34941: 1311 uPSI_crlfParser,
34942: 1312 unit uPSI_DCPbase64;
34943: 1313 unit uPSI_FlyFilesUtils;
34944: 1314 uPSI_PJConsoleApp.pas
34945: 1315 uPSI_PJStreamWrapper.pas
34946: 1316 uPSI_LatLonDist,    //DFF
34947: 1317 uPSI_cHash2.pas     //Fundamentals
34948: 1318 uPSI_ZLib2.pas
34949: 1319 unit uPSI_commDriver
34950: 1320 unit uPSI_PXLNetComs.pas   //PXL
34951: 1321 unit uPSI_PXLTiming.pas    //PXL
34952: 1322 uPSI_Odometer.pas
34953: 1323 unit uPSI_UIntList2;
34954: 1324 uPSI_UIntegerpartition.pas
34955: 1325 unit uPSI_idPHPRunner.pas  //prepare for PHP4D
34956: 1326 unit uPSI_idCGIRunner.pas
34957: 1327 uPSI_DrBobCGI,             //4.7.1.20
34958: 1328 uPSI_OverbyteIcsLogger, / / n  e w   c o u n t   1 2 2 8 - - > 1 3 2 8
34959: 1329 uPSI_OverbyteIcsNntpCli,  testset
34960: 1330 uPSI_OverbyteIcsCharsetUtils,
34961: 1331 uPSI_OverbyteIcsMimeUtils,
34962: 1332 uPSI_OverbyteIcsUrl(CL: TPSPascalCompiler);
34963: 1333 uPSI_uWebSocket.pas
34964: 1334 uPSI_KhFunction.pas
34965: 1335 uPSI_ALOpenOffice.pas
34966: 1336 unit uPSI_ALLibPhoneNumber
34967: 1337 unit uPSI_ALPhpRunner2;
34968: 1338 unit uPSI_ALWebSpider2;
34969: 1339 unit uPSI_ALFcnHTML2; // RunJavaScript2
34970: 1340 unit uPSI_ALExecute2.pas
34971: 1341 uPSI_ALIsapiHTTP.pas
34972: 1342 uPSI_ALOpenOffice_Routines
34973: 1343 unit uPSI_uUsb;
34974: 1344 uPSI_uWebcam.pas
34975: 1345 uPSI_PersistSettings.pas //fixing & refactoring
34976: 1346 uPSI_uTPLb_MemoryStreamPool.pas
34977: 1347 uPSI_uTPLb_Signatory.pas
34978: 1348 unit uPSI_uTPLb_Constants.pas  //TurboPower
34979: 1349 uPSI_uTPLb_Random.pas
34980: 1350 unit uPSI_uTPLb_PointerArithmetic;
34981: 1351 unit uPSI_EwbCoreTools.pas
34982: 1352 unit uPSI_EwbUrl.pas
34983: 1353 unit uPSI_SendMail_For_Ewb.pas
34984: 1354 unit uPSI_MaskEdit.pas    FCL
34985: 1355 unit uPSI_SimpleRSSTypes; BlueHippo
34986: 1356 unit uPSI_SimpleRSS;      BlueHippo
34987: 1357 unit uPSI_psULib.pas      Prometheus
34988: 1358 unit uPSI_psUFinancial;   Prometheus
34989: 1359 uPSI_PsAPI_2.pas          mX4
34990: 1360 uPSI_PersistSettings_2    mX4
34991: 1361 uPSI_rfc1213util2.pas     IP
34992: 1362 uPSI_JTools.pas           JCL
34993: 1363 unit uPSI_neuralbit.pas   CAI
34994: 1364 unit uPSI_neuralab.pas    CAI
34995: 1365 unit uPSI_winsvc2.pas     TEK
34996: 1366 unit uPSI_wmiserv2.pas    TEK
34997: 1367 uPSI_neuralcache.pas      CAI
34998: 1368 uPSI_neuralbyteprediction CAI
34999: 1369 unit uPSI_USolarSystem;   glscene.org
35000: 1370 uPSI_USearchAnagrams.pas  DFF
35001: 1371 uPSI_JsonsUtilsEx.pas     Randolph
35002: 1372 unit uPSI_Jsons.pas       Randolph
35003: 1373 unit uPSI_HashUnit;       DFF
35004: 1374 uPSI_U_Invertedtext.pas   DFF
35005: 1375 unit uPSI_Bricks;         Dendron
35006: 1376 unit uPSI_lifeblocks.pas  Dendron
```

```
35007: 1377 unit uPSI_SystemsDiagram.pas  Dendron
35008: 1378 unit uPSI_qsFoundation.pas    Dendron
35009: 1379 uPSI_JclStringLists2          JCL
35010: 1380 uPSI_cInternetUtils2          FLC
35011: 1381 uPSI_cWindows.pas            FLC
35012: 1382 uPSI_flcSysUtils.pas  +TBytes utils
35013: 1383 unit uPSI_RotImg.pas          DA
35014: 1384 uPSI_SimpleImageLoader.pas    LAZ
35015: 1385 uPSI_HSLUtils.pas             LAZ
35016: 1386 uPSI_GraphicsMathLibrary.pas  EF
35017: 1387 unit uPSI_umodels.pas         DMath
35018: 1388 uPSI_flcStatistics.pas        FLC5
35019: 1389 uPSI_flcMaths.pas             FLC5
35020: 1390 uPSI_flcCharSet.pas           FLC5
35021: 1391 uPSI_flcBits32.pas
35022: 1392 uPSI_flcTimers.pas
35023: 1393 uPSI_cBlaiseParserLexer.pas
35024: 1394 uPSI_flcRational.pas
35025: 1395 uPSI_flcComplex.pas
35026: 1396 unit uPSI_flcMatrix (uPSI_flcVectors.pas)
35027: 1397 unit uPSI_flcStringBuilder.pas
35028: 1398 unit PJResFile_Routines;
35029: 1399 uPSI_flcASCII.pas
35030: 1400 uPSI_flcStringPatternMatcher;
35031: 1401 unit uPSI_flcUnicodeChar.pas
35032: 1402 unit uXmlDates.pas;
35033: 1403 unit uPSI_SemaphorGrids;
35034: 1404 unit uXmlDates2;
35035: 1405 unit uPSI_JclTimeZones;
35036: 1406 unit uPSI_XmlDocRssParser.pas
35037: 1407 unit uPSI_RssParser.pas
35038: 1408 uPSI_SimpleParserRSS.pas
35039: 1409 unit uPSI_SimpleRSSUtils;
35040: 1410 unit uPSI_RssModel;     _BlueHippo
35041: 1411 unit uPSI_StrUtil;      _FIBPlus
35042: 1412 unit uPSI_TAChartUtils; _TEE
35043: 1413 unit uPSI_PythonEngine.pas _P4D_Beta
35044: 1414 unit uPSI_VclPythonGUIInputOutput;
35045: 1415 unit uPSI_VarPyth;
35046: 1416 unit JclUsesUtils;
35047: 1417 unit uPSI_cParameters;
35048: 1418 unit uPSI_WDCCMisc; (uPSI_cFileTemplates);
35049: 1419 uPSI_WDCCOleVariantEnum.pas
35050: 1420 unit uPSI_WDCCWinInet.pas  _WDCC
35051: 1421 uPSI_PythonVersions.pas    _P4D_
35052: 1422 unit uPSI_PythonAction.pas _P4D_
35053: 1423 uPSI_SingleList.pas
35054: 1424 unit uPSI_AdMeter.pas; Async Professional
35055: 1425 unit uPSI_neuralplanbuilder; CAI
35056: 1426 unit uPSI_neuralvolume.pas; CAI
35057: 1427 unit uPSI_neuralvolumev.pas; CAI
35058: 1428 unit uPSI_DoubleList4;  CapJack
35059: 1429 unit uPSI_ByteListClass;  CapJack
35060: 1430 unit uPSI_flcVectors4; FLC5
35061: 1431 unit uPSI_flcMatrix4;  FLC5
35062: 1432 uPSI_CurlHttpCodes.pas
35063: 1433 unit uPSI_NeuralNetwork.pas; CAI
35064: 1434 unit unit uPSI_neuralfit; CAI
35065: 1435 unit uPSI_neuraldatasets; CAI
35066: 1436 unit uPSI_neuraldatasetsv.pas CAI
35067: 1437 uPSI_flcFloats.pas FLC5
35068: 1438 unit UBigIntsForFloatV4.pas DFF
35069: 1439 unit uPSI_CustApp.pas Pas2js
35070: 1440 unit uPSI_NeuralNetworkCAI.pas CAI
35071: 1441 unit uPSI_neuralgeneric.pas; CAI
35072: 1442 unit uPSI_neuralthread.pas; CAI
35073: 1443 unit uPSI_uSysTools; TuO
35074: 1444 unit upsi_neuralsets; mX4
35075: 1445 unit uPSI_uWinNT.pas mX4
35076: 1446 unit uPSI_URungeKutta4.pas ICS
35077: 1447 unit uPSI_UrlConIcs.pas ICS
35078: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
35079: 1449 unit uPSI_Numedit2 mX4
35080: 1450 unit uPSI_PsAPI_3.pas mX4
35081: 1451 unit uPSI_SeSHA256.pas
35082: 1452 unit IdHashMessageDigest_max2;
35083: 1453 unit uPSI_BlocksUnit.pas
35084: 1454 unit uPSI_DelticsCommandLine.pas
35085: 1455 unit uPSI_DelticsStrUtils;
35086: 1456 unit uPSI_DelticsBitField;
35087: 1457 unit uPSI_DelticsSysUtils;
35088: 1458 unit uPSI_ALIniFiles2.pas
35089: 1459 unit uPSI_StarCalc2.pas
35090: 1460 unit uPSI_IdHashMessageDigest2.pas
35091: 1461 unit uPSI_U_Splines;
35092: 1462 unit uPSI_U_CoasterB.pas;
35093: 1463 U_SpringMass2.pas
35094: 1464 uPSI_MARSCoreUtils;
35095: 1465 unit uPSI_clJsonParser.pas
```

```
35096: 1466 unit uPSI_SynHighlighterPython.pas
35097: 1467 unit uPSI_DudsCommonDelphi;
35098: 1468 unit uPSI_AINNNeuron;
35099: 1469 unit uPSI_PJConsoleApp2;  Console Application Runner
35100: 1470 unit uPSI_PJPipeFilters2; CAR
35101: 1471 unit uPSI_uHTMLBuilder;
35102: 1472 unit uPSI_PJPipe2;
35103: 1473 uPSI_WinApiDownload.pas
35104: 1474 uPSI_pxQRcode,     //
35105: 1475 unit uPSI_neuralplanbuilder2
35106: 1476 unit uPSI_DelphiZXingQRCode;
35107: 1477 unit uPSI_RestJsonUtils;
35108: 1478 unit UtilsTimeCode;
35109: 1479 unit uPSC_classes2.pas; //TList
35110: 1480 unit uPSC_std2.pas
35111: 1481 unit uPSI_TBytes.pas
35112: 1482 unit uPSI_DelticsSysUtils2.pas
35113: 1483 unit uPSI_maxIniFiles.pas
35114: 1484 unit uROPSImports.pas
35115: 1485 unit uROPSServerLink.pas
35116: 1486 unit KLibVC_Redist.pas;
35117: 1487 unit HTTPApp2.pas;
35118: 1488 unit uPSI_XCollection2;
35119: 1489 unit uPSI_KLibWindows;
35120: 1490 unit KlibConstants;
35121: 1491 unit uPSI_AzuliaUtils.pas
35122: 1492 unit uPSI_ALHttpClient2;
35123: 1493 unit uPSI_ALWininetHttpClient2;
35124: 1494 unit uPSI_UtilsMax41.pas
35125: 1495 unit uPSI_JclSysUtils1;
35126: 1496 unit uPSI_RestUtils;
35127: 1497 unit uPSI_TeEngine2.pas
35128: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
35129: 1499 unit uPSI_PSResources.pas
35130: 1500 unit uPSI_TeCanvas2_1.pas
35131: 1501 unit uPSI_DataSetConverter4DUtil;
35132: 1502 unit uPSI_neuralfit2.pas;
35133: 1503 unit uPSI_SynCrtSock.pas
35134: 1504 uPSI_RunElevatedSupport.pas
35135: 1505 unit synTHttpRequest.pas;
35136: 1506 unit uPSI_VelthuisFloatUtils.pas
35137: 1507 unit HttpConnection.pas
35138: 1508 unit uPSI_HttpConnectionWinInet.pas
35139: 1509 unit UHexUtils.pas
35140: 1510 unit UExeFileType.pas
35141: 1511 unit uPSI_UConsoleApp.pas
35142: 1512 unit uPSI_CompilersURunner.pas
35143: 1513 unit uPSI_HttpConnection.pas
35144: 1514 unit uPSI_DataSetUtils.pas
35145: 1515 unit uPSI_HTTPSender.pas
35146:
35147: Amount of Functions: 21460
35148: Amount of Procedures: 12722
35149: Amount of Constructors: 2009
35150: Amount of Destructors: 14
35151: Totals of Calls: 36205
35152: SHA1: of 4.7.6.20 BF042A1DC1C479C706100BB83F8DEE351554F74B
35153: CRC32: 4A7231F8:  34057496 bytes
35154: ▯▯▯ mX4 executed: 15/12/2022 16:50:31  Runtime: 0:15:18.987  Memload: 41% use
35155: Total of Function Calls: 36205
35156: SHA1: 4.7.6.20 BF042A1DC1C479C706100BB83F8DEE351554F74B
35157:                bf042a1dc1c479c706100bb83f8dee351554f74b
35158: CRC32: 4A7231F8 32.4 MB (34,057,496 bytes)
35159: Compilation Timestamp 2022-12-15 14:45:59 UTC Signing time 15 Dec 2022 15:50:51
35160: Entry Point 24930856 - Contained Sections 10
35161: sha1: bf042a1dc1c479c706100bb83f8dee351554f74b
35162: sha256: 102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3
35163:
35164: Total of Function Calls: 35958
35165: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
35166: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
35167: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
35168: Entry Point 24783132 - Contained Sections 10
35169:
35170: ****************************************************************
35171: Release Notes maXbox 4.7.6.20 Oct 2022 mX476
35172: ****************************************************************
35173: Add 60 Units + 18 Tutorials
35174:
35175: 1441 unit uPSI_neuralgeneric.pas; CAI
35176: 1442 unit uPSI_neuralthread.pas; CAI
35177: 1443 unit uPSI_uSysTools; TuO
35178: 1444 unit upsi_neuralsets; mX4
35179: 1445 unit uPSI_uWinNT.pas mX4
35180: 1446 unit uPSI_URungeKutta4.pas ICS
35181: 1447 unit uPSI_UrlConIcs.pas ICS
35182: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
35183: 1449 unit uPSI_Numedit2 mX4
35184: 1450 unit uPSI_PsAPI_3.pas mX4
```

```
35185: 1451 unit uPSI_SeSHA256.pas
35186: 1452 unit IdHashMessageDigest_max2;
35187: 1453 unit uPSI_BlocksUnit.pas
35188: 1454 unit uPSI_DelticsCommandLine.pas
35189: 1455 unit uPSI_DelticsStrUtils;
35190: 1456 unit uPSI_DelticsBitField;
35191: 1457 unit uPSI_DelticsSysUtils;
35192: 1458 unit uPSI_ALIniFiles2.pas
35193: 1459 unit uPSI_StarCalc2.pas
35194: 1460 unit uPSI_IdHashMessageDigest2.pas
35195: 1461 unit uPSI_U_Splines;
35196: 1462 unit uPSI_U_CoasterB.pas;
35197: 1463 U_SpringMass2.pas
35198: 1464 uPSI_MARSCoreUtils;
35199: 1465 unit uPSI_clJsonParser.pas
35200: 1466 unit uPSI_SynHighlighterPython.pas
35201: 1467 unit uPSI_DudsCommonDelphi;
35202: 1468 unit uPSI_AINNNeuron;
35203: 1469 unit uPSI_PJConsoleApp2;
35204: 1470 unit uPSI_PJPipeFilters2;
35205: 1471 unit uPSI_uHTMLBuilder;
35206: 1472 unit uPSI_PJPipe2;
35207: 1473 uPSI_WinApiDownload,
35208: 1474 uPSI_pxQRcode,    //beta
35209: 1475 unit uPSI_neuralplanbuilder2
35210: 1476 unit uPSI_DelphiZXingQRCode;
35211: 1477 unit uPSI_RestJsonUtils;
35212: 1478 unit UtilsTimeCode;
35213: 1479 unit uPSC_classes2.pas; //TList
35214: 1480 unit uPSC_std2.pas
35215: 1481 unit uPSI_maxIniFiles.pas
35216: 1482 unit uROPSImports.pas
35217: 1483 unit uROPSServerLink.pas
35218: 1484 unit uPSI_KLibUtils;
35219: 1485 unit uPSI_PathFunc2;  //inno setup
35220: 1486 unit KLibVC_Redist.pas;
35221: 1487 unit HTTPApp2.pas;
35222: 1488 unit uPSI_XCollection2;
35223: 1489 unit uPSI_KLibWindows;
35224: 1490 unit KlibConstants;
35225: 1491 unit uPSI_AzuliaUtils.pas
35226: 1492 unit uPSI_ALHttpClient2;
35227: 1493 unit uPSI_ALWininetHttpClient2;
35228: 1494 unit uPSI_UtilsMax41.pas
35229: 1495 unit uPSI_JclSysUtils1;
35230: 1496 unit uPSI_RestUtils;
35231: 1497 unit uPSI_TeEngine2.pas
35232: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
35233: 1499 unit uPSI_PSResources.pas
35234: 1500 unit uPSI_TeCanvas2_1.pas
35235: 1501 unit uPSI_DataSetConverter4DUtil
35236:
35237: Total of Function Calls: 36130
35238: SHA1: 4.7.6.20 A71AA600D48CE895C76CAB68C72F28707C4CD90B
35239: CRC32: 5BD4B183 32.4 MB (33,991,448 bytes)
35240: Compilation Timestamp 2022-12-04 14:10:28 UTC Signing time 04 Dec 2022 15:12:37
35241: Entry Point 24893992 - Contained Sections 10
35242:
35243: ****************************************************************
35244: Release Notes maXbox 4.7.6.10 November 2021 mX476
35245: ****************************************************************
35246: Add 5 Units + 2 Tutorials
35247:
35248: 1441 unit uPSI_neuralgeneric.pas; CAI
35249: 1442 unit uPSI_neuralthread.pas; CAI
35250: 1443 unit uPSI_uSysTools; TuO
35251: 1444 unit upsi_neuralsets; mX4
35252: 1445 unit uPSI_uWinNT.pas mX4
35253: Total of Function Calls: 34880
35254: SHA1: of 4.7.6.10  CF939E3A8D4723DB1DEF383C5FC961E06728C58F
35255: CRC32: 38F88218 30.5 MB (32,022,344 bytes)
35256: V 4.7.5.90 VI
35257: Amount of Functions: 20549
35258: Amount of Procedures: 12307
35259: Amount of Constructors: 1937
35260: Amount of Destructors: 14
35261: Totals of Calls: 34807
35262: SHA1: of 4.7.5.90 96DCDE2028125E00B67E42A801721AC513A5EAFC
35263: CRC32: BBC3A7E5:  31974216 bytes
35264: ▯▯▯ mX4 executed: 03/11/2021 15:25:47  Runtime: 0:12:33.807  Memload: 39% use
35265: Amount of Functions: 20994
35266: Amount of Procedures: 12492
35267: Amount of Constructors: 1984
35268: Amount of Destructors: 14
35269: Totals of Calls: 35484
35270: SHA1: of 4.7.6.10 EC20D0A371A57AA078762D6D45F7229A370549A1
35271: CRC32: 82E481A4:  33273160 bytes
35272: ▯▯▯ mX4 executed: 14/03/2022 15:21:38  Runtime: 0:13:41.253  Memload: 42% use
35273:
```

```
35274: // Project:     State Machine
35275: // Module:      statmach.pas
35276: // Description: Visual Finite State Machine.
35277: // Version:     2.2a  - Release:  6
35278:
35279: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
35280: http://www.slideshare.net/maxkleiner1/codereview-topics
35281: ////////////////////////////////////////////////////////////////////////////
35282: //Form Template Library FTL
35283: ////////////////////////////////////////////////////////////////////////////
35284:
35285: FTL For Form Building Lib out of the Script, eg. 399_form_templates.txt
35286: 045 unit uPSI_VListView                        TFormListView;
35287: 263 unit uPSI_JvProfiler32;                   TProfReport
35288: 270 unit uPSI_ImgList;                        TCustomImageList
35289: 278 unit uPSI_JvImageWindow;                  TJvImageWindow
35290: 317 unit uPSI_JvParserForm;                   TJvHTMLParserForm
35291: 497 unit uPSI_DebugBox;                       TDebugBox
35292: 513 unit uPSI_ImageWin;                       TImageForm, TImageForm2
35293: 514 unit uPSI_CustomDrawTreeView;             TCustomDrawForm
35294: 515 unit uPSI_GraphWin;                       TGraphWinForm
35295: 516 unit uPSI_actionMain;                     TActionForm
35296: 518 unit uPSI_CtlPanel;                       TAppletApplication
35297: 529 unit uPSI_MDIEdit;                        TEditForm  //RichEditApp
35298: 535 unit uPSI_CoolMain; {browser}             TWebMainForm
35299: 538 unit uPSI_frmExportMain;                  TSynexportForm
35300: 585 unit uPSI_usniffer; {//PortScanForm}      TSniffForm
35301: 600 unit uPSI_ThreadForm;                     TThreadSortForm;
35302: 618 unit uPSI_delphi_arduino_Unit1;           TLEDForm
35303: 620 unit uPSI_fplotMain;                      TfplotForm1
35304: 660 unit uPSI_JvDBQueryParamsForm;            TJvQueryParamsDialog
35305: 677 unit uPSI_OptionsFrm;                     TfrmOptions;
35306: 718 unit uPSI_MonForm;                        TMonitorForm
35307: 742 unit uPSI_SimplePortMain;                 TPortForm1
35308: 770 unit uPSI_ovccalc;                        TOvcCalculator  //widget
35309: 810 unit uPSI_DbExcept;                       TDbEngineErrorDlg
35310: 812 unit uPSI_GL_actorUnit1;                  TglActorForm1  //OpenGL Robot
35311: 846 unit uPSI_synhttp_daemon;                 TTCPHttpDaemon, TTCPHttpThrd, TPingThread
35312: 867 unit uPSI_Debug;                          TfrmDebug
35313: 901 unit uPSI_JvAddPrinter                    TJvAddPrinter     //JCL
35314: 904 unit uPSI_U_HexView;                      THexForm2
35315: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)   TOscfrmMain
35316: 959 unit uPSI_AdSelCom;                       TComSelectForm
35317: 1029 unit uPSI_GPSUDemo;                      TFDemo
35318: 1031 unit uPSI_ScreenThreeDLab;               TFormLab3D
35319: 1051 unit uPSI_panUnit1;                      TPanForm1 //GLScene
35320: 1052 unit uPSI_DD83u1;  {Arduino Tester Frm}  TDD83f1
35321: 1059 unit uPSI_uconvMain;                     TfconvMain //PS
35322: 1076 unit uPSI_CustomizeDlg;                  TCustomizeDlg / TJvAddPrinterDialog;
35323: 1111 unit uPSI_JvExceptionForm;               TJvErrorDialog  //ShowException
35324: 1113 unit uPSI_MTMainForm;                    TvtMainForm
35325: 1119 unit uPSI_CPortTrmSet;                   TComTrmSetForm
35326: 1146 unit uPSI_ChessPrg;                      TChessForm1
35327: 1216 uPSI_KDialogs,                           TKBrowseFolderDialog
35328: 1217 eg.execStr('from delphivcl import *');   LoadPyClass
35329:
35330: FormTemplates with <Ctrl> J
35331:
35332: [myformtemplate | formtemplate statement | Borland.EditOptions.Pascal] //with
35333: [myFastForm | class declaration  | Borland.EditOptions.Pascal]  //Dialog Form
35334: [myForm | class declaration  | Borland.EditOptions.Pascal]   //with Events
35335: [aForm | class declaration  | Borland.EditOptions.Pascal]   //single Form
35336: [getForm(x,y)] of object TForm [loadForm(x,y)]
35337: [ticker | ticker statement | Borland.EditOptions.Pascal]
35338: [except | exception statement | Borland.EditOptions.Pascal]
35339:
35340: Dark Color Style Darkcolor:
35341: memo1.color:= RGB2TColor(32,32,32)  //20,18,18
35342: memo2.color:= RGB2TColor(20,18,18)
35343:
35344: Proc SIRegister_JvDesignUtils(CL: TPSPascalCompiler);
35345: begin
35346:  Func DesignClientToParent(const APt: TPoint; AControl,AParent: TControl) : TPoint;
35347:  Func DesignMin(AA, AB : Int):Int);
35348:  Func DesignMax(AA, AB: Int): Int;
35349:  Func DesignRectWidth( const ARect : TRect) : Int;
35350:  Func DesignRectHeight( const ARect : TRect) : Int;
35351:  Func DesignValidateRect( const ARect : TRect) : TRect;
35352:  Func DesignNameIsUnique( AOwner : TComponent; const AName :Str) :Bool;
35353:  Func DesignUniqueName( AOwner : TComponent; const AClassName :Str) :Str;
35354:  Proc DesignPaintRubberbandRect(AContainer:TWinControl;ARect:TRect;APenStyle:TPenStyle);
35355:  Proc DesignPaintGrid(ACanvs:TCanvas;const ARect:TRect;ABackClr:TColr;AGridClr:TColor;ADivPixels:Int)
35356:  Proc DesignPaintRules(ACanvas:TCanvas;const ARect:TRect;ADivPixels:Int;ASubDivs:Bool;
35357:  Proc DesignSaveComponentToStream( AComp : TComponent; AStream : TStream);
35358:  Func DesignLoadComponentFromStream(AComp:TComponent;AStram:TStream;AOnError:TReaderError):TComponent;
35359:  Proc DesignSaveComponentToFile( AComp : TComponent; const AFileName :Str);
35360:  Proc DesignLoadComponentFromFile(AComp:TComp;const AFileName:str;AOnError:TReaderError);
35361: end;
35362:
```

```
35363: ex.:with TEditForm.create(self) do begin
35364:       caption:= 'Template Form Tester';
35365:       FormStyle:= fsStayOnTop;
35366:       with editor do begin
35367:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
35368:         SelStart:= 0; Modified:= False;
35369:       end;
35370:     end;
35371:   with TWebMainForm.create(self) do begin
35372:     URLs.Text:= 'http://www.kleiner.ch';
35373:     URLsClick(self); Show;
35374:   end;
35375:   with TSynexportForm.create(self) do begin
35376:     Caption:= 'Synexport HTML RTF tester';
35377:     Show;
35378:   end;
35379:   with TThreadSortForm.create(self) do begin
35380:      showmodal; free;
35381:   end;
35382:   with TCustomDrawForm.create(self) do begin
35383:        width:=820; height:=820;
35384:        image1.height:= 600; //add properties
35385:        image1.picture.bitmap:= image2.picture.bitmap;
35386:        //SelectionBackgroundlClick(self) CustomDrawlClick(self);
35387:        Background1.click;
35388:        bitmap1.click; Tile1.click;
35389:        Showmodal;
35390:        Free;
35391:      end;
35392:    with TfplotForm1.Create(self) do begin
35393:      BtnPlotClick(self);
35394:      Showmodal; Free;
35395:    end;
35396:   with TOvcCalculator.create(self) do begin
35397:     parent:= aForm;  //free;
35398:     setbounds(550,510,200,150);
35399:     displaystr:= 'maXcalc';
35400:   end;
35401:   with THexForm2.Create(self) do begin
35402:      ShowModal;
35403:      Free;
35404:   end;
35405:
35406: Func CheckBox:Str;
35407: var idHTTP: TIDHTTP;
35408:  begin
35409:    result:= 'version not found';
35410:    if IsInternet then begin
35411:      idHTTP:= TIdHTTP.Create(NIL);
35412:      try
35413:        result:= idHTTP.Get(MXVERSIONFILE2);
35414:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
35415:        if result = MBVER2 then begin
35416:          //Speak(' A new Version '+vstr+' of max box is available! ;
35417:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
35418:        end;  //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
35419:      finally
35420:        idHTTP.Free
35421:    end;
35422:  end;
35423: end;
35424:
35425: charinset make your own charinset to prevent typematch:
35426:
35427: Func CharInSet4(C: AnsiChar; const CharSet: TSysCharSet):Bool;
35428: begin
35429:   Result:= C in CharSet;
35430: end;
35431:
35432: Func CharInSet5(C: AnsiChar; const CharSet: CharSet):Bool;
35433: begin
35434:   Result:= C in CharSet;
35435: end;
35436:
35437: //Runtimer Spec Functions Edition 190 and more code blocks
35438: Func ApWinExecAndWait32(FileName:PChar;CommandLine:PChar; Visibility:Int):Int;
35439: Func KillTask(ExeFileName:Str): Int;
35440: Proc KillProcess(hWindowHandle: HWND);
35441: Func FindWindowByTitle(WindowTitle:Str): Hwnd;
35442: Func OpenIE(aURL:Str):Bool;
35443: Func XRTLIsInMainThread:Bool;
35444: Func IsInMainThread:Bool;
35445: Func IntToFloat(i: Int): double;
35446: Func AddThousandSeparator(S:Str; myChr: Char):Str;
35447: Func mciSendString(cmd: PChar; ret: PChar; len: Int; callback: Int):Card;
35448: Proc FormAnimation(Sender: TObject; adelay: Int);
35449: Proc LoadResourceFile2(aFile:str; ms:TMemoryStream);
35450: Func putBinResTo(binresname: pchar; newpath:Str):Bool;
35451: Proc ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:str);
```

```
35452: Func IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:str;MouseX,MouseY:Int;var HyperLink:str):Bool
35453: Func GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD) : DWORD;;
35454: Func GetWindowTask( hWnd : HWND) : THandle;
35455: Func LoadBitmap( hInstance : HINST; lpBitmapName : PChar) : HBITMAP;
35456: Func GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL;
35457: Func WinExecAndWait32(FileName:Str; Visibility: Int): Longword;
35458: Func MakeHash( const s : TbtString) : Longint;
35459: Func GetUsedUnitList( list : Tstringlist) :Str;
35460: Func ConsoleCapture(const _dirName, _exeName, _cmdLine:Str; amemo: TStringlist): Bool;
35461: Func ConsoleCaptureDOS(const _dirName, _exeName, _cmdLine:Str; amemo: TStrings): Bool;
35462:          srlist:= TStringlist.create;
35463:          ConsoleCapture('C:\', 'cmd.exe', '/c dir *.*',srlist);
35464:          writeln(srlist.text); srlist.Free;
35465: Func RunCaptured(const _dirName, _exeName,_cmdLine:str; amemo:TStringlist):Bool;;
35466: Func SamePropTypeName( const Name1, Name2 : ShortString) :Bool;
35467: Func FloatToStrEx( Value : Extended) :Str;
35468: Func StrToFloatEx( const S :Str) : Extended;
35469: Proc PerformanceDelayMS(ams: Int);  //microsecond resolution delay!
35470: //http://www.swissdelphicenter.ch/en/showcode.php?id=2179
35471: Func ExecuteProcess(FileName:Str;Visibility:Int; BitMask:Int; Synch:Boolean):Longword;
35472: Func ExecuteMultiProcessor(FileName:str;Visibility:Int;BitMask:Int;Synch:Bool):Longword;
35473:       if ExecuteMultiProcessor('notepad.exe', SW_SHOW, 2, true) = 0 then
35474:                                writeln('Multiprocessing Runs on CPU 2;
35475: Proc StartServiceAfterInstall(aserv: TService);
35476: Func GetDllVersion2(DllName:Str; var DLLVersionInfo: TDLLVersionInfo):Bool;
35477: Proc SendCopyMessage(amess,astation:str); //comunicate process-spanned with WM_COPYDATA
35478: Func BrowseComputer2(DialogTitle:Str; var CompName:Str; bNewStyle: Bool): Bool;
35479: Func ChangeAlphaTo(input:Str; aoffset: byte):Str;;
35480: Func CheckIBAN(iban:Str):Bool;;
35481: Func CreateIDStack; //instance to IdStack of CreateStack
35482: Func GetRecordCount(DataSet: TBDEDataSet): Longint;;
35483: Func CountPos(const subtxt:Str; Text:Str): Int;
35484: Proc HTMLtoRTF(html:Str; var rtf: TRichedit);
35485: Proc ReversePlay(const szFileName:Str);
35486: Func ADOConnectionString(ParentHandle:THandle;InitialStr:WideString;out NewString:str):Bool;
35487: Proc ShowEOleException(AExc: EOleException; Query:str);
35488: Func UpdateBlob(Connection:TADOConnection;Spalte:str;Tabelle:str;Where:str;var
       ms:TMemoryStream):Bool;..save HTML pages as MHTML(HTML Archiv Format)
35489:    http://www.swissdelphicenter.ch/en/showcode.php?id=2300
35490: Func SaveToMHT(const AUrl,AFileName:Str; AShowErrorMessage:Boolean = False):Bool;
35491: Func FileType2MimeType(const AFileName:Str):Str;
35492: Func DownloadURL_NOCache(const aUrl:Str; var s:Str):Bool;
35493: Func IsCOMObjectActive(ClassName:Str):Bool;
35494: Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str = ';
35495: Func CheckCreditCard(c:Str): Int;
35496:    0: Card is invalid or unknown 1: is a AmEx 2: is a Visa 3: is a valid MasterCard
35497: Func GetNewGUID:Str;
35498: Func FormatGUID(const GUID:Str):Str;
35499: Func GetNewFormatedGUID:Str;
35500: Func getFormRes(classname:Str):Str;  //shows DFM Res of Exe!
35501: Proc OutputDebugString(PChar(Format('[%s][%s]%s',[aCaption,GetFormatDT(StartDT),aText])));
35502: Proc ScanNetworkResources(ResourceType, DisplayType: DWord; List: TStrings);
35503: Func PrepareConstraint(Src:Tstrings):str;
35504: Proc DeleteEmptyStr(Src:Tstrings);
35505: Func NormalizeSQLText(const SQL:Str;MacroChar:Char):Str;
35506: Func CountSelect(const SrcSQL:str):str;
35507: Func GetModifyTable(const SQLText:str;AlreadyNormal:boolean):str;
35508: Func GetCharFromVKey(vkey: Word):Str;
35509: Func Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile:Str):Bool;
35510: Func IsObjectActive(ClassName:Str):Bool;
35511:  Func GetActiveObject(ClassID:TGUID; anil:TObject; aUnknown:IUnknown):HRESULT;;
35512: Func RegisterOCX(FileName:Str):Bool;
35513: Func UnRegisterOCX(FileName:Str):Bool;
35514: Func RegisterServer2(const aDllFileName:Str; aRegister:Bool):Bool;
35515: Proc mIRCDDE(Service, Topic, Cmd:Str); //mIRCDDE('mIRC', 'COMMAND', '/say Hallo von SwissDelphiCenter.ch;
35516: Func OpenIE(aURL:Str):Bool;
35517: Func XRTLIsInMainThread:Bool;
35518: Func IsInMainThread:Bool;
35519: TryConvertStrToDateTime(const s, format:str; out value: TDateTime):Boolean;;
35520:   ConvertStrToDateTime(const s, format:Str): TDateTime;;
35521:  Func CreateNotifyThread2(const FolderName:Str; WatchSubtree:Bool; Filter : TFileChangeFilters2):
       TNotifyThread;
35522:  Proc DetectImage(const InputFileName:Str; BM: TBitmap);
35523:  Func BitmapToString(Bitmap: TBitmap):Str;
35524:  Func StringToBitmap(S:Str): TBitmap;
35525:  Func RemoveChar(CONST s:Str; CONST c: CHAR):Str;
35526:  Proc SecureClearStr(var S: Ansistr);;
35527:  Proc movestring(const Source:str; var Destination:str; CopyCount : Int );
35528:  Proc movestringJV(const Source:str; var Destination:str; CopyCount : Int );
35529:  Proc moveint(const Source:Int; var destination: Int; CopyCount : Int );
35530:  Proc movefloat(const Source:double; var destination: double; CopyCount : Int );
35531:  Proc moveextended(const Source:extended;var Destination:extended;CopyCount:Int);
35532:  Proc ShowFilePropertiesSH(Files: TStrings; aWnd: HWND);
35533: //ScanNetworkResources(RESOURCETYPE_DISK, RESOURCEDISPLAYTYPE_SERVER, ListBox1.Items);
35534: Func GrabLine(const s:Str; ALine: Int):Str;
35535: Func GrabLineFast(const s:Str; ALine: Int):Str;
35536: Func IsTextFile(const sFile: TFileName):Bool;
35537: Func getODBC: Tstringlist;
35538:  Func getODBCString:Str;
```

```
35539:  Proc GetJPGSize(const sFile:Str; var wWidth, wHeight: Word);
35540:  Proc GetPNGSize(const sFile:Str; var wWidth, wHeight: Word);
35541:  function GetPNGSize2(const FileName: string): TSize;
35542:  Proc GetGIFSize(const sGIFFile:Str; var wWidth, wHeight: Word);
35543: // note: using TForm's BorderIcons, etc. is slow (form blinks) and not reliable (for some
35544: //   reson it causes TListView.Items to lose all associated objects and other things happen).
35545: Proc ChangeWindowStyle(const Form: HWND; Style: DWord; AddIt:Bool);
35546: Func VarByteArrayOf(const s:Str): OleVariant;
35547: Func bigdiv2(aval1:Str; aval2: integer):Str;
35548: Func modbig(aval:Str; amod: integer): integer;
35549: Func bigmod(aval:Str; amod: integer): integer;
35550: Func modPowBig3(aval, apow, amod:Str):Str;;
35551: Func BigPowMod(aval, apow, amod:Str):Str;;
35552: Func RSAEncrypt(aval, apow, amod:Str):Str;;
35553: Func RSADecrypt(aval, apow, amod:Str):Str;;
35554: Func SwitchToThread : BOOL;
35555: Func SetThreadDesktop( hDesktop : HDESK) : BOOL;
35556: Func CloseDesktop( hDesktop : HDESK) : BOOL;
35557: Func GetThreadDesktop( dwThreadId : DWORD) : HDESK;
35558: Func SetSyscallHook():Bool;;
35559: Func SetSwapcontextHook():Bool;;
35560: Func UnhookAll():Bool;;
35561: Func getWorld:Str;;
35562: Func getIPConfigAll:Str;;
35563: Func getIPConfig:Str;;
35564: Func WinsockEnabled:Bool;
35565: Func HTTPEncode2(const AStr:Str):Str;
35566: Func ComputePEChecksum(FileName:Str): DWORD;
35567:
35568: if not DynamicDllCallName(user32, 'LockWorkStation',true,returned,parameters) then begin
35569: Func DynamicDllCallNames(Dll:Str; const Name:Str; HasResult:Bool; var Returned:Card; const Parameters:
       array of string):Bool;;
35570: Proc GetMIDASAppServerList(List: TStringList; const RegCheck :Str);
35571: Proc SQLDropField(dbName, tblName,  fldName:Str);  {Field Name to Drop}
35572: type TCastType = (ctSmallInt, ctInt, ctDecimal, ctNumeric, ctFloat,
35573:                ctChar, ctVarChar, ctDate, ctBoolean, ctBLOB, ctTime,
35574:                ctTimeStamp, ctMoney, ctAutoInc, ctBytes);
35575: {Blob definition type 1 = Memo, 2 = Binary, 3 = Formatted Memo, 4 = OLE Object, 5 = Graphic}
35576:
35577: Proc SQLAddField(dbName,tblName,fldName:str;fldType:TCastType;fldLength,precisOrBlobLen,
       scaleOrBlobType:Int);
35578: const
35579:  UrlGeoLookupInfo  ='http://ipinfodb.com/ip_query.php?timezone=true&ip=%s';
35580:  UrlGeoLookupInfo2 ='http://backup.ipinfodb.com/ip_query.php?timezone=true&ip=%s'; //backup
35581:  http://ip-api.com/csv/
35582:
35583: Proc GetGeoInfo(const IpAddress:str;var GeoInfo:TGeoInfo;const UrlGeoLookupInfo:str);
35584:  TGeoInfo','record status:Str; countrycode : '
35585:    +string; countryname:str; regioncode:Str; city:Str; zippostalcode:Str;
       latitude:Str;longitude:str;timezonename:str;gmtoffset:str;isdst:str; end;
35586:
35587: Proc SIRegister_ubigFloatV3(CL: TPSPascalCompiler);
35588: begin
35589:   TMaxSig', 'Int;
35590:   TView', '( normal, Scientific );
35591:   SIRegister_TFloatInt(CL); SIRegister_TBigFloat(CL);
35592: end;
35593:
35594: Proc SIRegister_UBigIntsV4(CL: TPSPascalCompiler);
35595: begin TDigits', 'array of int64; SIRegister_TInt(CL);
35596:  Proc SetBaseVal( const newbase : Int);
35597:  Func GetBasePower : Int;
35598:  Func GetBase : Int;
35599:  Proc SetThreadSafe( newval :Bool);
35600:  Func bigdiv2(aval1:Str; aval2: integer):Str;
35601:  Func modbig(aval:Str; amod: integer): integer; ;
35602:  Func bigmod(aval:Str; amod: integer): integer; ;
35603:  Func modPowBig3(aval, apow, amod:Str):Str;;
35604:  Func BigPowMod(aval, apow, amod:Str):Str;;
35605:  Func RSAEncrypt(aval, apow, amod:Str):Str;;
35606:  Func RSADecrypt(aval, apow, amod:Str):Str;;
35607:  Func SwitchToThread : BOOL;
35608: end;
35609:
35610: Func BigDiv(aone, atwo:Str):Str;
35611: var tbig1, tbig2: TInt;
35612: begin
35613:   tbig1:= TInt.create(10);
35614:   tbig2:= TInt.create(10);
35615:   try
35616:     tbig1.assign2(atwo)
35617:     tbig2.assign2(aone)
35618:     tbig2.Divide(tbig1)
35619:   finally
35620:     result:= tbig2.ConvertToDecimalString(false)
35621:     tbig1.Free;
35622:     tbig2.free;
35623:   end;
35624: end;
```

```
35625:
35626: Proc SIRegister_UDict(CL: TPSPascalCompiler);
35627: begin 'dichighletter','String').SetString( 'z;
35628:   SIRegister_TDicForm(CL);
35629:   SIRegister_TDic(CL);
35630: end;
35631:
35632: Proc SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: Int);
35633: var i: Int;
35634: begin setlength(arr, asize1);
35635:    for i:= 0 to asize1-1 do SetLength(arr[i], asize2);
35636: end;
35637:
35638: Example of dynamic array of panels:
35639: Proc SetArrayLength2Panels(var arr: array of array of TPanel;
35640:                                          asize1, asize2: Integer);
35641: var i: Integer;
35642: begin setlength(arr, asize1);
35643:    for i:= 0 to asize1-1 do SetLength(arr[i], asize2);
35644: end;
35645:
35646: Proc TMyFormInitialisePanels(aform: Tform; RowCount,ColCount: Integer);
35647: var aLeft,aTop,aWidth,aHeight, row,col: Integer;
35648:   Panel: TPanel;
35649:   FPanels: array of array of TPanel;
35650: begin
35651:   //SetLength(FPanels, RowCount, ColCount);
35652:   SetArrayLength2Panels(Fpanels, RowCount, ColCount)
35653:   aTop:= 0;
35654:   for Row:= 0 to RowCount-1 do begin
35655:     aLeft:= 0;
35656:     aHeight:= (aform.ClientHeight-aTop) div (RowCount-Row);
35657:     for Col:= 0 to ColCount-1 do begin
35658:       Panel:= TPanel.Create(Self);
35659:       FPanels[Row][Col]:= Panel;
35660:       Panel.Parent:= aform; //Self;
35661:       //panel.parentcolor:= false;
35662:       panel.ParentBackground:= false;
35663:       panel.color:= random(clred)
35664:       aWidth:= (aform.ClientWidth-aLeft) div (ColCount-Col);
35665:       Panel.SetBounds(aLeft, aTop, aWidth, aHeight);
35666:       inc2(aLeft, aWidth);
35667:     end;
35668:     inc2(aTop, aHeight);
35669:   end;
35670: end;
35671:   aform:= TForm.create(self)
35672:   aform.setbounds(10,10,600,500);
35673:   TMyFormInitialisePanels(aform, 26, 19);
35674:   aform.show;
35675:
35676: Proc SIRegister_UP10Build(CL: TPSPascalCompiler);
35677: begin Proc ParseFunction(FunctionString:str; Variables:TStringlist; FunctionOne,
       FunctionTwo:TStringList;UsePascalNumbers:bool;var FirstOP:{TObject}POperation;var Error:bool);
35678: end;
35679:
35680: Proc SIRegister_ComObj2(CL: TPSPascalCompiler);
35681: begin
35682:  ClassN(CL.FindClass('TOBJECT'),'TComObjectFactory;
35683:  SIRegister_TComServerObject(CL);
35684:  SIRegister_ADOConst(CL);
35685:   FieldTypeNames: array[0..41] of string = (
35686:     Unknown', 'String', 'SmallInt', 'Int', 'Word', 'Boolean', 'Float',
35687:     'Currency', 'BCD', 'Date', 'Time', 'DateTime', 'Bytes', 'VarBytes',
35688:     'AutoInc', 'Blob', 'Memo', 'Graphic', 'FmtMemo', 'ParadoxOle',
35689:     'dBaseOle', 'TypedBinary', 'Cursor', 'FixedChar', 'WideString',
35690:     'LargeInt', 'ADT', 'Array', 'Reference', 'DataSet', 'HugeBlob', 'HugeClob',
35691:     'Variant', 'Interface', 'Dispatch', 'Guid', 'SQLTimeStamp', 'FMTBcdField',
35692:   FixedWideChar', 'WideMemo', 'SQLTimeStamp', 'String;
35693:   TFactoryProc', 'Proc (Factory : TComObjectFactory);
35694:   TCallingConvention','(ccRegister, ccCdecl, ccPascal, ccStdCall, ccSafeCall);
35695:    SIRegister_TComClassManager(CL);
35696:    SIRegister_IServerExceptionHandler(CL); SIRegister_TComObject(CL);
35697:   //TComClass', 'class of TComObject;
35698:   TClassInstancing', '(ciInternal,ciSingleInstance,ciMultiInstance );
35699:   TThreadingModel', '(tmSingle, tmApartment, tmFree, tmBoth, tmNeutral );
35700:   SIRegister_TComObjectFactory(CL); SIRegister_TTypedComObject(CL);
35701:   //TTypedComClass', 'class of TTypedComObject;
35702:   SIRegister_TTypedComObjectFactory(CL);
35703:   TConnectEvent2','Procedure(const Sink : IUnknown; Connecting :Bool);
35704:   ClassN(CL.FindClass('TOBJECT'),'TAutoObjectFactory;
35705:   SIRegister_TAutoObject(CL); //TAutoObject2  ?? in OleAuto and ComObj
35706:   //TAutoClass', 'class of TAutoObject;
35707:   SIRegister_TAutoObjectFactory(CL); SIRegister_TAutoIntfObject(CL);
35708:   //ClassN(CL.FindClass('TOBJECT'),'EOleError;
35709:   ClassN(CL.FindClass('Exception'),'EOleError;
35710:   SIRegister_EOleSysError(CL); SIRegister_EOleException(CL);
35711:   ClassN(CL.FindClass('TOBJECT'),'EOleRegistrationError;
35712:  //Proc DispatchInvoke(const
       Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Result: PVariant);
```

```
35713:  //Proc DispatchInvokeError( Status : Int; const ExcepInfo : TExcepInfo);
35714:  //Func HandleSafeCallException(ExceptObject:TObject;ExceptAddr:Pointer;const ErrorIID  TGUID;const ProgID,
      HelpFileName:WideString):HResult;
35715:  Func CreateComObject( const ClassID : TGUID) : IUnknown;
35716:  Func CreateRemoteComObject(const MachineName: WideString;const ClassID:TGUID):IUnknown;
35717:  //Func CreateOleObject(const ClassName :Str): IDispatch;
35718:  //Func GetActiveOleObject(const ClassName:str): IDispatch;
35719:  Proc OleError2( ErrorCode : HResult);
35720:  Proc OleCheck2( Result : HResult);
35721:  Func StringToGUID2( const S :Str) : TGUID;
35722:  Func GUIDToString2( const ClassID : TGUID) :Str;
35723:  Func ProgIDToClassID2( const ProgID :Str) : TGUID;
35724:  Func ClassIDToProgID2( const ClassID : TGUID) :Str;
35725:  Proc CreateRegKey(const Key,ValueName,Value:str;RootKey:DWord);
35726:  Proc DeleteRegKey(const Key :Str; RootKey : DWord);
35727:  Func GetRegStringValue( const Key, ValueName :Str; RootKey : DWord) :Str;
35728:  //Func StringToLPOLESTR( const Source :Str) : POleStr;
35729:  Proc RegisterComServer( const DLLName :Str);
35730:  Proc RegisterAsService( const ClassID, ServiceName :Str);
35731:  Func CreateClassID2:Str;
35732: Proc InterfaceConnect(const Srce:IUnknown;const IID:TIID;const Sink:IUnknown;var Connectin:Longint);
35733:  Proc InterfaceDisconnect(const Source:IUnknown;const IID:TIID;var Connection:Longint);
35734:  Func GetDispatchPropValue(Disp: IDispatch; DispID : Int) : OleVariant;;
35735:  Func GetDispatchPropValue1(Disp: IDispatch; Name : WideString) : OleVariant;;
35736:  Proc SetDispatchPropValue2(Disp:IDispatch;DispID:Int;const Value:OleVariant);
35737:  Proc SetDispatchPropValue3(Disp: IDispatch; Name:WideString; const Value:OleVariant);;
35738:  Func ComClassManager : TComClassManager;
35739: //  from ADODB OLE Utils
35740:   TOleEnum', 'LongWord; //DataTypeEnum = TOleEnum;
35741:   DataTypeEnum', 'TOleEnum;
35742:  Func CreateADOObject( const ClassID : TGUID) : IUnknown;
35743:  Func ADOTypeToFieldType(const ADOType:DataTypeEnum; EnableBCD:Boolean):TFieldType;
35744:  Func FieldTypeToADOType(const FieldType: TFieldType):DataTypeEnum;
35745:  Func StringToVarArray( const Value :Str) : OleVariant;
35746:  Func VarDataSize( const Value : OleVariant) : Int;
35747:  Func OleEnumToOrd( OleEnumArray : array of TOleEnum; Value : TOleEnum) : Int;
35748:  Func GetStates( State : Int) : TObjectStates;
35749:  Func ExecuteOptionsToOrd(ExecuteOptions:TExecuteOptions): Int;
35750:  Func OrdToExecuteOptions(Options: Int) : TExecuteOptions;
35751:  Func ExtractFieldName( const Fields : WideString; var Pos : Int) : WideString;
35752:  Func GetFilterStr( Field : TField; Value : Variant; Partial :Bool) : WideString;
35753:  Func FieldListCheckSum( DataSet : TDataset) : Int;
35754:  Func GlobalAllocString(s: Ansistr): HGlobal;
35755:  Func ScanTime(const S:Str; var Pos: Int; var Time: TDateTime):Bool;
35756:  Func ScanChar(const S:Str; var Pos: Int; Ch: Char):Bool;
35757:  Func ScanNumber(const S:Str; var Pos: Int; var Number: Word):Bool;
35758:  Func ScanString(const S:Str; var Pos: Int; const Symbol:Str):Bool;
35759:  Proc LV_InsertFiles(strPath:Str; ListView: TListView; ImageList: TImageList);
35760:  Func GetPasteLinkInfo(var Service:Str; var Topic:str; var Item:Str):Boolean;
35761:  Func IPToHostName(const IP:Str):Str;
35762:  Proc GetZoneIcon(IconPath:Str; var Icon: TIcon);
35763:  Func GetZoneAttributes(const URL:Str): TZoneAttributes;
35764:  //unit uPSI_PsAPI;
35765:  Proc CGITester;
35766:  //CGI will take name and email address from command line and place it into HTML
35767:  Proc CreateBrowserOnForm(aform: TForm; aurl:Str);
35768:  Proc WebOnForm(aform: TForm; aurl:Str);;
35769:  Proc WebToForm(aform: TForm; aurl:Str);;
35770:  Proc SearchAndHighlightWebText(aform: TForm; aurl:Str; aText:Str);
35771:  Proc SaveImagesOnWeb(aurl, apath:Str);
35772:  Func GetProcessNameFromWnd(Wnd: HWND):Str; //get EXE path from window handle
35773:  Func getallEvents(aform: TForm): TStringlist;
35774:  Proc GetKLList(List: TStrings);;
35775:  Proc GetKeyboardList(List: TStrings);;
35776:  Func SetSuspendState(Hibernate:Boolean;ForceCritical:Boolean;DisableWakeEvent:Bool):bool;
35777:          call: SetSuspendState(True, False, False);
35778:  Func ServiceRunning(sMachine, sService: PChar):Bool;
35779:  Func isServiceRunning(sMachine, sService: PChar):Bool;
35780:  Proc CloseOpenSockets( Sockets : array of TIdStackSocketHandle);
35781:  //SIRegister_SvrHTTPIndy - Linux
35782:  Proc TransformError( const Msg :Str);
35783:  Proc StringToFile2( const S, FileName :Str);
35784:  Func GetXMLData( DataSet : TClientDataSet) :Str;
35785:  Proc EditComTerminal( ComTerminal : TCustomComTerminal);  //TComTrmSetForm
35786:  Func search_adapter_key_networkcard:Str;  //at registry
35787:  Func getNetworkCard:Str;;
35788:  Func GetMacAddresses2(const Machine:Str; const Addresses: TStrings): Int;
35789: Func ConnectDrive(_drvLetter:str;_netPath:str;_showError:Bool;_reconnect:Bool):DWORD;
35790: Func ConnectPrinterDevice(_lptPort:str;_netPath:str;_showError:Bool;_reconnect:Bool):DWORD;
35791: Func DisconnectNetDrive(_locDrive:str;_showError:Bool;_force:Bool; _save:Bool): DWORD;
35792:     //ConnectDrive('k:', '\\Servername\C', True, True);
35793:     //DisconnectNetDrive('k:', True, True, True);
35794:  Func GetConnectionKind(var strKind:Str):Bool; MODEM =1; LAN=2; PROXY=4; BUSY= 8;
35795:  Func DownloadJPGToBitmap(const URL :Str; ABitmap: TBitmap):Bool;
35796:  Proc GetImageLinks(AURL:Str; AList: TStrings);
35797:  Func GetCharEncoding( alias :Str; var _name :Str) : Int;
35798:  Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
35799:  Func MicrosoftLangageCodeToISOCode( langcode : Int) :Str;
35800:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str = ';
```

```
35801:  Func RFC1123ToDateTime(Date:Str): TDateTime;
35802:  Func DateTimeToRFC1123(aDate: TDateTime):Str;
35803:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str);;
35804:  Proc DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);;
35805:  Proc DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);;
35806:  Proc DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);;
35807:
35808:  IPToHostName
35809:   //from ADOInt.pas //  TOleEnum = type LongWord;
35810:   TypeS('CursorOptionEnum', 'TOleEnum;
35811:    Interface(CL.FindInterface('IUNKNOWN'),_Recordset, '_Recordset;
35812:   //TypeS('_RecordsetDisp', 'dispinterface;
35813:   Interface(CL.FindInterface('IUNKNOWN'),_Command, '_Command;
35814:   Interface(CL.FindInterface('IUNKNOWN'),_Connection, '_Connection;
35815:   // SIRegister__Recordset(CL); // SIRegister__Command(CL);
35816:   Const IID_Recordset20','TGUID').SetString('{0000054F-0000-0010-8000-00AA006D2EA4}''];
35817:   'IID__Recordset','string').SetString( '00000555-0000-0010-8000-00AA006D2EA4;
35818:       //test with stringToGUID
35819:  //'CLASS_Command','TGUID').SetString( '{00000507-0000-0010-8000-00AA006D2EA4};
35820: // 'CLASS_Recordset','TGUID').SetString( '{00000535-0000-0010-8000-00AA006D2EA4};
35821:  { TypeS('Connection', '_Connection;
35822:   TypeS('Command', '_Command;  TypeS('Recordset', '_Recordset;
35823:   TypeS('Parameter', '_Parameter; TypeS('DataSpace', 'IDataspace;
35824:   TypeS('SearchDirection', 'SearchDirectionEnum; }
35825:   //TypeS('Command', '_Command; //TypeS('Recordset', '_Recordset;
35826: end;
35827:
35828: Proc SIRegister_DIUtils(CL: TPSPascalCompiler);
35829: begin
35830:  //'CRLF','String').SetString( #$0D#$0A);
35831:  'REPLACEMENT_CHARACTER','LongWord').SetUInt( $FFFD);
35832:  'HANGUL_SBase','LongWord').SetUInt( $AC00);
35833:  'HANGUL_LBase','LongWord').SetUInt( $1100);
35834:  'HANGUL_VBase','LongWord').SetUInt( $1161);
35835:  'HANGUL_TBase','LongWord').SetUInt( $11A7);
35836:  'HANGUL_LCount','LongInt').SetInt( 19);
35837:  'HANGUL_VCount','LongInt').SetInt( 21);
35838:  'HANGUL_TCount','LongInt').SetInt( 28);
35839:  'KEY_WOW64_32KEY','LongWord').SetUInt( $0200);
35840:  'KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
35841:  'KEY_WOW64_RES','LongWord').SetUInt( $0300);
35842:   TypeS('TAnsiCharSet', 'set of AnsiChar;
35843:   TypeS('TIsoDate', 'Cardinal;
35844:   TypeS('TJulianDate', 'Int;
35845:   //TypeS('PJulianDate', '^TJulianDate // will not work;
35846:   TypeS('TValidateCharFunc', 'function(const c: Char):Bool;;
35847:   // TValidateCharFunc = function(  const c: Char):Bool;
35848:  TypeS('TProcedureEvent', 'Procedure;
35849:  'MT19937_N','LongInt').SetInt( 624);
35850:  'MT19937_M','LongInt').SetInt( 397);
35851:   SIRegister_TMT19937(CL);
35852:   //SIRegister_TWideStrBuf(CL);
35853:  TypeS('TDITextLineBreakStyle', '( tlbsLF, tlbsCRLF, tlbsCR );
35854:  Func AdjustLineBreaksW(const s:UnicodeStr;const Style:TDITextLineBreakStyle):UnicodeStr;
35855:  Func BrightenColor( const Color : Int; const amount : Byte) : Int;
35856:  Func BSwap4( const Value :Card) :Card;;
35857:  Func BSwap5( const Value : Int) : Int;;
35858:  //Func BufCompNumIW(p1:PWideChar; l1:Int; p2 : PWideChar; l2 : Int) : Int;
35859:  Func BufSameA( p1, p2 : PChar; l :Card) :Bool;
35860: // Func BufSameW( p1, p2 : PWideChar; l :Card) :Bool;
35861:  Func BufSameIA( p1, p2 : PChar; l :Card) :Bool;
35862:  //Func BufSameIW( p1, p2 : PWideChar; l :Card) :Bool;
35863:  Func BufPosCharA(const Buf:PChar;l:Card;const c:AnsiChar;const Start:Card):Int;
35864:  Func BufPosCharsA(const Buf:PChar;l:Card;const Search:TAnsiCharSet;const Start:Card):Int;
35865:  Func BufStrSame(const Buf:PChar;const BufCharCount:Card;const s:Str):Bool;
35866:  Func BufStrSameA(const Buf:PChar; const BufCharCount:Card;const s:RawByteString):Bool;
35867:  Func BufStrSameI(const Buf:PChar; const BufCharCount:Card; const s :Str):Bool;
35868:  Func BufStrSameIA(const Buf:PChar;const BufCharCount:Card;const s:RawByteString): Bool;
35869:  Func diChangeFileExt(const FileName,Extension:Str):Str;
35870:  Func ChangeFileExtA( const FileName, Extension : Ansistr) : Ansistr;
35871:  Func ChangeFileExtW( const FileName, Extension : UnicodeString) : UnicodeString;
35872:  //Func CharDecomposeCanonicalW( const c : WideChar) : PCharDecompositionW;
35873:  Func CharDecomposeCanonicalStrW( const c : WideChar) : UnicodeString;
35874:  //Func CharDecomposeCompatibleW( const c : WideChar) : PCharDecompositionW;
35875:  Func CharDecomposeCompatibleStrW( const c : WideChar) : UnicodeString;
35876:  Func CharIn8( const c, t1, t2 : WideChar) :Bool;;
35877:  Func CharIn9( const c, t1, t2, t3 : WideChar) :Bool;;
35878:  Proc ConCatBuf(const Buffer: PChar;const CharCount:Card;var d:str;var InUse:Card);
35879: Proc ConCatBufA(const Buff:PChar;const AnsiCharCount:Card;var d:RawByteStr;var InUse:Card);
35880:  Proc ConCatChar( const c : Char; var d :Str; var InUse :Card);
35881:  Proc ConCatCharA( const c : AnsiChar; var d : RawByteString; var InUse :Card);
35882: // Proc ConCatCharW( const c : WideChar; var d : UnicodeString; var InUse :Card);
35883:  Proc ConCatStr( const s :Str; var d :Str; var InUse :Card);
35884:  Proc ConCatStrA( const s: RawByteString; var d : RawByteString; var InUse :Card);
35885:  //Proc ConCatStrW(const w:UnicodeString; var d : UnicodeString; var InUse :Card);
35886:  Func diCountBitsSet( const Value : Int) : Byte;
35887:  //Func Crc32OfStrA( const s : RawByteString) : TCrc32;
35888:  //Func Crc32OfStrW( const s : UnicodeString) : TCrc32;
35889:  Func CurrentDay : Word;
```

```
35890:  Func CurrentJulianDate : TJulianDate;
35891:  Func CurrentMonth : Word;
35892:  Func CurrentQuarter : Word;
35893:  Func diCurrentYear : Int;
35894:  Func DarkenColor( const Color : Int; const amount : Byte) : Int;
35895:  Func diDeleteFile( const FileName :Str) :Bool;
35896:  Func DeleteFileA( const FileName : Ansistr) :Bool;
35897:  //Func DeleteFileW( const FileName : UnicodeString) :Bool;
35898:  Func diDirectoryExists( const Dir :Str) :Bool;
35899:  Func DirectoryExistsA( const Dir : Ansistr) :Bool;
35900:  //Func DirectoryExistsW( const Dir : UnicodeString) :Bool;
35901:  Func diDiskFree( const Dir :Str) : Int64;
35902:  Func DiskFreeA( const Dir : Ansistr) : Int64;
35903:  //Func DiskFreeW( const Dir : UnicodeString) : Int64;
35904:  Func diExpandFileName( const FileName :Str) :Str;
35905:  Func ExpandFileNameA(const FileName: Ansistr): Ansistr;
35906:  //Func ExpandFileNameW( const FileName : UnicodeString) : UnicodeString;
35907:  Proc diExcludeTrailingPathDelimiter( var s :Str);
35908:  Proc ExcludeTrailingPathDelimiterA( var s : RawByteString);
35909:  //Proc ExcludeTrailingPathDelimiterW( var s : UnicodeString);
35910:  Func diExtractFileDrive( const FileName :Str) :Str;
35911:  Func ExtractFileDriveA( const FileName : RawByteString) : RawByteString;
35912:  //Func ExtractFileDriveW( const FileName : UnicodeString) : UnicodeString;
35913:  Func diExtractFileExt( const FileName :Str) :Str;
35914:  Func ExtractFileExtA( const FileName : RawByteString) : RawByteString;
35915:  //Func ExtractFileExtW( const FileName : UnicodeString) : UnicodeString;
35916:  Func diExtractFileName( const FileName :Str) :Str;
35917:  Func ExtractFileNameA(const FileName: Ansistr): Ansistr;
35918:  //Func ExtractFileNameW( const FileName : UnicodeString) : UnicodeString;
35919:  Func diExtractFilePath( const FileName :Str) :Str;
35920:  Func ExtractFilePathA( const FileName : RawByteString) : RawByteString;
35921:  //Func ExtractFilePathW( const FileName : UnicodeString) : UnicodeString;
35922:  Func ExtractNextWord10(const s:str;const ADelimiter:Char;var AStartIndex:Int):str;);
35923: Func ExtractNextWordA11(const s:RawByteStr;const ADelimiter:AnsiChar;var AStartIndex:Int): RawByteStr;
35924: Func ExtractNextWordW12(const s:UnicodeStr;const ADelimiter:WideChar;var AStartIndex:Int): UnicodeStr;
35925:  Func ExtractNextWord13(const s:str;const ADelims:TAnsiCharSet;var AStartIndex:Int):str;
35926:  Func ExtractNextWordA14(const s:RawByteString;const ADelimiters:TAnsiCharSet;var
       AStartIndex:Int):RawByteString;
35927:  Func diExtractWord(const Number:Card;const s:RawByteString;const Delimiters:TAnsiCharSet): RawByteString;
35928: Func ExtractWordA const Numb:Card;const s RawByteStr;const Delimiters:TAnsiCharSet): RawByteStr;
35929:  Func ExtractWordStartsA(const s:RawByteString;const MaxCharCount:Card;const WordSeparators:TAnsiCharSet):
       RawByteString;
35930:  Func diFileExists( const FileName :Str) :Bool;
35931:  Func FileExistsA( const FileName : Ansistr) :Bool;
35932:  //Func FileExistsW( const FileName : UnicodeString) :Bool;
35933:  Func diGCD( x, y :Card) :Card;
35934:  Func diGetTempFolder :Str;
35935:  Func GetTempFolderA : Ansistr;
35936:  //Func GetTempFolderW : UnicodeString;
35937:  Func diGetUserName( out UserName :Str) :Bool;
35938:  Func GetUserNameA( out UserName : Ansistr) :Bool;
35939:  //Func GetUserNameW( out UserName : UnicodeString) :Bool;
35940:  Func HexCodePointToInt( const c :Card) : Int;
35941:  Func diHexToInt( const s :Str) : Int;
35942:  Func HexToIntA( const s : RawByteString) : Int;
35943:  //Func HexToIntW( const s : UnicodeString) : Int;
35944:  Func BufHexToInt( p : PChar; l :Card) : Int;
35945:  Func BufHexToIntA( p : PChar; l :Card) : Int;
35946:  //Func BufHexToIntW( p : PWideChar; l :Card) : Int;
35947:  Proc IncludeTrailingPathDelimiterByRef( var s :Str);
35948:  Proc IncludeTrailingPathDelimiterByRefA(var s:RawByteString);
35949:  //Proc IncludeTrailingPathDelimiterByRefW(var w:UnicodeString);
35950:  Func IntToHex16(const Value: Int; const Digits : NativeInt) :Str;;
35951:  Func IntToHex17(const Value: Int64; const Digits : NativeInt) :Str;;
35952:  Func IntToHex18(const Value: UInt64; const Digits : NativeInt) :Str;;
35953:  Func IntToHexA( Value : UInt64; const Digits : NativeInt) : RawByteString;
35954:  //Func IntToHexW(Value : UInt64; const Digits : NativeInt) : UnicodeString;
35955:  Func IntToStrA19(const i: Int) : RawByteString;;
35956:  Func IntToStrW20(const i: Int) : UnicodeString;;
35957:  Func IntToStrA21(const i: Int64) : RawByteString;;
35958:  Func IntToStrW22(const i: Int64) : UnicodeString;;
35959:  Func CharDecomposeHangulW(const c: WideChar) : UnicodeString;
35960:  Func diIsPathDelimiter( const s :Str; const Index :Card) :Bool;
35961:  Func IsPathDelimiterA( const s : RawByteString; const Index :Card) :Bool;
35962:  //Func IsPathDelimiterW( const s : UnicodeString; const Index :Card) :Bool;
35963:  Func IsPointInRect( const Point : TPoint; const Rect : TRect) :Bool;
35964:  Func JulianDateToIsoDateStr(const Julian: TJulianDate):str;
35965:  Func JulianDateToIsoDateStrA( const Julian : TJulianDate) : RawByteString;
35966:  //Func JulianDateToIsoDateStrW( const Julian : TJulianDate) : UnicodeString;
35967:  Func LeftMostBit( Value :Card) : ShortInt;;
35968:  Func LeftMostBit2( Value : UInt64) : ShortInt;;
35969:  //Func MakeMethod( const AData, ACode : Pointer) : TMethod;
35970:  Func StrIsEmpty( const s :Str) :Bool;
35971:  Func StrIsEmptyA( const s : RawByteString) :Bool;
35972:  //Func StrIsEmptyW( const s : UnicodeString) :Bool;
35973:  Func PadLeftA(const Source:RawByteString;const Count:Card;const c:AnsiChar):RawByteStr;
35974:  //Func PadLeftW(const Source: UnicodeString; const Count :Card; const c : WideChar): UnicodeString;
35975:  Func PadRightA(const Source:RawByteString;const Count:Cardi;const c:AnsiChar):RawByteStr;
35976: //Func PadRightW(const Source:UnicodeString;const Count:Cardi;const c WideChar):UnicodeStr;
```

```
35977:  Func ProperCase( const s :Str) :Str;
35978:  Func ProperCaseA( const s : RawByteString) : RawByteString;
35979:  //Func ProperCaseW( const s : UnicodeString) : UnicodeString;
35980:  Proc ProperCaseByRefA( var s : RawByteString);
35981:  //Proc ProperCaseByRefW( var s : UnicodeString);
35982:  Func RegReadRegisteredOrganization( const Access : REGSAM) :Str;
35983:  Func RegReadRegisteredOrganizationA( const Access : REGSAM) : Ansistr;
35984:  //Func RegReadRegisteredOrganizationW( const Access : REGSAM) : UnicodeString;
35985:  Func RegReadRegisteredOwner( const Access : REGSAM) :Str;
35986:  Func RegReadRegisteredOwnerA( const Access : REGSAM) : Ansistr;
35987:  //Func RegReadRegisteredOwnerW( const Access : REGSAM) : UnicodeString;
35988:  Func RegReadStrDef( const Key : HKEY; const SubKey :Str; const ValueName:Str;const Default:str;const
       Access:REGSAM):Str;
35989:  Func RegReadStrDefA(const Key:HKEY;const SubKey:Ansistr;const ValueName:Ansistr;const Default:Ansistr;
       const Access:REGSAM):Ansistr;
35990:  Func StrDecodeUrlA( const Value : RawByteString): RawByteString;
35991:  Func StrEncodeUrlA( const Value : RawByteString): RawByteString;
35992:  Func diStrEnd( const s : PChar) : PChar;
35993:  Func StrEndA( const s : PChar) : PChar;
35994:  //Func StrEndW( const s : PWideChar) : PWideChar;
35995:  Proc StrIncludeTrailingChar( var s :Str; const c : Char);
35996:  Proc StrIncludeTrailingCharA( var s : RawByteString; const c : AnsiChar);
35997:  //Proc StrIncludeTrailingCharW( var s : UnicodeString; const c : WideChar);
35998:  Func diStrLen( const s : PChar) : NativeUInt;
35999:  Func StrLenA( const s : PChar) : NativeUInt;
36000:  //Func StrLenW( const s : PWideChar) : NativeUInt;
36001:  Func StrRandom(const ASeed:RawByteStr;const ACharacters:str;const ALength:Card):str);
36002:  Func StrRandomA(const ASeed:RawByteString;const ACharacters:RawByteString;const ALength:Card):RawByteStr;
36003:  Proc StrRemoveFromToIA(var Source:RawByteString;const FromString,ToString:RawByteString);
36004:  //Proc StrRemoveFromToIW(var Source:UnicodeString;const FromString, ToString : UnicodeString);
36005:  Proc StrRemoveSpacingA(var s:RawByteStr;const SpaceChars:TAnsiCharSet;const ReplaceChar:AnsiChar);
36006:  Proc diStrReplaceChar(var Source:Str;const SearchChar,ReplaceChar:Char);
36007:  Proc StrReplaceChar8(var s:Utf8String;const SearchChar,ReplaceChar:AnsiChar);
36008:  Proc StrReplaceCharA(var s:RawByteString;const SearchChar,ReplaceChar: AnsiChar);
36009:  //Proc StrReplaceCharW(var s:UnicodeString; const SearchChar, ReplaceChar : WideChar);
36010:  Func diStrReplace( const Source, Search, Replace :Str) :Str;
36011:  Func StrReplaceA( const Source, Search, Replace : RawByteString) : RawByteString;
36012:  //Func StrReplaceW( const Source, Search, Replace : UnicodeString) : UnicodeString;
36013:  Func StrReplaceI( const Source, Search, Replace :Str) :Str;
36014:  Func StrReplaceIA( const Source, Search, Replace : RawByteString) : RawByteString;
36015:  //Func StrReplaceIW( const Source, Search, Replace : UnicodeString) : UnicodeString;
36016:  Func StrReplaceLoopA( const Source, Search, Replace : RawByteString) : RawByteString;
36017:  //7Func StrReplaceLoopW( const Source, Search, Replace: UnicodeString) : UnicodeString;
36018:  Func StrReplaceLoopIA( const Source, Search, Replace : RawByteString) : RawByteString;
36019:  //Func StrReplaceLoopIW( const Source, Search, Replace: UnicodeString) : UnicodeString;
36020:  Func RightMostBit( const Value :Card) : ShortInt;;
36021:  Func RightMostBit2( const Value : UInt64) : ShortInt;;
36022:  Func LoadStrFromFile(const FileName:str; var s:RawByteString):Boolean;;
36023:  Func FileToStr( const FileName :Str; var s :Str) :Bool;;
36024:  Func LoadStrAFromFileA(const FileName:Ansistr;var s:RawByteString):Boolean;
36025:  //Func LoadStrAFromFileW(const FileName:UnicodeString; var s: RawByteString) :Bool;
36026:  Func LoadStrWFromFile28(const FileName:str;var s:UnicodeString):Boolean;;
36027:  Func LoadStrWFromFileA(const FileName: Ansistr;var s:UnicodeString):Bool;
36028:  //Func LoadStrWFromFileW(const FileName:UnicodeString;var s:UnicodeString):Bool;
36029:  Func QuotedStrW( const s : UnicodeString; const Quote : WideChar) : UnicodeString;
36030:  Func SaveStrToFile( const s :Str; const FileName :Str) :Bool;
36031:  Func StrToFile( const s :Str; const FileName :Str) :Bool;
36032:  Func SaveStrAToFile(const s:RawByteString;const FileName :Str) :Bool;
36033:  Func SaveStrAToFileA( const s : RawByteString; const FileName : Ansistr) :Bool;
36034:  //Func SaveStrAToFileW(const s:RawByteString; const FileName: UnicodeString) :Bool;
36035:  Func SaveStrWToFile(const s:UnicodeString;const FileName :Str):Boolean;
36036:  Func SaveStrWToFileA(const s: UnicodeString;const FileName:Ansistr): Bool;
36037:  //Func SaveStrWToFileW(const s:UnicodeString; const FileName : UnicodeString):Bool;
36038:  Func StrPosChar(const Source:str; const c:Char;const Start:Card):Card;
36039:  Func StrPosCharA(const Source:RawByteString;const c:AnsiChar;const Start:Car):Card;
36040:  //Func StrPosCharW(const Source:UnicodeStr;const c:WideChar;const Start:Cardi):Card;
36041:  Func StrPosCharBack(const Source:Str;const c:Char;const Start:Card):Card;
36042:  Func StrPosCharBackA(const Source:RawByteStr;const c:AnsiChar;const Start:Cardi):Card);
36043: Func StrPosCharsA(const Sourc:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
36044:  Func StrPosCharsBackA(const Src:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
36045:  Func StrPosNotCharsA(const Src:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
36046:  Func StrPosNotCharsBackA(const Source:RawByteString;const Search:TAnsiCharSet; const Start:Card):Card;
36047:  Func SetFileDate(const FileHandle:THandle;const Year:Int;const Month,Day:Word):Bool);
36048:  Func SetFileDate2( const FileName :Str; const JulianDate : TJulianDate) :Bool;
36049:  Func SetFileDateA( const FileName: Ansistr; const JulianDate:TJulianDate) :Bool;
36050:  //Func SetFileDateW(const FileName:UnicodeString;const JulianDate:TJulianDate):Boolean;
36051:  Func SetFileDateYmd(const FileName:str;const Year:Int;const Month,Day:Word):Bool;
36052:  Func SetFileDateYmdA(const FileName:Ansistr;const Year:Int;const Month,Day:Word):Bool;
36053:  Func SetFileDateYmdW(const FileName:UnicodeStr;const Year:Int;const Month,Day:Word):Bool;
36054:  Func StrContainsChar(const s:str; const c:Char; const Start :Card) :Bool;
36055:  Func StrContainsCharA(const s:RawByteString;const c:AnsiChar;const Start:Card):Bool;
36056:  //Func StrContainsCharW(const s:UnicodeString;const c:WideChar;const Start:Cardi):Bool;
36057:  Func StrContainsCharsA(const s:RawByteStr;const Chars:TAnsiCharSet;const Start:Card):Bool;
36058:  Func diStrSame( const s1, s2 :Str) :Bool;
36059:  Func StrSameA( const s1, s2 : RawByteString) :Bool;
36060:  //Func StrSameW( const s1, s2 : UnicodeString) :Bool;
36061:  Func StrSameI( const s1, s2 :Str) :Bool;
36062:  Func StrSameIA( const s1, s2 : RawByteString) :Bool;
36063:  //Func StrSameIW( const s1, s2 : UnicodeString) :Bool;
```

```
36064:  Func StrSameStart( const s1, s2 :Str) :Bool;
36065:  Func StrSameStartA( const s1, s2 : RawByteString) :Bool;
36066:  //Func StrSameStartW( const s1, s2 : UnicodeString) :Bool;
36067:  Func StrSameStartI( const s1, s2 :Str) :Bool;
36068:  Func StrSameStartIA( const s1, s2 : RawByteString) :Bool;
36069:  //Func StrSameStartIW( const s1, s2 : UnicodeString) :Bool;
36070:  Func diStrComp( const s1, s2 :Str) : Int;
36071:  Func StrCompA( const s1, s2 : RawByteString) : Int;
36072:  //Func StrCompW( const s1, s2 : UnicodeString) : Int;
36073:  Func StrCompI( const s1, s2 :Str) : Int;
36074:  Func StrCompIA( const s1, s2 : RawByteString) : Int;
36075:  //Func StrCompIW( const s1, s2 : UnicodeString) : Int;
36076:  Func StrCompNum( const s1, s2 :Str) : Int;
36077:  Func StrCompNumA( const s1, s2 : RawByteString) : Int;
36078: // Func StrCompNumW( const s1, s2 : UnicodeString) : Int;
36079:  Func StrCompNumI( const s1, s2 :Str) : Int;
36080:  Func StrCompNumIA( const s1, s2 : RawByteString) : Int;
36081: // Func StrCompNumIW( const s1, s2 : UnicodeString) : Int;
36082:  Func StrContains(const Search,Source:str; const Start:Card):Bool;
36083:  Func StrContainsA(const Search,Source:RawByteString;const Start:Card):Bool;
36084:  //Func StrContainsW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Bool;
36085:  Func StrContainsI(const Search, Source :Str; const Start :Card) :Bool;
36086:  Func StrContainsIA(const Search,Source:RawByteString; const Start:Card):Bool;
36087:  //Func StrContainsIW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Bool;
36088:  Func StrCountChar(const ASource:str;const c:Char;const AStartIdx:Card):Card;
36089:  Func StrCountCharA(const ASource:RawByteStr;const c:AnsiChar;const AStartIdx:Cardi):Card;
36090:  Func StrMatchesA(const Search,Source:RawByteString; const AStartIdx:Card):Bool;
36091:  Func StrMatchesIA(const Search,Source:RawByteString;const AStartIdx:Card):Bool;
36092:  Func StrMatchWild(const Source,Mask:str;const WildChar:Char;const MaskChar:Char):Bool;
36093:  Func StrMatchWildA(const Src,Mask:RawByteString;const WildChr:AnsiChar;const MaskChr:AnsiChar):Bool;
36094:  Func StrMatchWildI(const Source,Mask:str;const WildChar:Char;const MaskChar:Char):Bool;
36095:  Func StrMatchWildIA(const Source,Mask:RawByteString;const WildChar:AnsiChar;const MaskChar:AnsiChar):Bool;
36096:  Func diStrPos( const ASearch, ASource :Str; const AStartPos :Card) :Card;
36097:  Func StrPosA(const ASearch,ASource:RawByteString;const AStartPos:Card):Card;
36098:  //Func StrPosW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Card;
36099:  Func StrPosI( const ASearch, ASource :Str; const AStartPos :Card) :Card;
36100:  Func StrPosIA(const ASearch,ASource:RawByteString; const AStartPos:Card):Card;
36101:  //Func StrPosIW(const ASearch, ASource:UnicodeString;const AStartPos:Card):Card;
36102:  Func StrPosBackA( const ASearch, ASource: RawByteString; AStart :Card) :Card;
36103:  Func StrPosBackIA(const ASearch, ASource: RawByteString; AStart :Card) :Card;
36104:  //Func StrToIntDefW( const w : UnicodeString; const Default : Int) : Int;
36105:  Func StrToInt64DefW( const w : UnicodeString; const Default : Int64) : Int64;
36106:  Func StrToUpper( const s :Str) :Str;
36107:  Func StrToUpperA( const s : RawByteString) : RawByteString;
36108:  //Func StrToUpperW( const s : UnicodeString) : UnicodeString;
36109:  Proc StrToUpperInPlace( var s :Str);
36110:  Proc StrToUpperInPlaceA( var s : Ansistr);
36111:  //Proc StrToUpperInPlaceW31( var s : WideString);;
36112:  //Proc StrToUpperInPlaceW32( var s : UnicodeString);;
36113:  Func StrToLower( const s :Str) :Str;
36114:  Func StrToLowerA( const s : RawByteString) : RawByteString;
36115:  //Func StrToLowerW( const s : UnicodeString) : UnicodeString;
36116:  Proc StrToLowerInPlace( var s :Str);
36117:  Proc StrToLowerInPlaceA( var s : Ansistr);
36118:  //Proc StrToLowerInPlaceW33( var s : WideString);;
36119:  //Proc StrToLowerInPlaceW34( var s : UnicodeString);;
36120:  Proc StrTimUriFragmentA( var Value : RawByteString);
36121:  //Proc StrTrimUriFragmentW( var Value : UnicodeString);
36122:  //Func StrExtractUriFragmentW( var Value : UnicodeString): UnicodeString;
36123:  Func StrCountUtf8Chars(const AValue: Utf8String) :Card;
36124:  Func StrDecodeUtf8( const AValue: Utf8String): UnicodeString;
36125:  Func StrEncodeUtf8( const AValue: UnicodeString): Utf8String;
36126:  Func diSysErrorMessage( const MessageID :Card) :Str;
36127:  Func SysErrorMessageA(const MessageID:Card): Ansistr;
36128:  //Func SysErrorMessageW( const MessageID :Card) : UnicodeString;
36129:  Func TextExtentW( const DC : HDC; const Text : UnicodeString) : TSize;
36130:  Func TextHeightW( const DC : HDC; const Text : UnicodeString) : Int;
36131:  Func TextWidthW( const DC : HDC; const Text : UnicodeString) : Int;
36132:  Func diStrTrim( const Source :Str) :Str;
36133:  Func StrTrimA( const Source : RawByteString) : RawByteString;
36134:  //Func StrTrimW( const w : UnicodeString) : UnicodeString;
36135:  Func StrTrimCharA(const Source:RawByteString;const CharToTrim:AnsiChar): RawByteString;
36136:  Func StrTrimCharsA(const Source:RawByteStr;const CharsToTrim:TAnsiCharSet):RawByteString;
36137: //Func StrTrimCharsW(const s:UnicodeStr;const IsCharToTrim:TValidateCharFuncW):UnicodeStr;
36138:  Proc TrimLeftByRefA( var s : RawByteString; const Chars : TAnsiCharSet);
36139:  Func TrimRightA( const Source : RawByteString; const s : TAnsiCharSet) : RawByteString;
36140:  Proc TrimRightByRefA( var Source : RawByteString; const s : TAnsiCharSet);
36141:  Proc StrTrimCompressA(var s:RawByteString;const TrimCompressChars:TAnsiCharSet;const
       ReplaceChar:AnsiChar);
36142:  Func TryStrToIntW( const w : UnicodeString; out Value : Int) :Bool;
36143:  Func TryStrToInt64W( const w : UnicodeString; out Value : Int64) :Bool;
36144:  Func ValInt( const p : PChar; const l : Int; out Code : Int) : Int;;
36145:  Func ValIntA36( p : PChar; l : Int; out Code : Int) : Int;;
36146:  //Func ValIntW37( p : PWideChar; l : Int; out Code : Int) : Int;;
36147:  Func ValInt2(const s:Str; out Code : Int): Int;;
36148:  Func ValIntA39( const s : RawByteString; out Code : Int) : Int;;
36149:  Func ValIntW40( const s : UnicodeString; out Code : Int) : Int;;
36150:  Func ValInt64A41( p : PChar; l : Int; out Code : Int) : Int64;;
36151:  //Func ValInt64W42( p : PWideChar; l : Int; out Code : Int) : Int64;;
```

```
36152:  Func ValInt64A43( const s : RawByteString; out Code : Int) : Int64;;
36153:  Func ValInt64W44( const s : UnicodeString; out Code : Int) : Int64;;
36154:  Func YmdToIsoDateStr(constYear:Int;const Month:Word;const Day:Word):str;
36155:  Func YmdToIsoDateStrA(const Year:Int;const Month:Word; const Day:Word):RawByteString;
36156:  Func YmdToIsoDateStrW(const Year:Int;const Month:Word; const Day:Word):UnicodeString;
36157:  Func CharIsLetterW( const c : WideChar) :Bool;
36158:  Func CharIsLetterCommonW( const c : WideChar) :Bool;
36159:  Func CharIsLetterUpperCaseW(const c : WideChar) :Bool;
36160:  Func CharIsLetterLowerCaseW(const c : WideChar) :Bool;
36161:  Func CharIsLetterTitleCaseW(const c : WideChar) :Bool;
36162:  Func CharIsLetterModifierW(const c : WideChar) :Bool;
36163:  Func CharIsLetterOtherW(const c : WideChar) :Bool;
36164:  Func CharIsMarkW( const c : WideChar) :Bool;
36165:  Func CharIsMarkNon_SpacingW( const c : WideChar) :Bool;
36166:  Func CharIsMarkSpacing_CombinedW(const c: WideChar):Bool;
36167:  Func CharIsMarkEnclosingW( const c : WideChar) :Bool;
36168:  Func CharIsNumberW( const c : WideChar) :Bool;
36169:  Func CharIsNumber_DecimalW( const c : WideChar) :Bool;
36170:  Func CharIsNumber_LetterW( const c : WideChar) :Bool;
36171:  Func CharIsNumber_OtherW( const c : WideChar) :Bool;
36172:  Func CharIsPunctuationW( const c : WideChar) :Bool;
36173:  Func CharIsPunctuation_ConnectorW( const c : WideChar) :Bool;
36174:  Func CharIsPunctuation_DashW( const c : WideChar) :Bool;
36175:  Func CharIsPunctuation_OpenW( const c : WideChar) :Bool;
36176:  Func CharIsPunctuation_CloseW( const c : WideChar) :Bool;
36177:  Func CharIsPunctuation_InitialQuoteW( const c : WideChar) :Bool;
36178:  Func CharIsPunctuation_FinalQuoteW( const c : WideChar) :Bool;
36179:  Func CharIsPunctuation_OtherW( const c : WideChar) :Bool;
36180:  Func CharIsSymbolW( const c : WideChar) :Bool;
36181:  Func CharIsSymbolMathW( const c : WideChar) :Bool;
36182:  Func CharIsSymbolCurrencyW( const c : WideChar) :Bool;
36183:  Func CharIsSymbolModifierW( const c : WideChar) :Bool;
36184:  Func CharIsSymbolOtherW( const c : WideChar) :Bool;
36185:  Func CharIsSeparatorW( const c : WideChar) :Bool;
36186:  Func CharIsSeparatorSpaceW( const c : WideChar) :Bool;
36187:  Func CharIsSeparatorLineW( const c : WideChar) :Bool;
36188:  Func CharIsSeparatorParagraphW( const c : WideChar) :Bool;
36189:  Func CharIsOtherW( const c : WideChar) :Bool;
36190:  Func CharIsOtherControlW( const c : WideChar) :Bool;
36191:  Func CharIsOtherFormatW( const c : WideChar) :Bool;
36192:  Func CharIsOtherSurrogateW( const c : WideChar) :Bool;
36193:  Func CharIsOtherPrivateUseW( const c : WideChar) :Bool;
36194:  Func BitClear( const Bits, BitNo : Int) : Int;
36195:  Func BitSet( const Bits, BitIndex : Int) : Int;
36196:  Func BitSetTo( const Bits, BitIndex : Int; const Value :Bool) : Int;
36197:  Func BitTest( const Bits, BitIndex : Int) :Bool;
36198:  Func CharCanonicalCombiningClassW( const Char : WideChar) :Card;
36199:  Func CharIsAlphaW( const c : WideChar) :Bool;
36200:  Func CharIsAlphaNumW( const c : WideChar) :Bool;
36201:  Func CharIsCrLf( const c : Char) :Bool;
36202:  Func CharIsCrLfA( const c : AnsiChar) :Bool;
36203:  //Func CharIsCrLfW( const c : WideChar) :Bool;
36204:  Func diCharIsDigit( const c : Char) :Bool;
36205:  Func CharIsDigitA( const c : AnsiChar) :Bool;
36206:  //Func CharIsDigitW( const c : WideChar) :Bool;
36207:  Func CharIsHangulW( const Char : WideChar) :Bool;
36208:  Func CharIsHexDigitW( const c : WideChar) :Bool;
36209:  Func CharIsWhiteSpaceW( const c : WideChar) :Bool;
36210:  Func CharToCaseFoldW( const Char : WideChar) : WideChar;
36211:  Func CharToLowerW( const Char : WideChar) : WideChar;
36212:  Func CharToUpperW( const Char : WideChar) : WideChar;
36213:  Func CharToTitleW( const Char : WideChar) : WideChar;
36214:  Func DayOfJulianDate( const JulianDate : TJulianDate) : Word;
36215:  Func diDayOfWeek( const JulianDate : TJulianDate) : Word;
36216:  Func DayOfWeekYmd( const Year : Int; const Month, Day : Word) : Word;
36217:  Func diDaysInMonth( const JulianDate : TJulianDate) : Word;
36218:  Func DaysInMonthYm( const Year : Int; const Month : Word) : Word;
36219:  Proc DecDay( var Year : Int; var Month, Day : Word);
36220:  Proc DecDays( var Year : Int; var Month, Day : Word; const Days : Int);
36221:  Func diDeleteDirectory( const Dir :Str; const DeleteItself :Bool) :Bool;
36222:  Func DeleteDirectoryA( Dir : Ansistr; const DeleteItself :Bool) :Bool;
36223:  //Func DeleteDirectoryW( Dir : UnicodeString; const DeleteItself :Bool) :Bool;
36224:  Func diEasterSunday( const Year : Int) : TJulianDate;
36225:  Proc EasterSundayYmd( const Year : Int; out Month, Day : Word);
36226:  Func diFirstDayOfWeek( const JulianDate : TJulianDate) : TJulianDate;
36227:  Proc FirstDayOfWeekYmd( var Year : Int; var Month, Day : Word);
36228:  Func diFirstDayOfMonth( const Julian : TJulianDate) : TJulianDate;
36229:  Proc FirstDayOfMonthYmd( const Year : Int; const Month : Word; out Day : Word);
36230:  Func diForceDirectories( const Dir :Str) :Bool;
36231:  Func ForceDirectoriesA( Dir : Ansistr) :Bool;
36232:  //Func ForceDirectoriesW( Dir : UnicodeString) :Bool;
36233:  Proc FreeMemAndNil( var Ptr: TObject);
36234:  Func diGetCurrentFolder :Str;
36235:  Func GetCurrentFolderA : Ansistr;
36236:  //Func GetCurrentFolderW : UnicodeString;
36237:  Proc SetCurrentFolder( const NewFolder :Str);
36238:  Proc SetCurrentFolderA( const NewFolder : Ansistr);
36239:  //Proc SetCurrentFolderW( const NewFolder : UnicodeString);
36240:  Func diGetDesktopFolder :Str;
```

```
36241:  Func GetDesktopFolderA : Ansistr;
36242:  //Func GetDesktopFolderW : UnicodeString;
36243:  Func diGetFileSize( const AFileName :Str) : Int64;
36244:  Func GetFileSizeA( const AFileName : Ansistr) : Int64;
36245:  //Func GetFileSizeW( const AFileName : UnicodeString) : Int64;
36246:  Func diGetDesktopDirectoryFolder :Str;
36247:  Func GetDesktopDirectoryFolderA : Ansistr;
36248:  //Func GetDesktopDirectoryFolderW : UnicodeString;
36249:  Func GetFileLastWriteTime(const FileName:Str; out FileTime : TFileTime) :Bool;
36250:  Func GetFileLastWriteTimeA(const FileName:Ansistr; out FileTime:TFileTime):Boolean;
36251:  //Func GetFileLastWriteTimeW(const FileName:UnicodeString;out FileTime:TFileTime):Bool;
36252:  Func diGetPersonalFolder( const PersonalFolder : Int) :Str;
36253:  Func GetPersonalFolderA : Ansistr;
36254:  //Func GetPersonalFolderW : UnicodeString;
36255:  Func GetSpecialFolder( const SpecialFolder : Int) :Str;
36256:  Func GetSpecialFolderA( const SpecialFolder : Int) : Ansistr;
36257:  //Func GetSpecialFolderW( const SpecialFolder : Int) : UnicodeString;
36258:  Proc diIncMonth( var Year : Int; var Month, Day : Word);
36259:  Proc diIncMonths(var Year :Int; var Month, Day:Word; const NumberOfMonths:Int);
36260:  Proc diIncDay( var Year : Int; var Month, Day : Word);
36261:  Proc IncDays( var Year : Int; var Month, Day : Word; const Days : Int);
36262:  Func IsDateValid( const Year : Int; const Month, Day : Word) :Bool;
36263:  Func IsHolidayInGermany( const Julian : TJulianDate) :Bool;
36264:  Func IsHolidayInGermanyYmd( const Year : Int; const Month, Day : Word) :Bool;
36265:  Func diIsLeapYear( const Year : Int) :Bool;
36266:  Func ISODateToJulianDate( const ISODate : TIsoDate) : TJulianDate;
36267:  Proc ISODateToYmd(const ISODate: TIsoDate; out Year : Int; out Month, Day : Word);
36268:  Func IsCharLowLineW( const c : WideChar) :Bool;
36269:  Func IsCharQuoteW( const c : WideChar) :Bool;
36270:  Func IsShiftKeyDown :Bool;
36271:  Func IsCharWhiteSpaceOrAmpersandW( const c : WideChar) :Bool;
36272:  Func IsCharWhiteSpaceOrNoBreakSpaceW( const c : WideChar) :Bool;
36273:  Func IsCharWhiteSpaceOrColonW( const c : WideChar) :Bool;
36274:  Func CharIsWhiteSpaceGtW( const c : WideChar) :Bool;
36275:  Func CharIsWhiteSpaceLtW( const c : WideChar) :Bool;
36276:  Func CharIsWhiteSpaceHyphenW( const c : WideChar) :Bool;
36277:  Func CharIsWhiteSpaceHyphenGtW( const c : WideChar) :Bool;
36278:  Func IsCharWordSeparatorW( const c : WideChar) :Bool;
36279:  Func diISOWeekNumber( const JulianDate : TJulianDate) : Word;
36280:  Func ISOWeekNumberYmd( const Year : Int; const Month, Day : Word) : Word;
36281: Func ISOWeekToJulianDate(const Year:Int;const WeekOfYear:Word;const DayOfWeek:Word):TJulianDate);
36282:  Func JulianDateIsWeekDay( const JulianDate : TJulianDate) :Bool;
36283:  Func JulianDateToIsoDate( const Julian : TJulianDate) : TIsoDate;
36284:  Proc JulianDateToYmd(const JulianDate:TJulianDate;out Year: Int;out Month,Day:Word);
36285:  Func LastDayOfMonth( const JulianDate : TJulianDate) : TJulianDate;
36286:  Proc LastDayOfMonthYmd( const Year : Int; const Month : Word; out Day : Word);
36287:  Func LastDayOfWeek( const JulianDate : TJulianDate) : TJulianDate;
36288:  Proc LastDayOfWeekYmd( var Year : Int; var Month, Day : Word);
36289:  Func LastSysErrorMessage :Str;
36290:  Func LastSysErrorMessageA : Ansistr;
36291:  Func LastSysErrorMessageW : UnicodeString;
36292:  Func diMax( const a : Int; const b : Int) : Int;;
36293:  Func diMax3( const a, b, c : Int) : Int;
36294:  Func MaxCard( const a :Card; const b :Card) :Card;;
36295:  Func MaxCard3( const a :Card; const b :Card; const c :Card):Card;;
36296:  Func diMaxint64( const a : Int64; const b : Int64) : Int64;;
36297:  Func dimaxint643( const a : Int64; const b : Int64; const c : Int64) : Int64;;
36298:  Func diMin( const a, b : Int) : Int;;
36299:  Func diMin3( const a, b, c : Int) : Int;
36300:  Func MinCard( const a, b :Card) :Card;;
36301:  Func Mincard3( const a, b, c :Card) :Card;;
36302:  Func diMinint64( const a, b : Int64) : Int64;;
36303:  Func diMinint643( const a, b, c : Int64) : Int64;;
36304:  Func diMinint64U( const a, b : UInt64) : UInt64;;
36305:  Func diMinint643U( const a, b, c : UInt64) : UInt64;;
36306:  Func MonthOfJulianDate( const JulianDate : TJulianDate) : Word;
36307:  Func YearOfJuilanDate( const JulianDate : TJulianDate) : Int;
36308:  Func YmdToIsoDate( const Year : Int; const Month, Day : Word) : TIsoDate;
36309:  Func YmdToJulianDate( const Year : Int; const Month, Day : Word) : TJulianDate;
36310: end;
36311:
36312:  Func FiboMaxE2(n: Int): Extended;
36313:   begin
36314:     result:= (pow((1+SQRT5)/2,n)-pow((1-SQRT5)/2,n))/SQRT5
36315:   end;
36316:  TDLLVersionInfo=Record
36317:     cbSize,   // Size of the structure, in bytes.
36318:     dwMajorVersion, // Major version of  DLL
36319:     dwMinorVersion, // Minor version of  DLL
36320:     dwBuildNumber, // Build number of  DLL
36321:     dwPlatformID: DWord; //Identifies the platform for which the DLL was built
36322:  end;
36323:
36324:  {$IFDEF MSWINDOWS}
36325: Proc TIdAntiFreeze_Process;
36326: var Msg: TMsg;
36327:   ApplicationHasPriority:Bool;
36328: begin
36329:   if ApplicationHasPriority then begin
```

```
36330:     Application.ProcessMessages;
36331:   end else begin
36332:     // This guarantees it will not ever call Application.Idle
36333:     if PeekMessage(Msg, 0, 0, 0, PM_NOREMOVE) then begin
36334:       Application.HandleMessage;
36335:     end;
36336:   end;
36337: end;
36338: {$ENDIF}
36339:
36340: with lockbox you do sha1 with strings and hex:
36341: writeln(SHA1ofStr(loadstringj(exepath+'maxbox4.exe')))
36342: // ¢'ÞRL'mÍÕïÝ§1ÎÓõî
36343: tempstr:= SHA1ofStr(loadstringj(exepath+'maxbox4.exe'))
36344: writeln(' ')
36345: for i:= 1 to 20 do
36346:   Write(charToHexStr(tempstr[i]));
36347:   A291DE524C1B916DCDD502EFDDA76C0BCED3F0EE
36348:
36349: Example with WMI:
36350: Func TestWMIOS2:Str;
36351: var wmiService, computerName, objectsList: OLEvariant;
36352:   sysID:Str;
36353:   isloc: ISWBemLocator; isser: ISWBemServices;
36354:   iset: ISWbemObjectSet; ENum: IEnumVariant;
36355:   tempObj: OleVariant;
36356: begin
36357:   computerName := '.';
36358:   isloc:= WMIStart;
36359:   isser:= WMIConnect2(isloc, '\root\cimv2','maxbox10','max',''); //: ISWBemServices;
36360:   //Exception: incorrect credentials. WMI connection failed.
36361:   //Set wmiService = GetObject("WinMgmts:{impersonationLevel=impersonate, "&_
36362:   // "(SystemTime, Shutdown)}!\\" & computerName & "\root\cimv2")
36363:   //Set objectsList = wmiService.InstancesOf("Win32_OperatingSystem")
36364:   //For Each ObjService in objectsList
36365:
36366:   iset:= WMIExecQuery(isser,'Select * from Win32_OperatingSystem')
36367:   result:=(botoStr(WMIRowFindFirst(iset, ENum, tempObj)));
36368:   repeat
36369:         sysid:= ('OSid: '+vartoStr(tempobj.csname)+
36370:         ' - '+vartoStr(tempobj.name)+
36371:         ' - '+vartoStr(tempobj.caption))
36372:   until not WMIRowFindNext(ENum, tempobj); //}
36373:   tempObj:= unassigned;
36374:   result:= result+' '+sysID;
36375: //Call Log.Message("SysInfo", sysID)
36376: End; // Sub
36377:
36378: >>> Sysid2: TRUE OSid: MAXBOX10 - Microsoft Windows 10 Home|C:\WINDOWS|\Device\Harddisk0\Partition4 -
       Microsoft Windows 10 Home
36379:   command1:= 'play "'+songpath+'maxbox.wav"'; command2:= 'play "'+songpath+'moon.wav"';
36380:   SendMCICommand('open waveaudio shareable;   //parallels
36381:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav";
36382:   SendMCICommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav";
36383:   SendMCICommand('close waveaudio;
36384: //MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
36385:   if Not fileexists(exepath+'\mytest1.wav') then begin
36386:     Writeln('Generate 2 sounds with 440 and 443 Hz');
36387:     MakeSound(440,5000, 80, Exepath+'\mytest1.wav');
36388:     MakeSound(443,5000, 80, Exepath+'\mytest2.wav');
36389:   end else begin
36390:     Writeln('Play Beat of 2 sounds together with 3 Hz Amp Mod:');
36391:     mciSendString('PLAY "'+exepath+'\mytest1.wav'+'"', 'Nil', 0, 0);
36392:     mciSendString('PLAY "'+exepath+'\mytest2.wav'+'"', 'Nil', 0, 0);
36393:   end;
36394:
36395: Strings (and other non ordinals) in Case Statements
36396: This has to be most requested feature by far in history of D imho, sure to make many long time D fans
       happy.}
36397:   writeln(floattostr(pow(256,200000)))
36398:     //writeln(itoa(length(bigpow(256,200000))))   //481648
36399:   sr:= sworld;
36400:   case sr of
36401:     'hello' : xi := 1;
36402:     'goodbye' : xi := 2;
36403:     sWorld : xi := 3; //sWorld is a string constant
36404:   end;
36405:   writeln('res of case '+itoa(xi));
36406:
36407: initialization  XRaise
36408:
36409: type
36410:  EPointsArrayIsNotAssigned = Exception;
36411:  writeln(itoa(length(aptsarr)));
36412:  setlength(aptsarr, 2)
36413:  writeln(itoa(length(aptsarr)));
36414:  if length(aptsarr) = 0 then begin
36415:    Xraise(EPointsArrayIsNotAssigned.Create('Points array is not assigned...'));
36416:    //Exit;
```

```
36417:  end;
36418:
36419: maXbox SSL fox471 for
36420: The first step the following components, maybe we will add further in the future:
36421: 1. Orpheus (Win32 and Win64)
36422: 2. Abbrevia (Win32, Win64, MacOSX, iOS and Android)
36423: 3. Virtual Tree (Win32 and Win64)
36424: 4. SynEdit (Win32 and Win64)
36425: 5. LockBox (Win32, Win64, MacOSX, iOS and Android)
36426: 6. Async Professional (Win32)
36427: 7. PowerPDF (Win32 and Win64)
36428:
36429: ////////////////////////////////////////////////////////////////////////////
36430: All maXbox Tutorials Table of Content 2014/2015/2016/2017/2018/2019/2020/2021/2022
36431: ////////////////////////////////////////////////////////////////////////////
36432:   Tutorial 00 Function-Coding  (Blix the Programmer)
36433:   Tutorial 01 Procedural-Coding
36434:   Tutorial 02 OO-Programming
36435:   Tutorial 03 Modular Coding
36436:   Tutorial 04 UML Use Case Coding
36437:   Tutorial 05 Internet Coding
36438:   Tutorial 06 Network Coding
36439:   Tutorial 07 Game Graphics Coding
36440:   Tutorial 08 Operating System Coding
36441:   Tutorial 09 Database Coding
36442:   Tutorial 10 Statistic Coding
36443:   Tutorial 10_1 Probability Coding
36444:   Tutorial 11 Forms Coding
36445:   Tutorial 12 SQL DB Coding
36446:   Tutorial 13 Crypto Coding
36447:   Tutorial 14 Parallel Coding
36448:   Tutorial 15 Serial RS232 Coding
36449:   Tutorial 16 Event Driven Coding
36450:   Tutorial 17 Web Server Coding
36451:   Tutorial 18 Arduino System Coding
36452:   Tutorial 18_3 RGB LED System Coding
36453:   Tutorial 19 WinCOM /Arduino Coding
36454:   Tutorial 20 Regular Expressions RegEx
36455:   Tutorial 21 Android Coding (coming 2013)
36456:   Tutorial 22 Services Programming
36457:   Tutorial 23 Real Time Systems
36458:   Tutorial 24 Clean Code
36459:   Tutorial 25 maXbox Configuration I+II
36460:   Tutorial 26 Socket Programming with TCP
36461:   Tutorial 27 XML & TreeView
36462:   Tutorial 28 DLL Coding (available)
36463:   Tutorial 29 UML Scripting (2014)
36464:   Tutorial 30 Web of Things (2014)
36465:   Tutorial 31 Closures (2014)
36466:   Tutorial 32 SQL Firebird (2014)
36467:   Tutorial 33 Oscilloscope (available)
36468:   Tutorial 34 GPS Navigation (2014)
36469:   Tutorial 35 Web Box (available)
36470:   Tutorial 36 Unit Testing (avail)
36471:   Tutorial 37 API Coding (avail)
36472:   Tutorial 38 3D Coding (coming 2015)
36473:   Tutorial 39 GEO Map Coding (available)
36474:   Tutorial 39_1 GEO Map Layers Coding (available)
36475:   Tutorial 40 REST Coding (coming 2015)
36476:   Tutorial 41 Big Numbers Coding (2015)
36477:   Tutorial 42 Parallel Processing (2015)
36478:   Tutorial 43 Code Metrics: June2016
36479:   Tutorial 44 IDE Extensions
36480:   Tutorial 45 Robotics: July2016
36481:   Tutorial 46 WineHQ: Dez2016
36482:   Tutor 47 RSA Crypto Jan2017
36483:   Tutor 48 Microservice Jan2017
36484:   http://www.softwareschule.ch/download/maxbox_starter48.pdf
36485:   Tutorial 49 Refactoring: March 2017
36486:   Tutorial 50 Big Numbers II: April 2017
36487:   Tutorial 51 5 Use Cases April 2017
36488:   Tutorial 52 Work with WMI Mai 2017
36489:   Tutorial 53 Real Time UML 2017
36490:   Tutor 54 Microservice II Nov 2017
36491:   Tutor 55 ASCII Talk Dez 2017
36492:   Tutor 56 Neural Network 2018
36493:   Tutor 57 Neural Network II
36494:   Tutor 58 Data Science
36495:   Tutor 59 Big Data Feb 2018
36496:   Tutor 60 Machine Learning March 2018
36497:   Tutor 60.1 Sentiment Analysis
36498:   Tutor 60.2 ML II
36499:   Tutor 63 Machine Games
36500:   Tutor 64 Install Routines
36501:   Tutor 65 Machine Learning III
36502:   Tutor 66 Machine Learning IV
36503:   Tutor 67 Machine Learning V
36504:   Tutor 68 Machine Learning VI
36505:   Tutor 69 Machine Learning VII
```

```
36506:   Tutor 70 No GUI Shell
36507:   Tutor 71 CGI and WebSocket
36508:   Tutor 72 Multilanguage
36509:   Tutor 73 EKON 24 Edition
36510:   Tutor 74 BASTA 2020 Vision
36511:   Tutor 75 Object Detection
36512:   Tutor 76 Image Classifier CAI
36513:   Tutor 77 Image Classifier Matrix
36514:   Tutor 78 Portable pixmap format(PPM)
36515:   Tutor 79 Unit Testing with Asserts
36516:   Tutor 80 My Tips & Tricks
36517:   Tutor 81 RSS Feeds of BBC News - Digital Forensic March 2021
36518:   Tutorial 82 JSON Code API
36519:   Tutorial 82_2 Sentiment NLTK API Code
36520:   #Tutorial 83 Unicode + Bayes Theorem & Confusion Matrix April 2021
36521:   #Tutorial 84 JSON Primer Mai 2021
36522:   Tutorial 83 Machine Learning XI Classification April 2021
36523:   Tutorial 84 Machine Learning XII Baseline Mai 2021
36524:   Tutorial 85 JSON4Delphi Automation Mai 2021
36525:   Tutorial 86 Python4Delphi P4D June 2021 (Part 1-4)
36526:   Tutorial 87 Image Detection with Lazarus
36527:   Tutorial 88 Performance Tuning
36528:   Tutorial 89 CAI mX4
36529:   Tutorial 89_1 CAI mX4 model define
36530:   Tutorial 89_2 CAI mX4 model fits
36531:   Tutorial 90 Python CheatSheet Nov. 2021
36532:   Tutorial 91 CAI CheatSheet Faker SynDat Dez. 2021
36533:   Tutorial 92 Python_CAI CheatSheet Jan. 2022
36534:   Tutorial 92_1 VCL4Python Feb. 2022
36535:   Tutorial 93 Geocoding March 2022
36536:   Tutorial 94 Post API Services April 2022
36537:   Tutorial 95 Language Services April 2022
36538:   Tutorial 96 CNN Pipeline June 2022
36539:   Tutor 97 Operating System Services OSS
36540:   Tutor 98 Python Integration
36541:   Tutorial 99 DataScience API October 2022
36542:   Tutorial 100 Data Science Story November 2022
36543:   Tutorial 101 Data Science Story2 Dezember 2022
36544:
36545: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
36546: using Docu for this file is maxbox_functions_all.pdf
36547: PEP - Pascal Education Program  Low Lib Lab ShellHell in UDEMY
36548:
36549: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
36550: http://stackoverflow.com/tags/pascalscript/hot
36551: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
36552: http://sourceforge.net/projects/maXbox   #locs:62000
36553: http://sourceforge.net/apps/mediawiki/maXbox
36554: http://www.blaisepascal.eu/
36555: https://github.com/maxkleiner/maXbox3.git
36556: https://github.com/maxkleiner/maXbox4.git
36557: http://www.heise.de/download/maxbox-1176464.html
36558: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
36559: https://www.facebook.com/pages/Programming-maXbox/166844836691703
36560: http://www.softwareschule.ch/arduino_training.pdf
36561: http://www.delphiarea.com
36562: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
36563: http://entwickler-konferenz.de/2014/speakers/max-kleiner
36564: http://www.heise.de/download/maxbox-1176464.html
36565: https://www.udemy.com/learn-coding-from-the-scratch
36566: http://www.slideshare.net/maxkleiner1/codesign-2015
36567: https://www.dropbox.com/s/yolconwmg4oqta4/Blaise_2and3_SP_Total.pdf?dl=0
36568: http://www.softwareschule.ch/download/maxbox_promo.png
36569: http://max.kleiner.com/maxbox_functions_all.htm
36570: http://max.kleiner.com/boxart.htm
36571: http://www.jurgott.org/linkage/util.htm
36572: http://www.swissdelphicenter.ch/en/niklauswirth.php
36573: http://www.softwareschule.ch/images/maxbox_20years_delphi.jpg
36574: https://softwareschule.code.blog/
36575: https://repl.it/@MaxKleiner/machinelearning4#main.py
36576: https://github.com/project-jedi/jvcl/tree/master/jvcl/examples/JvChartDemo
36577: https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
36578:
36579: {$ All maXbox 4 Examples List Exampleslist  }
36580: https://github.com/maxkleiner/maXbox3/releases
36581: https://github.com/maxkleiner/maXbox4/releases
36582: http://www.softwareschule.ch/download/exampleedition2016.zip
36583:
36584: ***********************************************************************************************
36585: 000_pas_baseconvert.txt                          282_fadengraphik.txt
36586: 000_pas_baseconvert.txt_encrypt                  283_SQL_API_messagetimeout.txt
36587: 000_pas_baseconvert.txt_decrypt                  284_SysTools4.txt
36588: 001_1_pas_functest - Kopie.txt                   285_MineForm_GR32.TXT
36589: 001_1_pas_functest.txt                           285_MineForm_GR32main.TXT
36590: 001_1_pas_functest2.txt                          285_MineForm_GR32mainsolution.TXT
36591: 001_1_pas_functest_clx2.txt                      285_MineForm_propas.TXT
36592: 001_1_pas_functest_clx2_2.txt                    285_MineForm_propas2.TXT
36593: 001_1_pas_functest_openarray.txt                 285_minesweeper2.TXT
36594: 001_pas_lottogen.txt                             285_Patterns_process.txt
```

36595: 001_pas_lottogen_template.txt
36596: 001_pas_lottogen.txtcopy
36597: 002_pas_russianroulette.txt
36598: 002_pas_russianroulette.txtcopy
36599: 002_pas_russianroulette.txtcopy_decrypt
36600: 002_pas_russianroulette.txtcopy_encrypt
36601: 003_pas_motion.txt
36602: 003_pas_motion.txtcopy
36603: 004_pas_search.txt
36604: 004_pas_search_replace.txt
36605: 004_search_replace_allfunctionlist.txt
36606: 005_pas_oodesign.txt
36607: 005_pas_shelllink.txt
36608: 006_pas_oobatch.txt
36609: 007_pas_streamcopy.txt
36610: 008_EINMALEINS_FUNC.TXT
36611: 008_explanation.txt
36612: 008_pas_verwechselt.txt
36613: 008_pas_verwechselt_ibz_bern_func.txt
36614: 008_stack_ibz.TXT
36615: 009_pas_umlrunner.txt
36616: 009_pas_umlrunner_all.txt
36617: 009_pas_umlrunner_componenttest.txt
36618: 009_pas_umlrunner_solution.txt
36619: 009_pas_umlrunner_solution_2step.txt
36620: 010_pas_oodesign_solution.txt
36621: 011_pas_puzzlepas_defect.txt
36622: 012_pas_umlrunner_solution.txt
36623: 012_pas_umlrunner_solution2.txt
36624: 013_pas_linenumber.txt
36625: 014_pas_primetest.txt
36626: 014_pas_primetest_first.txt
36627: 014_pas_primetest_sync.txt
36628: 015_pas_designbycontract.txt
36629: 015_pas_designbycontract_solution.txt
36630: 016_pas_searchrec.txt
36631: 017_chartgen.txt
36632: 018_data_simulator.txt
36633: 019_dez_to_bin.txt
36634: 019_dez_to_bin_grenzwert_ibz.txt
36635: 020_proc_feedback.txt
36636: 021_pas_symkey.txt
36637: 021_pas_symkey_solution.txt
36638: 022_pas_filestreams.txt
36639: 023_pas_find_searchrec.txt
36640: 023_pas_pathfind.txt
36641: 024_pas_TFileStream_records.txt
36642: 025_prime_direct.txt
36643: 026_pas_memorystream.txt
36644: 027_pas_shellexecute_beta.txt
36645: 027_pas_shellexecute_solution.txt
36646: 028_pas_dataset.txt
36647: 029_pas_assignfile.txt
36648: 029_pas_assignfile_dragndropexe.txt
36649: 030_palindrome_2.txt
36650: 030_palindrome_tester.txt
36651: 030_pas_recursion.txt
36652: 030_pas_recursion2.txt
36653: 031_pas_hashcode.txt
36654: 032_pas_crc_const.txt
36655: 033_pas_cipher.txt
36656: 033_pas_cipher_def.txt
36657: 033_pas_cipher_file_2_solution.txt
36658: 034_pas_soundbox.txt
36659: 035_pas_crcscript.txt
36660: 035_pas_CRCscript_modbus.txt
36661: 036_pas_includetest.txt
36662: 036_pas_includetest_basta.txt
36663: 037_pas_define_demo32.txt
36664: 038_pas_box_demonstrator.txt
36665: 039_pas_dllcall.txt
36666: 040_paspointer.txt
36667: 040_paspointer_old.txt
36668: 041_pasplotter.txt
36669: 041_pasplotter_plus.txt
36670: 042_pas_kgv_ggt.txt
36671: 043_pas_proceduretype.txt
36672: 044_pas_14queens_solwith14.txt
36673: 044_pas_8queens.txt
36674: 044_pas_8queens_sol2.txt
36675: 044_pas_8queens_solutions.txt
36676: 044_queens_performer.txt
36677: 044_queens_performer2.txt
36678: 044_queens_performer2tester.txt
36679: 045_pas_listhandling.txt
36680: 046_pas_records.txt
36681: 047_pas_modula10.txt
36682: 048_pas_romans.txt
36683: 049_pas_ifdemo.txt

286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
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

36684: 049_pas_ifdemo_BROKER.txt
36685: 050_pas_primetest2.txt
36686: 050_pas_primetester_thieves.txt
36687: 050_program_starter.txt
36688: 050_program_starter_performance.txt
36689: 051_pas_findtext_solution.txt
36690: 052_pas_text_as_stream.txt
36691: 052_pas_text_as_stream_include.txt
36692: 053_pas_singleton.txt
36693: 054_pas_speakpassword.txt
36694: 054_pas_speakpassword2.txt
36695: 054_pas_speakpassword_searchtest.txt
36696: 055_pas_factorylist.txt
36697: 056_pas_demeter.txt
36698: 057_pas_dirfinder.txt
36699: 058_pas_filefinder.txt
36700: 058_pas_filefinder_pdf.txt
36701: 058_pas_filefinder_screview.txt
36702: 058_pas_filefinder_screview2.txt
36703: 058_pas_filefinder_screview3.txt
36704: 059_pas_timertest.txt
36705: 059_pas_timertest_2.txt
36706: 059_pas_timertest_time_solution.txt
36707: 059_timerobject_starter2.txt
36708: 059_timerobject_starter2_ibz2_async.txt
36709: 059_timerobject_starter2_uml.txt
36710: 059_timerobject_starter2_uml_main.txt
36711: 059_timerobject_starter4_ibz.txt
36712: 060_pas_datefind.txt
36713: 060_pas_datefind_exceptions2.txt
36714: 060_pas_datefind_exceptions_CHECKTEST.txt
36715: 060_pas_datefind_fulltext.txt
36716: 060_pas_datefind_plus.txt
36717: 060_pas_datefind_plus_mydate.txt
36718: 061_pas_randomwalk.txt
36719: 061_pas_randomwalk_plus.txt
36720: 062_paskorrelation.txt
36721: 063_pas_calculateform.txt
36722: 063_pas_calculateform_2list.txt
36723: 064_pas_timetest.txt
36724: 065_pas_bitcounter.txt
36725: 066_pas_eliza.txt
36726: 066_pas_eliza_include_sol.txt
36727: 067_pas_morse.txt
36728: 068_pas_piezo_sound.txt
36729: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
36730: 069_my_LEDBOX.TXT
36731: 069_pas_ledmatrix.txt
36732: 069_pas_LEDMATRIX_Alphabet.txt
36733: 069_pas_LEDMATRIX_Alphabet_run.txt
36734: 069_pas_LEDMATRIX_Alphabet_tester.txt
36735: 069_PAS_LEDMATRIX_COLOR.TXT
36736: 069_pas_ledmatrix_fixedit.txt
36737: 069_pas_LEDMATRIX_soundbox.txt
36738: 069_pas_LEDMATRIX_soundbox2.txt
36739: 069_Richter_MATRIX.TXT
36740: 070_pas_functionplot.txt
36741: 070_pas_functionplotter2.txt
36742: 070_pas_functionplotter2_mx4.txt
36743: 070_pas_functionplotter2_tester.txt
36744: 070_pas_functionplotter3.txt
36745: 070_pas_functionplotter4.txt
36746: 070_pas_functionplotter_digital.txt
36747: 070_pas_functionplotter_elliptic.txt
36748: 070_pas_function_helmholtz.txt
36749: 070_pas_textcheck_experimental.txt
36750: 071_pas_graphics.txt
36751: 071_pas_graphics_drawsym.txt
36752: 071_pas_graphics_drawsym_save.txt
36753: 071_pas_graphics_random.txt
36754: 072_pas_fractals.txt
36755: 072_pas_fractals_2.txt
36756: 072_pas_fractals_blackhole.txt
36757: 072_pas_fractals_perfomance.txt
36758: 072_pas_fractals_perfomance_new.txt
36759: 072_pas_fractals_perfomance_sharp.txt
36760: 072_pas_fractals_performance.txt
36761: 072_pas_fractals_performance_mX4.txt
36762: 073_pas_forms.txt
36763: 074_pas_chartgenerator.txt
36764: 074_pas_chartgenerator_solution.txt
36765: 074_pas_chartgenerator_solution_back.txt
36766: 074_pas_charts.txt
36767: 075_bitmap_Artwork2.txt
36768: 075_pas_bitmappuzzle.txt
36769: 075_pas_bitmappuzzle24.prod.txt
36770: 075_pas_bitmappuzzle2_prod.txt
36771: 075_pas_bitmappuzzle3.txt
36772: 075_pas_bitmapsolve.txt

318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docutype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_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

36773: 075_pas_bitmap_Artwork.txt
36774: 075_pas_puzzlepas_solution.txt
36775: 076_pas_3dcube.txt
36776: 076_pas_circle.txt
36777: 077_pas_mmshow.txt
36778: 078_pas_pi.txt
36779: 079_pas_3dcube_animation.txt
36780: 079_pas_3dcube_animation4.txt
36781: 079_pas_3dcube_plus.txt
36782: 080_pas_hanoi.txt
36783: 080_pas_hanoi2.txt
36784: 080_pas_hanoi2_file.txt
36785: 080_pas_hanoi2_sol.txt
36786: 080_pas_hanoi2_tester.txt
36787: 080_pas_hanoi2_tester_fast.txt
36788: 080_pas_hanoi3.txt
36789: 081_pas_chartist2.txt
36790: 082_pas_biorythmus.txt
36791: 082_pas_biorythmus_solution.txt
36792: 082_pas_biorythmus_solution_3.txt
36793: 082_pas_biorythmus_test.txt
36794: 083_pas_GITARRE.txt
36795: 083_pas_soundbox_tones.txt
36796: 084_pas_waves.txt
36797: 085_mxsinus_logo.txt
36798: 085_sinus_plot_waves.txt
36799: 086_pas_graph_arrow_heart.txt
36800: 087_bitmap_loader.txt
36801: 087_pas_bitmap_solution.txt
36802: 087_pas_bitmap_solution2.txt
36803: 087_pas_bitmap_subimage.txt
36804: 087_pas_bitmap_test.txt
36805: 088_pas_soundbox2_mp3.txt
36806: 088_pas_soundbox_mp3.txt
36807: 088_pas_sphere_2.txt
36808: 089_pas_gradient.txt
36809: 089_pas_maxland2.txt
36810: 090_pas_sudoku4.txt
36811: 090_pas_sudoku4_2.txt
36812: 091_pas_cube4.txt
36813: 092_pas_statistics4.txt
36814: 093_variance.txt
36815: 093_variance_debug.txt
36816: 094_pas_daysold.txt
36817: 094_pas_stat_date.txt
36818: 095_pas_ki_simulation.txt
36819: 096_pas_geisen_problem.txt
36820: 096_pas_montyhall_problem.txt
36821: 097_lotto_proofofconcept.txt
36822: 097_pas_lottocombinations_beat_plus.txt
36823: 097_pas_lottocombinations_beat_plus2.txt
36824: 097_pas_lottocombinations_universal.txt
36825: 097_pas_lottosimulation.txt
36826: 098_pas_chartgenerator_plus.txt
36827: 099_pas_3D_show.txt
36828: 200_big_numbers.txt
36829: 200_big_numbers2.txt
36830: 201_streamload_xml.txt
36831: 202_systemcheck.txt
36832: 203_webservice_simple_intftester.txt
36833: 204_webservice_simple.txt
36834: 205_future_value_service.txt
36835: 206_DTD_string_functions.txt
36836: 207_ibz2_async_process.txt
36837: 208_crc32_hash.txt
36838: 209_cryptohash.txt
36839: 210_public_private.txt
36840: 210_public_private_cryptosystem5_ibz_spektrum.txt
36841: 210_public_private_cryptosystem.txt
36842: 211_wipe_pattern.txt
36843: 211_wipe_pattern2.txt
36844: 211_wipe_pattern_solution.txt
36845: 212_pas_statisticmodule4.TXT
36846: 212_pas_statisticmoduletxt.TXT
36847: 212_statisticmodule4.txt
36848: 213_pas_BBP_Algo.txt
36849: 214_mxdocudemo.txt
36850: 214_mxdocudemo2.txt
36851: 214_mxdocudemo3.txt
36852: 215_hints_test.TXT
36853: 216_warnings_test.TXT
36854: 217_pas_heartbeat.txt
36855: 218_biorythmus_studio.txt
36856: 219_cipherbox.txt
36857: 219_crypt_source_comtest_solution.TXT
36858: 220_cipherbox_form.txt
36859: 220_cipherbox_form2.txt
36860: 221_bcd_explain.txt
36861: 222_memoform.txt

366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maXbook_v4tester.txt
372stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefunctesterfilemon.txt
380_coolfunc.txt
380_coolfunc2.txt
380_coolfunc_tester.txt
381_bitcoin_simulation.txt
382_GRMath.TXT
382_GRMath_PI_Proof.TXT
382_GRMath_Riemann.TXT
383_MDAC_DCOM.txt
384_TeamViewerID.TXT
386_InternetRadio.TXT
387_fulltextfinder.txt
387_fulltextfinder_cleancode.txt
387_fulltextfinder_fast.txt
387_fulltext_getscripttest.txt
388_TCPServerSock.TXT
388_TCPServerSock2.TXT
388_TCPServerSockClient.TXT
389_TAR_Archive.TXT
389_TAR_Archive_test.TXT
389_TAR_Archive_test2.TXT
390_Callback3.TXT
390_Callback3Rec.TXT
390_CallbackClean.TXT
390_StringlistHTML.TXT
391_ToDo_List.TXT
392_Barcode.TXT
392_Barcode2.TXT
392_Barcode23.TXT
392_Barcode2scholz.TXT
392_Barcode3scholz.TXT
393_QRCode.TXT
393_QRCode2.TXT
393_QRCode2Direct.TXT
393_QRCode2DirectIndy.TXT
393_QRCode2Direct_detlef.TXT
393_QRCode3.TXT
394_networkgraph.TXT
394_networkgraph_depwalkutilstest.TXT
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

36862: 223_directorybox.txt
36863: 224_dialogs.txt
36864: 225_sprite_animation.txt
36865: 226_ASCII_Grid2.TXT
36866: 227_animation.txt
36867: 227_animation2.txt
36868: 228_android_calendar.txt
36869: 229_android_game.txt
36870: 229_android_game_tester.txt
36871: 230_DataProvider.txt
36872: 230_DataSetProvider.txt
36873: 230_DataSetXMLBackupScholz.txt
36874: 231_DBGrid_access.txt
36875: 231_DBGrid_XMLaccess.txt
36876: 231_DBGrid_XMLaccess2.txt
36877: 231_DBGrid_XMLaccess_locatetester.txt
36878: 231_DBGrid_XML_CDS_local.txt
36879: 232_outline.txt
36880: 232_outline_2.txt
36881: 233_modular_form.txt
36882: 234_debugoutform.txt
36883: 235_fastform.TXT
36884: 236_componentpower.txt
36885: 236_componentpower_back.txt
36886: 237_pas_4forms.txt
36887: 238_lottogen_form.txt
36888: 239_pas_sierpinski.txt
36889: 239_pas_sierpinski2.txt
36890: 240_unitGlobal_tester.txt
36891: 241_db3_sql_tutorial.txt
36892: 241_db3_sql_tutorial2.txt
36893: 241_db3_sql_tutorial2fix.txt
36894: 241_db3_sql_tutorial3.txt
36895: 241_db3_sql_tutorial3connect.txt
36896: 241_db3_sql_tutorial3_fptest.txt
36897: 241_RTL_SET2.txt
36898: 241_RTL_SET2_tester.txt
36899: 242_Component_Control.txt
36900: 243_tutorial_loader.txt
36901: 244_script_loader_loop.txt
36902: 245_formapp2.txt
36903: 245_formapp2_tester.txt
36904: 245_formapp2_testerX.txt
36905: 246_httpapp.txt
36906: 247_datecalendar.txt
36907: 248_ASCII_Grid2_sorted.TXT
36908: 249_picture_grid.TXT
36909: 250_tipsandtricks2.txt
36910: 250_tipsandtricks3.txt
36911: 250_tipsandtricks3api.txt
36912: 250_tipsandtricks3_admin_elevation.txt
36913: 250_tipsandtricks3_tester.txt
36914: 250_tipsandtricks4_tester.txt
36915: 250_tipsandtricks4_tester2.txt
36916: 251_compare_noise_gauss.txt
36917: 251_whitenoise.txt
36918: 251_whitenoise2.txt
36919: 252_hilbert_turtle.txt
36920: 252_pas_hilbert.txt
36921: 253_opearatingsystem3.txt
36922: 254_dynarrays.txt
36923: 255_einstein.txt
36924: 256_findconsts_of_EXE.txt
36925: 256_findfunctions2_of_EXE.txt
36926: 256_findfunctions2_of_EXEaverp.txt
36927: 256_findfunctions2_of_EXEspec.txt
36928: 256_findfunctions3.txt
36929: 256_findfunctions_of_EXE.txt
36930: 257_AES_Cipher.txt
36931: 258_AES_cryptobox.txt
36932: 258_AES_cryptobox2.txt
36933: 258_AES_cryptobox2_passdlg.txt
36934: 259_AES_crypt_directory.txt
36935: 260_sendmessage_2.TXT
36936: 260_sendmessage_beta.TXT
36937: 261_probability.txt
36938: 262_mxoutputdemo4.txt
36939: 263_async_sound.txt
36940: 264_vclutils.txt
36941: 264_VCL_utils2.txt
36942: 265_timer_API.txt
36943: 266_serial_interface.txt
36944: 266_serial_interface2.txt
36945: 266_serial_interface3.txt
36946: 267_ackermann_rec.txt
36947: 267_ackermann_variants.txt
36948: 268_DBGrid_tree.txt
36949: 269_record_grid.TXT
36950: 270_Jedi_FunctionPower.txt

407_indydemo.txt
408_testroboter.txt
409_excel_control.txt
410_keyboardevent.txt
411_json_test.txt
412_Zeosutils.txt
413_listview2.txt
414_avrdude_flash.txt
415_avrdude_writehex.txt
416_sonar_startscriptEKON.TXT
416_sonar_startscriptEKON_reporting.TXT
417_GRMath_PI_Proof2.TXT
418_functional_paradigm.txt
419_archimedes_spiral.txt
419_archimedes_spiral2.txt
420_archimedes_arduino.txt
420_Lissajous.txt
421_PI_Power.TXT
421_PI_Power2.TXT
422_world_bitboxx.txt
423_game_of_life.TXT
423_game_of_life2.TXT
423_game_of_life3.TXT
423_game_of_life3_test.TXT
423_game_of_life4.TXT
423_game_of_life4_kryptum.TXT
424_opengl_tester.txt
425_reversi_game.txt
426_IBUtils.TXT
427_IBDatabase.TXT
428_SortGrid.TXT
429_fileclass.txt
430_fileoperation.txt
430_fileoperation_tester.txt
431_performance_index.txt
432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
437_WinAPItop_Firebirdtester.txt
438_OvcInternational.txt
439_AsyncFreeDemo.txt
439_AsyncFreeDemoForm.txt
440_DLL_Tutor.txt
440_DLL_Tutor2.txt
440_XML_Tutor.txt
440_XML_Tutor2.txt
441_make_app.txt
442_arduino_rgb_led.txt
443_webserver_arduino_rgb_light.txt
443_webserver_arduino_rgb_light4.txt
444_webserver_arduino3ibz_rgb_led_basta.txt
445_datagrid.txt
445_datagrid2.txt
445_datagrid_android_arduino.txt
446_arduino_timer.txt
447_patternFrm_mx3.txt
448_Synapse.txt
448_Synapse2.txt
449_dweb_start_tester.txt
450_Synapse_HTTPS.txt
450_Synapse_Mime.txt
450_Synapse_ScanPing.txt
451_ocx_player.txt
451_OCX_WinPlayer2.txt
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

36951: 270_Jedi_FunctionPowertester.txt
36952: 271_closures_study.txt
36953: 271_closures_study_workingset2.txt
36954: 272_pas_function_show.txt
36955: 273_pas_function_show2.txt
36956: 274_library_functions.txt
36957: 275_turtle_language.txt
36958: 275_turtle_languagehouse2_time2.txt
36959: 275_turtle_language_save.txt
36960: 276_save_algo.txt
36961: 276_save_algo2.txt
36962: 277_functionsfor39.txt
36963: 278_DB_Dialogs.TXT
36964: 279_hexer2.TXT
36965: 279_hexer2macro.TXT
36966: 279_hexer2macroback.TXT
36967: 280_UML_process.txt
36968: 280_UML_process_knabe2.txt
36969: 280_UML_process_knabe3.txt
36970: 280_UML_process_TIM_Botzenhardt.txt
36971: 280_UML_TIM_Seitz.txt
36972: 281_picturepuzzle.txt
36973: 281_picturepuzzle2.txt
36974: 281_picturepuzzle3.txt
36975: 281_picturepuzzle4.txt
36976: 479_inputquery.txt
36977: 480_regex_pathfinder2.txt
36978: 482 processPipe.txt
36979: 483_PathFuncTest_mX.txt
36980: 485_InnoFunc.txt
36981: 487_asyncKeyState.txt
36982: 489_simpleComport.txt
36983: 491_analogmeter.txt
36984: 493_gadgets.txt
36985: 496_InstallX.txt
36986: 498_UnitTesting.txt
36987: 500_diceoflifes.txt
36988: 502_findalldocs.txt
36989: 504_fileclass.txt
36990: 506_colormatrix.txt
36991: 508_simplecomportmorse.txt
36992: 509_509_GEOMap2_SReverse.TXT
36993: 511_LEDLabel.txt
36994: 513_StreamIntegration.txt
36995: 515_ledclock3.txt
36996: 517_animation7.txt
36997: 519_powtils.txt
36998: 521_iputils2.txt
36999: 523_NMEA.txt
37000: 525_GEO84s.txt
37001: 527_GPSDemo.txt
37002: 529_profilertest.txt
37003: 531_profilertest.txt
37004: 533_syncasync_demo.txt
37005: 535_Battleship3.pas
37006: 537_iniplus.TXT
37007: 539_timeturtle123.txt
37008: 541_webserver_arduino_motorturtle.txt
37009: 543_MATH_TurboP.PAS
37010: 545_strips.TXT
37011: 547_regexmaster.TXT
37012: 549_3D_Panorama.txt
37013: 551_ArduinoTester.txt
37014: 553_ArduinoCockBit3.txt
37015: 556_stringlistrandom.TXT
37016: 558_highrestimer.TXT -559_higthrestimer2
37017: 561_newfunctions399160.txt
37018: 563_moonpaper.txt
37019: 565_ConsoleCapture.txt
37020: 567_SquareWordGrids2.txt
37021: 569_keylog.txt - 569_ServiceMgr2.TXT
37022: 571_myPing.txt
37023: 573_modbusfrm_Main.pas
37024: 575_TARTARUGA_Desktop.txt
37025: 577_listbox2list.txt
37026: 579_numbersystems_sort.txt
37027: 581_stringstream.txt
37028: 583_VirtualConstructor_savereport.txt
37029: 585_fulltextfinder_cleancode_override.txt
37030: 587_one_function.txt
37031: 589_avi_animate.txt
37032: 591_emailattach.txt
37033: 593_round_time.txt
37034: 595_check_memory.txt
37035: 597_ole_commands.txt
37036: 599_bug.txt
37037: 601_PECheckSum.txt
37038: 603_cupids_arrow.TXT
37039: 605_maxonmotor3DTage2BASTA2015.TXT

464_function_procedure3.txt
465_U_HexView.txt
466_moon.txt
466_moon_inputquery.txt
4671_cardmagic.txt
467_helmholtz_graphic.txt
468_URLMon.txt

468_URLMon2.txt
469_formarrow.txt
469_formarrow_datepicker.txt
469_formarrow_datepicker_ibz_result.txt
469_ibzresult.txt
470_DFFUtils_compiled.txt
470_DFFUtils_ScrollingLED.txt
470_Oscilloscope.txt
470_Oscilloscope_code.txt
471_cardmagic.txt
471_cardmagic2.txt
472_allcards.TXT
473_comboset.txt
474_wakeonlan.txt
474_wakeonlan2.txt
476_getscripttest.txt
477_filenameonly.txt
480_regex_pathfinder.txt
481_processList.txt
482 processPipeGCC.txt
484_filefinder3.txt
486_VideoGrabber.txt
488_asyncTerminal.txt
490_webCamproc.txt
492_snowflake2.txt
495_fourierfreq.txt
497_LED.txt
499_mulu42.txt
501_firebird_datasnap_tests.txt
503_led_switch.txt
505_debug.txt
507_derutils.txt
509_GEOMap2.txt
510_510_bonn_gpsdata_mX4.pas
512_LED_moon.txt
514_LED_moon2.txt
516_mapview.txt
518_sensors_meter.txt
520_run_bytecode.txt
522_getgeocode.txt
524_NAV_Utils.txt
526_Compass_meter.txt
528_linescount.txt
530_3DLab.txt
532_mcicommand.txt
534_arduino_cockpit.TXT
536_ressource_grid2.txt
538_shellbatch.txt
540_NeuralNetwork.pas
542_arduino_sound.txt
544_UTIL01.PAS
546_fourier3.pas
548_STExpressions.TXT -Services
550_Expressions.TXT - 550_ADO_OLEDB.txt
552_WaitExec32.txt
554_Watdchdog.txt - 555_CODEsign2.txt
557_4dice2015.txt
560_PSUtils.TXT
562_shellctrldemo.txt
564_queryperformance.txt
566_queryperformance2.txt
568_U_BigFloatTestscript2.pas
570_turingspeech.txt
572_shellctrlplus.txt
574_arduino_cockpit5.TXT
576_outlineEX11.PAS
578_access_db_logsimulation3.txt
580_indystacksearch_geo.txt
582_indystackwin.txt
584_ProcessList2fontwidth.txt last
586_STRandom.txt
588_XSBuiltins.txt
590_HTML_to_RTF.txt
592_getTypeLibList.txt
594_check_creditcard.txt
596_time_delays.txt
598_software_list.txt
600_surprise_nice.txt
602_multilang_game.txt (moonbug)
604_GEOCodeReverse4.TXT
606_U_FibonacciSunflower.TXT

```
37040: 607_DataSetProvider_CDS_ADO.txt                608_ColorMixer_Arduino2.txt
37041: 609_ScriptExecutor (beta)                      610_3D_DLL.txt
37042: 611_Arduino_COMOutputs.txt                     612_MTerminal.pas
37043: 611_Arduino_COMOutputs_TMP36_3.txt
37044: 613_uPSI_DIUtils_test.pas                      614_inbrowserock.txt
37045: 615_SONAR_51_Starter.txt                       616_ComTerminalDlg.pas
37046: (589_AVI_Throbber.res.txt)                     589_avi_animate.txt
37047: 616_ComTerminalDlg.pas                         //end for 3.9.9.195
37048:
37049: 617_API_coding_tut37.txt                       617_API_coding_tut39.txt
37050: 618_bayes_filter.txt                           618_bayes_filter_dic.txt
37051: 618_bayes_filter_dic2.txt                      619_crypto_package_demo.txt
37052: 619_crypto_package_demo_policy.txt             619_crypto_packr.txt
37053: 620_serialtimer.txt                            621_bruteforceattack_dic.txt
37054: 622_netsh_master_restore4.psb                  622_netsh_master_sonar.pas
37055: 623_exspawnu.pas                               623_exspawnu_buffoverflow.pas
37056: 624_Misctest_lexscanner.pas                    624_Misctest_lexscanner2.pas
37057: 625_binominal_bigint3_testcase.txt             626_lorentztransformation.TXT
37058: 627_4gewinnt_main3.txt                         628_FormatFloat.pas
37059: 629_Word_FORMULA.PAS                           630_multikernel2.TXT
37060: 630_multikernel3.TXT                           631_dmath_testfunc_mx3.pas
37061: 632_safecall12.txt                             632_safecheck2.txt
37062: 632_safecheck2_regex_basta.txt                 633_gamma_func.txt
37063: 634_briefcase_XMLDB.xml                        634_BriefcaseMain2.pas
37064: 634_maxbase3server.mdb                         635_bitcount_bytecodestudy.txt
37065: 636_rest_apiexamples.txt                       637_psnconvert3.TXT
37066: 637_psnconvert4.TXT                            637_startscript_psnconvert2.TXT
37067: 638_Mathslib_prime.pas.txt                     639_prime_factors.txt
37068: 640_rest_geocode.txt                           640_rest_weather_report.txt
37069: 640_weather_cockpit5.TXT                       640_weather_cockpit6.TXT
37070: 641_voicecommand.TXT                           642_BDE_export.PAS
37071: 643_EKON19_Tester3_OLE_Reg.TXT                 644_Synapse_Mimetest_regEx3.txt
37072: 645_age_guess_REX.pas                          646_pi_evil.TXT
37073: 646_pi_evil2.TXT                               647_StringtoHTMLBasta.TXT
37074: 648_SYNEdit_ExportDemo.TXT                     649_dayio.pas
37075: 649_StringlistHTMLtestBasta.TXT                650_sched.pas
37076: 650_http_server2tester.txt                     650_time_routines.txt
37077: 650_dbblober.txt                               650_drawdice2016.txt
37078: 651_StrUtilmaxbase.pas                         652_graph3DMainUnit2.pas
37079: 653_PdoxTstbox.pas                             654_spectrum_dice2016_3.txt
37080: 655_arduino_chess.txt                          655_arduino_chess2.txt
37081: 656_XML_RpcCommon.pas                          656_XML_RpcCommon2.pas
37082: 657_PEM_Cert.txt                               658_ASN1_Cert.txt
37083: 659_VirtListBox.txt                            660_X509_Cert.pas
37084: 660_InetUtils_Rest.txt                         661_REXX.pas
37085: 662__VisualChronForm.pas                       663_Tokens.pas
37086: 664_DataAnnotationsValid.pas.pas               665_keditcommon.pas.pas
37087: 666_SysInfoCtrls.pas.pas                       667_URobo1.pas.pas
37088: 667_URobo2.pas.pas                             668_U_SoundGen3.pas.pas
37089: 669_uptime.txt.pas                             700_interestingX.txt
37090: 670_interesting_birthday_paradox.txt           671_replacedigit.txt //File
37091: 672_regex_ask_task.txt                         673_pipe_singapure.txt //LAB
37092: 674_pipe_graphics.txt                          675_bitcoin_doublehash.txt
37093: 676_run_as_admin.txt                           677_pingtest4.txt
37094: 678_fmTraceRouteMainU.pas                      679_messagefMain.pas
37095: 680_gravity_waves.txt                          681_kronos_3DLab.TXT
37096: 682_ScreenMatrixToBitmap.pas                   683_mx422_functions_demo.txt
37097: 684_mathe4.pas                                 685_leanfitmath.txt
37098: 686_namedpipe_demo.txt                         686_namedpipe_demo2.txt
37099: 687_memorymapped_filestream.txt                688_KGraphics_Call.txt
37100: 689_proctype_alias.txt                         690_calc_pi_ex.pas
37101: 691_httpservermain3.txt                        692_fservermain3.pas
37102: 692_imageserv_fClient33.pas                    693_OverbyteIcsUtils_tester.pas
37103: 694_OverbyteIcsAsn1Utils.pas                   695_IdAntiFreeze_tester.pas
37104: 696_kmemo_demo_Main.pas                        696_kmemo_demo.pas
37105: 697_OverbyteIcsTicks64.pas                     697_OverbyteIcsTicks64Demo1.pas
37106: 698_new_classes_tester.pas                     693_OverbyteIcsDnsQuerytester.pas
37107: 698_Bitcoin_blockchaintester.pas               699_OverbyteIcsSha1_tester.pas
37108: 700_mX422_functions_demo2.txt                  700_mX422_functions_demo3.txt
37109: 700_new_function_snippets.pas                  700_function_snippets3_metrics.inc
37110: 701_all_fibonacci.txt                          702_pascal_fibonacci.txt
37111: 702_X509_Cert_OpenSSL_1.txt                    702_X509_Cert_OpenSSL_2.txt
37112: 703_pc_sensors.txt                             704_entropy_test.txt
37113: 705_magsubs1_functions.pas                     705_parser_demo_case.txt
37114: 706_population_count.txt                       707_GrayCode.txt
37115: 708_XML_DOM_ReaderTutor.txt                    709_BarcodeReader.txt
37116: 710_enigma_riddler.txt                         710_EnumRegKeys.txt
37117: 710_geo_distance_form_mapbox.txt               710_geo_distance_dialog.txt
37118: 710_RegSvrUtils_mX4.pas                        711_geo_satellite_mapbox.txt
37119: 712_rootsfunction.txt                          712_towerofhanoi_animation.pas
37120: 713_SnakeA2.pas                                713_internet_traces_mx_screenshot2.png
37121: 714_astroids_GAME3.PAS                         714_GAME_spaceinvaders.PAS
37122: 715_shape_calculator.PAS                       716_webplan.PAS
37123: 717_AdLog.PAS                                  718_AdPerformanceCounter.PAS
37124: 719_rootsfunction.txt                          720_euler_project.txt
37125: 720_HTML_TableExport_CSS.pas                   720_selfdescribing_number.txt
37126: 721_DOSCapture_netstat2.TXT                    722_refererHeader_monitor.TXT
37127: 723_remote_notepad_message.TXT                 724_spyware_monitor_test.TXT
37128: 725_CRC32_live.txt                             725_Install_Routines.TXT
```

37129: 725_725_Cholesky.pas
37130: 725_blockchain_sensors3.txt
37131: 726_DBGrid_XML_CDataSet.txt
37132: 728_mci_macros.TXT
37133: 729_PrivateKey_mX.pas
37134: 730_modular_exponentiation_IBZ.txt
37135: 730_ternary_logice.txt
37136: 731_DWS_Client4_EKON20.pas
37137: 732_hailstonesequence_delphi.txt
37138: 733_jensensdevice.txt
37139: 735_Linear_congruential_generator.txt
37140: 735_Euler_p117_openSSL_scripts.txt
37141: 737_xmastree2.txt
37142: 738_mX4_exact_fishertest2.pas
37143: 739_GIFPAL.PAS
37144: 740_Globalreplacetext.txt
37145: 740_msgpump1.txt
37146: 740_uFTPServermX2_listtest.pas
37147: 740_RosettaIsaac_Cryptocode_Delphi
37148: 741_DrawYinAndYang.txt
37149: 742_VectorProducts.txt
37150: 744_Web_scraping_UTCTime.txt
37151: 745_crypto_memory.txt
37152: 746_uText.pas
37153: 747_uSteganos.pas
37154: 749_helloWebServer.txt
37155: 749_helloWebServer3.txt
37156: 750_EchoServer3.txt
37157: 210_RSA_crypto_complete8hybrid.txt
37158: 750_ibz_cryptomem_RSA_proof_64.txt
37159: 750_SimpleTCPMain.pas
37160: 751_Elevator_Simulator.pas
37161: 752_BitmapHistogram.txt
37162: 752_U_AstroDemo_mX4_2.pas
37163: 753_xbase_tester.pas
37164: 755_science_ibz_spektrum.txt
37165: 755_sedol2.txt
37166: 756_shellscripthell_javascript.txt
37167: 756_shellscripthell_javascript2.txt
37168: 667_URobo2EKON_FUN_Tracking2.pas
37169: 757_tower_of_hanoiX.txt
37170: 758_speechmachine.txt
37171: 760_Normal_distribution.txt
37172: 760_XML_DOM_OLE.TXT
37173: 760_systeminfo_tester.txt
37174: 766_wmi_GetServiceStatus.txt
37175: 760_systeminfo_tester2.txt
37176: 761_BoulesBillard3.txt
37177: 760_systeminfo_tester2.txt
37178: 761_BoulesBillard3.txt
37179: 762_regex_analytica.txt
37180: 762_matlab.txt
37181: 764_reverse.txt
37182: 765_codesign.txt
37183: 766_wmi_management.txt
37184: 767_mint_geo_map.txt
37185: 769_tzutil.pas
37186: 770_pascal_coin_api_ascii_talk.txt
37187: 770_U_ParsingDemo3_ASCIITalk.pas
37188: 770_3hires_benchmarktimer3.txt
37189: 771_fibonacci_first.txt
37190: 773_streamfibonacci_first.txt
37191: 774_findwindow_calc.txt
37192: 776_fiboseq_state_android_pi.txt
37193: 777_netportscan.txt
37194: 777_tictactoe2.txt
37195: 777_tictac4picboard_16F887_arduino.txt
37196: 778_advapi32_dll_SHA256.txt
37197: 779_drivemanager.txt
37198: 780_mandelbrotset_max.txt
37199: 780_set_enumeration.txt
37200: 780_newfunctions_calctest_42610.txt
37201: 780_abc_problem.txt
37202: 780_mandelbrotset_max2.txt
37203: 780_thingspeak_mathworks.txt
37204: 780_JSONDemo.txt
37205: 780_ppk_xmltransform.txt
37206: 780_xrtl_net_URIUtils.pas
37207: 781_ProxyUtils.txt
37208: 781_ProxyUtils2performance.txt
37209: 782_XML_DOM_ADO_API_REST_demo.txt
37210: 783_turtle_sign.txt
37211: 785_Matrix_Transpose.txt
37212: 787_cologne_Bitmap21.txt
37213: 789_DOM_XML.txt
37214: 791_uFTPServer.pas
37215: 793_uOneInstance.pas
37216: 795_uOutput.pas
37217: 797_SynEditCodeFolding_routines.pas

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
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

37218: 799_list_duplicates.txt
37219: 800_isodatetime.txt
37220: 802_indystack_geo2_picture.txt
37221: 803_textanalysis2.txt
37222: 805_webspider_Mainconsole.pas
37223: 805_webspider_Mainconsole2_2.pas
37224: 807_FANN_XorSample.pas
37225: 807_FANN_XorSample2.pas
37226: 810_bullcow_mastermind.txt
37227: 810_neuralnetwork2.txt
37228: 810_neuralnetwork21.txt
37229: 811_mXpcatest_dmath_datascience2.pas
37230: 813_PCA_datascience_iris.txt
37231: 814_bigmulu.txt
37232: 813_PCA_datascience_iris3.txt
37233: 815_look_and_say.txt
37234: 816_SnowballProgram.pas
37235: 818_spider_traversialdemo.txt
37236: 819_ES.Utils.pas
37237: 820_U_ProbabilityDist.pas
37238: 820_U_ProbabilityDist2.psb
37239: 820_bostondataset_socks4test.pas
37240: 820_bostondataset_socks4test_irisset.pas
37241: 820_bostondataset_irisset_clean.pas
37242: 821_Narcissistic.txt
37243: 822_perceptron1_2.htm
37244: 822_perceptron1_2.txt
37245: 822_perceptron1_3.htm
37246: 807_FANN_PerceptronSample.pas
37247: 822_perceptron_for_pascal2.txt
37248: 822_perceptron_for_pascal2regression.txt
37249: 824_catamorphism.txt
37250: 825_uTPLb_Hash_tester.pas
37251: 826_sentiment_api2_1.txt
37252: 826_sentiment_api2_2.txt
37253: 820_bytecodetest.txt
37254: 826_reglin_regression_fit2.pas
37255: 827_save_UTF8.txt
37256: 828_install_routine2.txt
37257: 829_save_JSON.txt
37258: 830_RSA_cryptosystem_wiki.txt
37259: 831_regex_light.txt
37260: 833_findfactors.txt
37261: 835_fibonsteps_sequence.txt
37262: 837_penneysgame.txt
37263: 838_endofuniverse_explain.txt
37264: 840_maxboxmandel2.png
37265: 840_URungeKutta4test.pas
37266: 842_U_Keno2.pas
37267: 842_U_JPGClock2.pas
37268: 844_OpenSSL_synapsetester.txt
37269: 845_Gray_Code.txt
37270: 845_TestPermutations2.txt
37271: 846_longmultiplication.pas
37272: 848_xml_intfn.pas
37273: 849_pi3.pas
37274: 810_neuralnetwork21pascal_tutor66.txt
37275: 810_neuralnetwork21pascal_tutor66predict.txt
37276: 854_U_Invertedtext.pas
37277: 855_dll_from_maXbox.pas
37278: 856_snake_maxbox4_4.png
37279: 856_snake_maxbox4_5.png
37280: 856_snake_maxbox4_6.png
37281: 856_SnakeAmX42_tab.pas
37282: 859_maputils_client.pas
37283: 860_virtualbox_overview.png
37284: 826_similarity_api2.txt
37285: 860_similarity_api3.txt
37286: 860_sexy_primes2performance.txt
37287: 860_sexy_primes2performance.psb
37288: 860_sexy_primes.txt
37289: 860_PascalMatrix_Out.txt
37290: 860_graph3.png
37291: 826_similarity_api2.txt
37292: 860_virtualbox_overview.png
37293: 859_maputils_client.pas
37294: 861_stringtokenizer.txt
37295: 861_stringtokenizer2.txt
37296: 862_benfordslaw_synedit.txt
37297: 863_stack_tcp_chatroom.TXT
37298: 865_caesar_cipher.txt
37299: 865_winservices2.txt
37300: 866_tutor70consolepowershell2.png
37301: 780_abc_problem_delphi_synapse2.txt
37302: 781_LatLonDistmX.pas
37303: 781_JLatLon_JFK-ZRH.JPG
37304: 782_U_MakeCityLocations2mX4.pas
37305: 784_checkers.txt
37306: 786_web_scrap.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
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

```
37307: 788_PETRA.MID                                    788_pcu_mX4.pas
37308: 789_totient_primes_tester.txt                    789_totient_primes_tester.psb
37309: 790_sha512.txt                                   790_googlemaps.txt
37310: 867_metadefener_api.txt                          868_6MainFmnetworkmX47.pas
37311: 869_factsum.txt                                  870_totient_primes_tester_471_functions.txt
37312: 871_checkemptyfolders.txt                        496_installX.txt
37313: 872_RSA_codes.txt                                872_RSA_codes2.txt
37314: 873_Unittester_mX47.txt                          873_Unittester_mX47_2.txt
37315: 873_Unittester_mX47_2.psb                        873_httpclient.png
37316: 874_LinkGrabUnit1.pas                            875_filehistounit2app.txt
37317: 875_filehistounit2app_forensic.txt               875_codetester_GUI_maXbox4exe.png
37318: 876_console_output.txt                           877_flowpanel.pas
37319: 877_flowpanel2.pas                               878_code_reflection.txt
37320: 879_psvCGI.pas                                   880_cgiMain_server.pas
37321: 880_visualstatemachine_main.pas                  880_ModemEmulator.pas
37322: 880_cgiMain_server2.pas                          640_API_LayerDetection_EKON23.TXT
37323: 881_odometer_uMain2.pas                          882_bitmap_brusher.txt
37324: 883_create_dataset.txt                           884_google_chart_api.TXT
37325: 884_google_chart_api2.TXT                        885_IP_Geolocation.TXT
37326: 886_AmicablePairs.TXT                            887_Anti_Primes.TXT
37327: 888_Sundial.TXT                                  889_Jaro_distance.TXT
37328: 890_Jensens_Device.TXT                           891_uturing_machine.TXT
37329: 892_microwaves.TXT                               891_uturing_machine2.TXT
37330: 891_universal_turing_machine3.TXT                892_microwaves2.TXT
37331: 893_montecarlo_PI.TXT                            893_montecarlo_PI.psb
37332: 894_csv_to_html.TXT                              895_screenpixel.TXT
37333: 896_tickets_tree.png                             896_textclassification.py
37334: 896_all_tickets.csv                              690_calc_pi_ex_stat4.pas
37335: 897_openoffice_ole_automation.TXT                898_exit_procedure.TXT
37336: 899_remote_control_key.TXT                       569_ServiceMgr3.TXT
37337: 900_XML.TXT                                      XML.TXT
37338: 901_turtle_purple_circle.txt                     902_sha1_explicit.TXT
37339: 902_sha1_explicit_aes.TXT                        903_ODBC_list.TXT
37340: 904_Unit1NEMESIS.pas                             905_uiptools.pas
37341: 905_rfc1213util.pas                              905_uipservices.pas
37342: 905_PlotCompCode.pas                             905_net_ugraph.pas
37343: 905_net_ugraph_formplot.pas                      903_microtonal_csound.TXT
37344: 905_rfc1213sys.pas                               906_OverbyteIcsLogger.pas
37345: 907_OverbyteIcsSHA1Test1.pas                     170_4gewinnt_main3.txt
37346: 880_cgiMain_server3.pas                          880_psvCGI.pas
37347: 880_CGI1.exe                                     880_CGI1.pas 880_DrBobCGI.pas
37348: 908_Netlister.pas                                909_MandelBrot.pas
37349: 909_sinewave_circle.pas                          910_tictac.pas 910_tictac_form.pas
37350: 880_cgiMain_server4.pas                          880_CGI1.exe
37351: 910_kh_function_plot_demo2.pas                   910_kh_plot.png
37352: 910_tictactoe_form.png                           910_tictac_form2.pas
37353: 911_gapfulnumbers2ibz.pas                        912_Primestest_AKS_2.pas
37354: 912_TTCPBlockSocket_Tester3.txt                  912_ssl_blcksock_tester.pas
37355: 729_U_BigIntTest_mX4.pas                         913_al_php_runner_test.pas
37356: 913_al_web_spider_test.pas                       913_ALOpenOffice_tester.pas
37357: 915_OpenOffice_API.pas                           897_openoffice_ole_automation2.TXT
37358: 916_fmTraceRouteMainU.pas                        914_run_javascript_471.pas
37359: 916_CGIMailerfMain.pas                           917_EwbCoreTools_experimantal.pas
37360: 918_TIdDateTimeStamp_main.pas                    919_uLockBox_HugeCardinalTestCases.pas
37361: 920_TURL_class_test.pas                          919_uLockBox_RSA_TestCases.pas
37362: 920_TURL_class_test2.pas                         920_TimeServer_TickClock.pas
37363: 919_uLockBox_Signatory_TestCasesmX471.pas        919_umfmMakeSampleKey.pas
37364: 913_al_web_spider34form_trivial.pas              913_al_web_spider34form.pas
37365: 921_object_filehandling.pas                      922_u_stringlist.pas
37366: 922_u_dir.pas                                    922_u_c_url.pas
37367: 915_OpenOffice_API_2.pas                         915_OpenOffice_API_21.pas
37368: 923_indybasicclientserverfMain.pas               924_arrays.pas
37369: 925_U_DoubletsX3_2.pas                           925_U_CountPhrases.pas
37370: 925_U_CountPhrases_form.pas                      925_phrasecount2.png
37371: 926_singlesamplepredict.py                       927_primetrial.pas
37372: 927_Emirp.pas                                    927_Emirp.psb
37373: 927_Emirp2.pas                                   928_wave_async_parallel.pas
37374: 928_cd_catalog2.xml                              929_xmlstream.pas
37375: 927_Emirp2.pas                                   928_wave_async_parallel.pas
37376: 928_cd_catalog2.xml                              929_xmlstream.pas
37377: 930_service_tools.pas                            930_service_tools2.rtf
37378: 930_service_tools2.pdf                           930_service_tools2.pas
37379: 930_service_tools2.htm                           931_spiralmatrix.txt
37380: 931_spiralmatrix2.txt                            932_ordered_words4.txt
37381: 932_unixdict.txt                                 932_ordered_words4.psb
37382: 932_ordered_words42.txt                          932_getgithub2.txt
37383: 932_correlation.txt                              932_correlationplot2.png
37384: 932_correlation_regression.txt                   932_correlation_regression_light.txt
37385: 548_STExpressions_Services2.TXT                  932_getgithub22.txt
37386: 749_helloWebServer3refresh2.txt                  938_xmldemo.xml
37387: 938_xmldemo_out.xml                              938_xml_output.pas
37388: 937_xml_input.pas                                936_md5.pas
37389: 935_Primality_by_trial_division.pas              934_datamanipulation.pas
37390: 933_draw_sphere.pas                              939_soundex.pas
37391: 940_primefactors.psb                             940_primefactors.pas
37392: 941_picturebox.txt                               941_picturebox2.txt
37393: 942_gaussen.txt                                  942_gaussen_data_science.psb
37394: 942_gaussen_data_science.txt                     942_statpackage.png
37395: 942_statpackage2.png                             942_gauss_jordan2.pas
```

```
37396: 942_statpackage2.png
37397: 942_gauss_jordan.pas
37398: 942_gauss_jordan2.pas
37399: 944_Perl_Regex.txt
37400: 945_Many_Regex.txt
37401: 946_sudokuform.png
37402: 945_registrysearch_processlist2.psb
37403: 947_TWinControl_Canvas.txt
37404: 949_setbit.txt
37405: 950_uniqueinstance_pythonversions.pas
37406: 950_saint_source.txt
37407: 185_memorymax3_internet.txt
37408: 942_gaussen_data_science3.txt
37409: 952_psUFinancial.pas
37410: 953_Res2BmpUtils.pas
37411: 953_JTools.pas
37412: 955_logregclassifier.py
37413: 957_blobstream.txt
37414: 958_webcam.txt
37415: 959_convert_temperature2.txt
37416: 960_newjapan.mp3
37417: 960_neuralbitmx2.pas
37418: 960_Legendre_Gaussintegration.txt
37419: 950_saint_source2.txt
37420: 963_movie_reviewd_words42.txt
37421: 963_moviereviewswordlist2listboxsort.txt
37422: 861_stringtokenizer2primes_moviesreview2.txt
37423: 363_compress_services3.txt
37424: 964_geo_map_zoom.txt
37425: 965_RSA_IBZ_PrivatePublicKey_Form.pas
37426: 965_RSA_IBZ_PrivatePublicKey_Form3p.pas
37427: 966_GUItester2.png
37428: 967_U_RSADemo2.pas
37429: 967_CryptoGUItester4.png
37430: 967_CryptoGUItester1.png
37431: 968_recordobject.txt
37432: 969_gapfulnumbers.txt
37433: 970_integrate.txt
37434: 970_integrate_vandercorput_sequence.txt
37435: 970_U_wordladder_mX4.pas
37436: 970_book_portal_usecase2.png
37437: 970_skype_firewall.png
37438: 970_U_Invertedtext.uc
37439: 970_U_Decrypt3.pas
37440: 970_U_Decrypt4Forms.pas
37441: 971_U_DrawScrambledPie.pas
37442: 971_wordscramble_forms1.png
37443: 970_U_ScrambledPiemX4Forms2.pas
37444: 972_words_speclinelist_hashdemocount.txt
37445: 972_json_tester.txt
37446: 972_char_counts.txt
37447: 972_json_tester2.txt
37448: 973_RCDATA_resource_export64.txt
37449: 470_U_ScrollingLEDs3mX4_21.pas
37450: 975_facofinteger.txt
37451: 975_U_PrimeFactors1.pas
37452: 975_U_PrimeFactors2.pas
37453: 975_euler_pot.txt
37454: 975_euler_pot2.psb
37455: 602_multilang_game2.txt
37456: 976_stripchars.txt
37457: 977_Rosetta_Dijkstra_Console.htm
37458: 978_wininet_universaldownload.txt
37459: 979_U_DensityPlot.pas
37460: 979_U_ReactionTimes4.pas
37461: 980_Chart_DensityPlot.pas
37462: 980_U_TelephoneWords.pas
37463: 980_telephonewords.png
37464: 980_U_TelephoneWords2.pas
37465: 980_U_TelephoneWords2.psb
37466: 980_telephonewords21.png
37467: 980_Chart_DensityPlot2.pas
37468: 982_win_graphviz.txt
37469: 984_Drawaclock_computergraphic.txt
37470: RXMisc.pas
37471: 985_nist.dat
37472: 985_open_subtitles.txt
37473: 985_sha1_impl2.txt
37474: 985_sha1_impl21.psb
37475: 985_missing_permutation2.txt
37476: 986_findallfiles.txt
37477: detector_jupyter.ipynb
37478: hoppenstedtstwinwiz_out2.jpg
37479: 987_shell32_operation.txt
37480: 988_anti_aliasing.txt
37481: 990_ADO_Connection.txt
37482: 990_GUI_ADO_Output.png
37483: 990_ADO_Connection_Create_DB.txt
37484: 990_ADO_Connection_Create_DB2.txt
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
970_U_ScrambledPiemX4Forms.pas
970_U_Decrypt4Forms22.pas
972_U_HashedWordCount.pas
972_U_HashedWordCount2.pas
972_primenumber_routines.txt
973_RCDATA_resource_export.txt
974_systemsdiagram_prepare.pas
974_64bit_Routines.txt
975_facofinteger2.txt
975_facofinteger2.psb
975_U_PrimeFactors21.pas
975_euler_pot2.txt
602_multilang_game2ball.txt
602_multilang_gameball.txt
977_Rosetta_Dijkstra_Console.txt
978_ALHttpclient_cookie.txt
979_U_ResponseStats3.pas
979_UDFFRegistry.pas
979_testdetail.rsd
980_testdetail.rsd
980_telephone-keypad-letters.jpg
980_U_TelephoneWords2.rtf
980_U_TelephoneWords2.htm
980_U_TelephoneWords21.pas
980_U_TelephoneWords22.pas
981_tesseractutils.pas
983_Average_loop_length.txt
984_Drawaclock_computergraphic.psb
985_nist_regression_eval.pas
985_sha1_impl.txt
985_regex_subtitles.txt
985_sha1_impl21.txt
985_missing_permutation.txt
986_wiwin32.txt
detector21_wdef.py
objectdetector3.ipynb
twinwiz_out2.jpg
987_shell32_file_operations.txt
989_paintbox.txt
990_maxbase3.mdb
989_paintbox.png
990_SQL_Workbench_ADOCreate_Output.png
990_maxbase4runtimedb6.mdb
```

```
37485: 990_metadefender_API.txt                          986_findallfiles_metadefender.txt
37486: 990_metadefender_API_2.txt                        990_ADO_Connection_Create_DB21.txt
37487: 990_httpsend.png                                  990_httpsend2.png
37488: 990_httpsend.pac                                  990_metadefender_API_21.txt
37489: 990_winhttprequest_API_51.txt                     990_httprequest_API.txt
37490: 991_oma_chartregression2.py                       991_python_oma_svr.txt
37491: 991_python_oma_svr21.txt                          991_python_oma_svr2.txt
37492: 991_oma_charts_market_btc_150y.png                991_python_oma_svr21integrate.txt
37493: 840_RungeKuttaTest.exe                            840_URungeKutta4test2.pas
37494: 991_python_oma_svr21integrate_dotnetcore.txt      991_pyoma_svr22integrate_dotnetcore2.txt
37495: 992_detector21_wget_integrate.txt                 992_detector21_wget_integrate.htm
37496: 992_detector21_wget_integrate2.rtf                992_detector21_wget_integrate2.txt
37497: 992_detector21_wget_integrate2.htm                993_perceptron_graph2.png
37498: 993_perceptron_form.txt                           993_perceptron_graph.png
37499: 993_perceptron_form2.txt                          993_perceptron_form2.psb
37500: 994_Find_URI_in_text.txt                          995_RunLengthEncoding.txt
37501: 995_RunLengthEncoding.psb                         995_RunLengthEncoding_Pascal.txt
37502: 996_Precision_Functions.txt                       997_Base64_decode_data.txt
37503: 998_string_has_all_unique_characters.txt          998_string_has_all_unique_characters.psb
37504: 998_weatherapp474_forecast2.txt                   weatherapp474.txt
37505: 999_word_frequency.txt                            999_word_frequency2.txt
37506: 999_synedit_codecompletion.txt                    999_RotImg.pas
37507: 1000_earth_rotation.txt                           1000_GraphicsMathLibrary.PAS
37508: 1000_coffeemax.bmp                                1000_earth_rotation3d.txt
37509: 1000_3DLabScreen.txt                              1000_GraphicsMathLibrary_Pantograph.PAS
37510: 961_URLDecoding_indyresponse_WMI_Info.txt         1000_graph2.pas
37511: 1000_GenomeLibrary.pas                            1000_Coordinates222.txt
37512: 1000_chr01_GEN_sequence.txt                       1000_Coordinates_Gen.ods
37513: 1000_JVCL_Chart_Demo.PAS                          1000_JVCL_Chart_Demo2.PAS
37514: 1000_GraphicsMathLibrary_PantographF.PAS          1000_JVCL_Chart_Demo2Form.PAS
37515: 1001_entropy_test.txt                             1001_entropy_test2.txt
37516: 1001_entropy_test21.txt                           1002_protein_translation.txt
37517: 1002_protein_translation2.txt                     1000_sphere_rotation3d.txt
37518: 1000_sphere_rotation3direct.txt                   1000_Sphere_big.png
37519: 1000_sphere_sourceforge2.png                      1000_sphere_sourceforge.png
37520: 1000_under_overfit.png                            1000_Sphere_small.png
37521: 1000_Sphere_4poles.png                            1000_sphere_sourceforge.txt
37522: 1000_sphere_sourceforge2.txt                      1000_Nap.bmp
37523: 1003_molecules.txt                                1003_molecules1_2.txt
37524: 1003_molecules1_3.txt                             1003_molecules1_4.txt
37525: 1003bellsound.wav                                 1003doorbell.wav
37526: 1003templebell.wav                                1003_sound_molecules.png
37527: 1003_molecules1_5.txt                             1004_wordwheel.txt
37528: 1005_drawsphere_ascii.txt                         1006_write_PPM.txt
37529: 1006_write_PPM_2.txt                              1006_write_PPM_2read.txt
37530: 1006_rendersphere full2.png                       1006_writeSphere_PPM_2readBMP.txt
37531: 1006_writeSphere_PPM_2readBMPfix.txt              1006_writeSphere_PPM_2readBMPfix2.txt
37532: 1006spheremaxsmallout2.png                        1006_writeSphere_PPM_2readBMPfix2frm.txt
37533: 1006_bitbang_spheremaxsmallout2w.png              1006_bitbang_spheremaxsmallout2r.png
37534: 1006_bitbang_spheremaxsmallout2y.png              1007_disabled_bitmap.txt
37535: 1006_manmachine.ppm                               1008_Knuth_shuffle.txt
37536: 1009_with_multiple_ranges.txt                     1010_ScreenPixel.txt
37537: 1010_ScreenPixel2.txt                             1010_JPGClock.txt
37538: 1010_ScreenPixel2.txt                             1010_ProcsVecteurs.pas
37539: 1010_U_GraphicsEffects2.pas                       1010_geometer_tangent.png
37540: 1010_geometer_intersection2.png                   1010_circle_tangent.png
37541: 1010_circle_point.png                             1010_geometer_polygon.png
37542: 1010_U_Geometry4Forms.psb                         1010_U_Geometry4Forms.pas
37543: 1010_U_Geometry4Forms.htm                         1010_U_ClockAngles2.htm
37544: 1010_U_ClockAngles2.pas                           1010_U_DecToFract2.pas
37545: 1011_cInternetUtils.pas                           1011_flcStatistics_mX4.pas
37546: 1012_flcTimers.pas                                1012_flcCharSet.pas
37547: 1012_flcBits32.pas                                1010_U_DecToFract21.pas
37548: 1013_saint_source23tests.txt                      unittests4.txt
37549: 210_public_private_cryptosystem6_ibz_tuned.txt    1014_jcl_stringlist.txt
37550: 1014_jcl_stringlist2.txt                          1014_jcl_stringlist21.txt
37551: 1015_Scripting_Dictionary.txt                     1015_Scripting_Dictionary_RegEx.txt
37552: 1016_newsfeed_sentiment.py                        1016_newsfeed_sentiment.htm
37553: 1016_sentiment_api2_bbc_newsfeed.txt              1016_sentiment_api2_bbc_newsfeed2.txt
37554: 1017_XmlDocRssParser.pas                          1017_bbcnews.xml
37555: 1017_bbcnews2.xml                                 1017_XmlDocRssParser_BBCNewsDemo.pas
37556: 1017_RSSexportUnit2Webform.pas                    1017_RSSWexportUnit2formSSLStream.pas
37557: 1018_Twelve_statements.txt                        1018_ludic_numbers.txt
37558: 1018_Twelve_statements2.txt                       1019_math_source_cap5train.pas
37559: 1019_math_source_cap5.pas                         1018_Twelve_statements21.txt
37560: 1019_math_learn_tool.png                          1020_perlinNoise.txt
37561: 1020_FASTA_format.txt                             1020_FASTA_format2.txt
37562: 1020_Power2FirstDigits.psb                        1020_Power2FirstDigits.txt
37563: 1020_TopRankPerGroup.txt                          1020_ReST_Countries_API.txt
37564: 1020_CustomSort_Stringlist_Department.txt         1020_CustomSort_Stringlist.txt
37565: 1021_uCEFMiscFunctions_mX4.pas                    1021_uCEFMiscFunctions_mX47.pas
37566: 1022_U_SciGraph13_mX47.pas                        1022_scigraphplot.png
37567: 1023_FANN_NeuralNetworkLogTrainer.txt             1023_FANN_Alllogic2.png
37568: 1024_floydalgo.txt                                1025_Deranged_anagrams.txt
37569: 1025_Deranged_anagrams2.txt                       1026_SVC_BitVision.py
37570: 1026_json_automation_coviddata.txt                1026_covidtimeserie.txt
37571: 1026_json_automation_coviddata2.txt               1026_json_automation_coviddata21.txt
37572: 1026_worldwide5zoom.png                           1026_worldwide5.png
37573: 1026_worldwide4.png                               1026_json_automation_coviddata22.txt
```

37574: 1026_worldwide3.png
37575: 1026_worldwide.png
37576: 1026_swiss_weeklypattern.png
37577: 1026_swiss_firstwave.png
37578: 1026_jsondos_forecast.png
37579: 1026_json_automation_refactor.txt
37580: 1026_worldwide9_8confirmed.png
37581: 1017_RSSDatacampSSLStreamSentiment.pas
37582: 972_json_tester31.txt
37583: 1026_uJson_testeroauth_print.pas
37584: 1027_fpvectorial.jpg
37585: 1028_agg_math.pas
37586: 1028_JcfUnicodeFiles.pas
37587: 1028_filestream4test.pas
37588: 1028_filestream4test2.pas
37589: 1030_filepilot2.psb
37590: 1030_StrUtil.pas
37591: 1026_SVC_BitVision_MLPrimer.py
37592: 1031_Bitcoinaddress_validation.txt
37593: 1032_population_count.txt
37594: 1033_Corpus_from_Dir.py
37595: 1034_tamath.pas
37596: 1034_tamath_safe.pas
37597: 1035_covidapp.txt
37598: 1035_covidapp3.htm
37599: 652_graph3Dspiral2.pas
37600: 1037_bufferoverflow_tester.txt
37601: 1035_covidapp31.txt
37602: 1039_pcatest.pas
37603: 1039_pcatest3.pas
37604: 1039_roman.pas
37605: 1016_newsfeed_sentiment_integrate2.py
37606: 1016_newsfeed_sentiment_integrate2.htm
37607: 1041_signproof_tester.txt
37608: 1042_superpermutation_minim.txt
37609: 1034_weather_report_py.png
37610: weatherbox.txt
37611: 1045_XMLTreeview2.png
37612: xmldemo.txt
37613: 1045_semaphorgrids.pas
37614: weatherapp475.txt
37615: 1046_openweather_integrate.txt
37616: 1046_openweather_P4D_2.png
37617: 1046_JclTimeZones.pas
37618: 1047_microtonal_tomath_mX4.txt
37619: 1047_microtonal_tomath_mX4_2.txt
37620: 1049_InetUtils_Callback.TXT
37621: 1050_saint_Pythonengine.txt
37622: 1050_pydemo2.txt
37623: 974_64bit_Routines2.txt
37624: 1051_semaphorgrid.txt
37625: 778_advapi32_dll_SHA512_plusRefactor.txt
37626: 1052_delphi_arduino2.txt
37627: 1052_delphi_arduino2.png
37628: 1052_delphi_arduino3.txt
37629: 1052_delphi_arduino31.txt
37630: 1052_delphi_arduino32.ini
37631: 1052_bodmas.txt
37632: 1050_pydemo_5_powerliners.txt
37633: 1052_OoMiscpas2PyHTTPSServer2LAB.txt
37634: 1052_pyhttpserver_integrate22.py
37635: 1053_U_TimeZoneDemo2.pas
37636: 1054_U_SoundGen31.pas
37637: 1054_U_SoundGen_mcisendstring32.pas
37638: 1055_U_IntListtest.pas
37639: 1055_tintlistdemo.png
37640: 1052_pydemo_pythonversion.txt
37641: 1056_neuralvolume_routines.pas
37642: 1057_pydemo_7_pyIO.txt
37643: 1058_Arduino.pas
37644: 1060_pypas_demo10.psb
37645: 1059_pypas_demo11.psb
37646: 1059_pypas_demo11.txt
37647: 1060_arduinoUnit1.pas
37648: 1062_utils.py
37649: 1061_gameoflifeUNIT1.PAS
37650: 1062_intervallunit2.txt
37651: 87_1062_neuraltrainer.png
37652: 1062_neuralnet_vectors2.txt
37653: 1064_httpd24.pas
37654: 1065_neuralnetwork_CAI.txt
37655: 1066_KMeans_tester.txt
37656: 1067_py_planet_moon.txt
37657: 1068_vector_matrix_tester.txt
37658: 1068_floattostr_tester.txt
37659: 1063_array_lists3.txt
37660: 1070_U_PerfectDeck.pas
37661: 1070_tprime_tshirt.png
37662: 1070_U_Keno2mX4.pas

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
336_digiclock3resource.txt
1050_saint_Pythonengine2.txt
1050_pydemo21.txt
1050_pydemo_51.txt
778_advapi32_dll_SHA512_plus.txt
1052_delphi_arduino21.png
1052_delphi_arduino.txt
1052_OoMiscpas2.txt
1052_delphi_arduino31_2.png
1052_delphi_arduino31.png
1052_delphi_arduino32.txt
1050pytest21_5powershttpserve.txt
1050_pydemo_5_powerliners23.txt
1052_OoMiscpas2output.txt
1050_pydemo_10_pythontips.txt
1054_U_SoundGen_mcisendstring31.pas
1055_WaveStorage.pas
1055_U_GridWordHighlight.pas
1055_tintlistdemo2.png
1055_U_IntListtest.psb
1052_pydemo_pythonversion2.txt
1019_math_source_cap5_pulse.pas
1057_pydemo_7_pyIO.psb
1060_pypas_demo10.txt
1060_pypas_demo10.htm
1059_pypas_demo11.htm
1059_pydemo11tester.png
1061neuraldatasets_pretester.pas
1062_3DCellularUnit1.pas
1062_intervallunit.txt
1062_neuralnet_vectors.txt
1063_vector_lists.txt
1064_vectorclassroutines.txt
httpd24.pas
1065_neuralnetwork_CAI_2.txt
1066_neuralvolume_tester.txt
1052_pydemo_pythonversion31.txt
1068_vector_matrix_tester2.txt
1063_vector_lists3.txt
1070_U_CardProbability2.pas
1070_tshirt_prime.txt
1069_pydemo30_performance.txt
1065_neuralnetwork_CAI_2_data.txt

```
37663: 1065__CAI_2_SimpleImageClassifier.txt            1066_KMeans_tester2.txt
37664: 1065__CAI_2_SimpleImageClassifier2.txt           1070_U_Keno2mX4Form.pas
37665: 1071_Newadds_V47590_Modules.pas                  1070_U_Keno2mX4_Forms2.pas
37666: 1070_tshirt_prime2_tutor88.txt                   1070_U_PrimesFromDigits.pas
37667: 1071_Newadds_V47590_Modules.pas                  1071_Newadds_V47590_Modules2.pas
37668: 1065__CAI_2_SiImageClassifier21_Tutor_89_test3.txt  1072_captcha.txt
37669: 1072_captcha2.txt                                weatherapp476.txt
37670: 1073__CAI_3_LearnerClassifier22_Tutor_89_2.txt   1074__CAI_3_visualautoencoder.txt
37671: 1074__CAI_3_visualautoencoder2.txt               1071_Newadds_V47610_Modules3.pas
37672: 1074_U_Systools_mX4Similarity.pas                1074_U_Systools_mX4Similarity2.pas
37673: 1075_primes30_performance5.txt                   1065__CAI_2_SimpleImageClassifier21_EKON_blogstarter.txt
37674: 1076_CAI_Ugradientascent.pas                     1076_CAI_UVisualGAN2.pas
37675: 1076_CAI_UVisualGAN2tester_MNIST.pas             1076_CAI_UVisualGAN2tester_MNIST2.pas
37676: 1077_U_Catapult4.pas                             1077_U_Catapult41.pas
37677: 1078_create_DB.pas                               1077_U_Catapult41forms.pas
37678: 1077_U_Catapult42forms.pas                       1077_U_Catapult43forms.pas
37679: 1077_U_metronome2.pas                            1078_bell_numbers.pas
37680: 1079_Fregan.pas                                  1079_Test.txt
37681: 1077_U_Catapult43forms2.pas                      1077_U_metronome21.pas
37682: 1080_trend.pas                                   1080_hist.txt
37683: 1080_hist.exe                                    1080_histdatgui.png
37684: 1080_tprob.pas                                   1080_fmTraceRouteMainU.pas
37685: 1080_VCLScanner2021.txt                          1080_VCLScanner2021.uc
37686: 1080_syndat_concept.png                          1081_syndat_pydemo32_2.txt
37687: 1081_syndat_pydemo32_2.psb                       781u_LatLonDistanceTestmX21.pas
37688: 1082_Idiomatically.txt                           1082_ToolsUtils.pas
37689: 1082_ToolsUtils2.pas                             1071_Newadds_V47610_Modules42.pas
37690: 1083_Deltics_TCommandline.pas                    1084_SeAES256_explain.pas
37691: 1084_DelticsStrUtils.pas                         602_moonbug25.txt
37692: 1085_kmemo_dictionary.pas                        1085_kmemo_dictionary2taunumber.pas
37693: 1086_bigdecimals2.pas                            1086_bigdecimals2python.pas
37694: 1086_bigdecimals2python2.pas                     1087_HQ9.pas
37695: 1088__CAI_3_Hypotenuse.txt                       1088__CAI_3_DenseNetBCL40.txt
37696: 1088__CAI_3_DenseNetBCL40_2.txt                  1050_pydemo_5_powerliners_def_2.txt
37697: 1089_RSA_Example_crypto.htm                      1089_RSA_Example_crypto.txt
37698: 1090_simpleneuralwebserver.txt                   1089_RSA_Example_crypto_Herdt.txt
37699: 1090_simple_neuralwebserver2.txt                 1090_uarduinoformworkspace.pas
37700: 1090_uformlamwinosettingspaths.pas               1090_Fractran_test.txt
37701: 1090_Fractran_test_Python.txt                    1090_U_Splines.pas
37702: 1090_U_Numbrix2.pas                              1090_numbrix_GUI.png
37703: 1090_Numbrix2.txt                                1090_U_Numbrix21.pas
37704: 1090_numbrix_GUI_24.png                          1090_Numbrix2data.txt
37705: 1091_U_SpringMass21.pas                          1091_springmassdemo.png
37706: 1091_U_SpringMass2.pas                           1092_Rollercoaster_Simulator.pas
37707: 1092_Rollercoaster_Simulator21.pas               1092_Rollercoaster_Simulator21.htm
37708: 1093_XMLUtils_Tutor92tester.txt                  1094_pydemo13tutor_PAS2JS.txt
37709: 1095_faker_tutor_PAS2JS.txt                      1096_MainFormSVG_2.pas
37710: 1096_VCL4Python.pas                              1096_P4D_VCL4Python.pas
37711: 1096_MainFormSVG_2test4Python.pas                1093_CAI_IdentityShortcutConnection2.pas
37712: 1096_pychats_demo4VCL4Python.png                 1096_standalonepyvclexe.png
37713: 1096_MainFormSVG_2test4Python_VCL4Python.pas     1096_MainFormSVG_2test4Python_VCL4Python2.pas
37714: 1100_901_turtle_purple_circle_EKON23.txt         1100_VectorProduct.pas
37715: 1100_U_Compass.pas                               1100_U_Compass2.pas
37716: 1097_Pierpont_primes.pas                         1097_Pierpont_primes2.pas
37717: 1098_image_noise.pas                             1098_jaro_similarity.txt
37718: 1098_image_noise2xmlnodes.pas                    1099_rest_localdb.pas
37719: 1001_ESP32_Loop.TXT                              1069_pydemo30_matrixperformance4.txt
37720: 1102_matrixmultiplication4.txt                   781u_LatLonDistanceTestmX22.pas
37721: 1102_matrixmultiplication41.txt                  1101_PJConsole.txt
37722: 1101_PJConsole1.txt                              1103_DudsCommonUtils.pas
37723: 1104_TNeuronClass.txt                            1104_TNeuronClass2.txt
37724: 281_picturepuzzle31res.txt                       1101_PJConsole12.txt
37725: 1101_PJConsoleApp_Echoer.png                     Echoer.exe
37726: 1101_PJConsole12Json.txt                         1105_HTML_Builder.txt
37727: 1105_HTML_Builder2DB.txt                         1101_PJConsoleApp_Echoer2.png
37728: 1101_PJConsole1ex2.txt                           1101_PJConsole1_demo9.txt
37729: 1105_HTML_Builder2Modules.txt                    1106_OnlineRandomizer.pas
37730: 1107_Iris_Console.pas                            1107_bezdekIris.fann.data.txt
37731: 1107_Iris_Console2.pas                           1106_OnlineRandomizer_Random_Main.pas
37732: 752_U_AstroDemo_mX4_41.pas                       752_astronomy_moon23.png
37733: 1108_CAI_Geocoding.pas                           1108_geocoding_api.png
37734: 1109_winapidownload.pas                          1110_JSONObject.txt
37735: 1111_lifeappearance_Unit1.pas                    1111_uvisualcifar10animalmachine.pas
37736: 1111_UVPlan1test_QR_Code.pas                     1111_URSOM_Network.pas
37737: 1071_Newadds_V47610_Modules47.pas                100U_CoinSearch.pas
37738: 1012_UTicTacToeCount.pas                         1012_delphi-utils.pas
37739: 1108_CAI_GeocodingTWininetAPI.pas                1108_CAI_GeocodingTWininetAPI_only.pas
37740: 1112_lastUtils.pas                               1112_lastUtils2.pas
37741: 100U_CoinSearch2.pas                             1111_machineanimal.png
37742: 1111_animalmachine_pictures12.pas                1011_U_CoinSearc_form.pas
37743: 1011_graphnodes_solution.png                     1011_U_CoinSearch_form.pas
37744: 1011_graphnodes_solution1.png                    1112_U_Wordgrid.pas
37745: 1112_U_Wordgrid.htm                              1112_wordgrid.png
37746: 1112_wordgrid1.png                               1112_full.dic
37747: 1113_composition_tree.pas                        1113_composition_tree1.pas
37748: 1071_Newadds_V47610_Modules47tester.pas          ubuntu16_session_maxbox4_cloud2022_full2.png
37749: 1114_polynomial_synthetic_division.txt           1115_Wireworld.txt
37750: 1115_unitKWKalender.pas                          1115_scriptDatenbankdiagnose.pas
37751: 1088__CAI_3_supersimplecorrelation.txt           1088__CAI_3_supersimplehyperbolictangent.txt
```

```
37752: 1115_gecodingapi_tee_helvetia_json.txt          1116_U_ChallengingmathTeasers.pas
37753: 1117_sunpos.png                                 1117_sunpos_bern.png
37754: 1117_U_SolarPos2Bern.pas                        1117_U_SolarPos2.pas
37755: 1117_sunpos_bern2.png                           1117_U_SolarPos2Bern2.pas
37756: 1118_Cumulative_standard_deviations.pas         1118_Cumulative_standard_deviations.htm
37757: 1108_CAI_GeocodingTWininetAPI_Maponly.pas       1117_StAstro.pas
37758: 1119_FloatToBinTests.pas                        1120_Department_numbers.pas
37759: 93_1108_CAI_GeocodingTWininetAPI_Maponly.pas.png 1120_DammAlgorithm.pas
37760: 1120_DammAlgorithm12.pas                        sentiment3.txt, sentiment4.txt
37761: 1121_sentiment_api3_bbc_newsfeed4rec.txt        1121_sentiment_api3_bbc_newsfeed4.txt
37762: 1121_sentiment_api3_bbc_newsfeed4rec2.txt       1122umain.pas
37763: 1121_sentiment_api3_bbc_newsfeed4rec21.rtf      1121_sentiment_api3_bbc_newsfeed4rec21.txt
37764: 1122_langtranslator.txt                         1122_langtranslator1.txt
37765: 1123_IntegerArraySort.pas                       1123_MATHBOX.PAS
37766: 1123_CODEWIZU.PAS                               1123_CODEWIZU_2.PAS
37767: 1124_CODEReflection.PAS                         1125_Forward_Difference.pas
37768: 1124_CODEReflection1.PAS                        1125_unitGlobal.pas
37769: 1125_unitGlobal1.pas                            1125_unitGlobal12.pas
37770: 1125_unitGlobal13.pas                           1100_U_Compass1_deltics_sysutils.pas
37771: 1126_Projectile_Physics.pas                     1126_Projectile_Physics2.pas
37772: 1126_Projectile_Physics21.pas                   1126_projectile2.png
37773: 1126_fetchapi1.png                              1126_fetch_api1.txt
37774: 727_mondrian_next.png                           727_mondrian_next21.png
37775: 727_Canvas_Artist_mondrian_gen.TXT              727_Canvas_Artist_mondrian_gen2.TXT
37776: 1127_RegistryWatchDog.pas                       727_Canvas_Artist_mondrian_gen21.TXT
37777: 1128_fpvutils.pas                               1128_mean_angle1.txt
37778: 1129_urlscan_api_json.txt                       1129_urlscan_api_json1.txt
37779: 1130_Scanner_gen21.TXT                          1130_mondrian_QRCode_P.png
37780: 1130_mondrian_QRCode_P2.png                     1130_Scanner_QR_21.TXT
37781: 1130_urlscan_api_json1_geoip.txt                1130_urlscan_api_json1_geoip1.txt
37782: 1131_rosetta_numerical_integration.txt          1132_rosetta_Farey_sequence.txt
37783: 1133_klib_utilities.txt                         1133_klib_utilities2.txt
37784: 1134_syn_dat_pop.txt                            1134_Cifar10Resize.pas
37785: 1134_Cifar10Resize_win_lessthread.pas           1135_Cifar10SeparableConvolution.pas
37786: 1135_Cifar10SeparableConvolution_50.pas         1136_hamming_numbers1.txt
37787: 1136_Chowla_numbers1.txt                        1135_uvisualcifar10test_mX4_1.psb
37788: 1135_visualCNN5.png                             1135_uvisualcifar10test_mX4_1.pas
37789: SimpleSeparableImageClassifier124_50_21.nn      1137_gauss_legendre.pas
37790: 1138_ocr_tesseract.txt                          1138_testmap2.png
37791: 1138_bmp.bmp                                    1139_pentagram.txt
37792: 1140_price_fraction.txt                         1140_Statistics.pas
37793: 1140_Statistics1.pas                            1140_USolarSystem.pas
37794: 1141_adafruit_tempsensor21.txt                  1140_PlanetData.pas
37795: 1140_Om_AstronomicalAlgorithms.pas              1140_doubleVector3D.pas
37796: 1140_quaternion_3D.pas                          1141_fast_sin.pas
37797: 1142_anti_primes.pas                            1139_pentagram1.txt
37798: 1142_list_virtualconstructors_test.pas          1142_setbit_comparison.pas
37799: 1142_list_collections_pydemo42.txt              1140_doubleVector3D_collection.pas
37800: 1143_KLibVC_Redist.pas                          1143_KLibVC_Redist2.pas
37801: 1071_Newadds_V47610_Modules48.pas               1071_Newadds_V47610_Modules49.pas
37802: 1145_271_closures_study_op_sys_tutor97.txt      1142_list_collections_pydemo42.txt
37803: 1144_AzuliaUtils_mX4.pas                        1145_271_closures_study_op_sys_tutor97_1.txt
37804: 1146_filesize_distribution.txt                  1147_practicalnumbers_2.txt
37805: 1148_julia_set.txt                              1148_julia_set.png
37806: 1148_julia-set.bmp                              1149_sum_to_100.txt
37807: 1150_DigitalRoot.txt                            1150_pydemo42httpheader.txt
37808: 1150_pydemo42httpheader1.txt                    1150_Partition_function_P.txt
37809: 1150_Partition_function_P1.txt                  1151_rapid_api_proto.txt
37810: 1151_sentiment_api4simple.txt                   1151_sentiment_api4simple1.txt
37811: 1151_rapid_api_proto1.txt                       1150_pydemo42httpheader2openssl.txt
37812: 1152_dos_output_callback.txt                    629_Word_FORMULA2super.PAS
37813: 1151_rapid_api_proto_googleapi12.txt            1152_shell_composer.txt
37814: 1153_space_invaders3.txt                        170_4gewinnt_main21.txt
37815: 170_4gewinnt_main21.htm                         170_4gewinnt_main31.txt
37816: 1142_list_collections_pydemo42_1.txt            1154_energy_api_agsi.txt
37817: 1154_energy_api_agsi_plot.txt                   1154_energy_api_agsi_plot1.txt
37818: 1154_energy_api_agsi_plot12.txt                 1154_energy_api_agsi_plot13.txt
37819: 1155_census_data.txt                            1155_census_data_graph.txt
37820: 1155_Utils47620.pas                             1155_dynmatrix47620.pas
37821: 1156_SearchUtils.pas                            1157_wininet_internetreadfile_dll.pas
37822: 1058_teetest_screen21map.TXT                    1158_resourceloader2.pas
37823: 1159_uvisualgan_mX4.pas                         1159_SearchUtils_Searchcode.pas
37824: 1159_uvisualgan_mX4_1.pas                       1160_google_safebrowsingapi12.txt
37825: 1161_wmiserv_commandlinelist.txt                1161_workshop_WMI_Proc32.png
37826: 1162_TIXML.TXT                                  1161_templatec_hack1_meterpreter.txt
37827: 1063_cryptosystem2testcrypto3.txt               1161_windefend_check.txt
37828: 1163_tcptable_dll.txt                           845_TestPermutations3.txt
37829: 1164_filelessStream.txt                         1164_filelessStream2.txt
37830: 1165_UAC_enabled.txt                            1166_farey_sequence.txt
37831: 1167_vcluster_py.txt                            1167_penguins_data_graphtester_py.txt
37832: 1168_teechart_demo3.txt                         1168_teechart_demo31.txt
37833: 1169_test_postrequest2.txt                      1169_test_postrequest3.txt
37834: 1169_test_postrequest2.txt                      1169_test_postrequest3.txt
37835: 1169_resourceloader2postmethod.pas              1168_teechart_demo31.txt
37836: 1170_UUtils_mX4.pas                             1071_Newadds_V47610_Modules50.pas
37837: 1170_UUtils_mX4_tester2.pas                     1170_uRegistry.pas
37838: 1170_wdc_GetRemoteFileSize.txt                  1170_wdc_GetRemoteFileSize2.txt
37839: 965_RSA_IBZ_PrivatePublicKey_Form4p_eng.pas     965_RSA2_4.png
37840: 1154_energy_api_agsi_plot14wininet.txt          1154_energy_api_agsi_plot14wininet2.txt
```

```
37841:
37842: 685 uPSI_ALQuickSortList;                        //alcinoe for list of double fast
37843: with TALDoubleList.create do begin  //singlelist
37844:   with CL.AddClassN(CL.FindClass('TALBaseQuickSortList'),'TALDoubleList') do begin
37845:     Function IndexOf( Item : Double) : Integer');
37846:     Function IndexOfObject( AObject : TObject) : Integer');
37847:     Function Add( const Item : Double) : Integer');
37848:     Function AddObject( const Item : Double; AObject : TObject) : Integer');
37849:     Function Find( item : Double; var Index : Integer) : Boolean');
37850:     Procedure Insert( Index : Integer; const item : Double);
37851:     Procedure InsertObject( Index : Integer; const item : Double; AObject : TObject);
37852:     RegisterProperty('Items', 'Double Integer', iptrw);
37853:     SetDefaultPropery('Items');
37854:     RegisterProperty('Objects', 'TObject Integer', iptrw);
37855:   end;
37856:
37857: https://sourceforge.net/projects/maxbox/files/Examples/13_General/
37858: http://www.softwareschule.ch/examples/signtoolbatch.txt
37859: http://www.softwareschule.ch/examples/signtoolbatch.htm
37860:
37861: https://cdn.top4download.com/img/award_120x60_pick.gif
37862: http://www.kleiner.com/boxart.htm
37863: http://max.kleiner.com/boxart.htm
37864: http://www.kleiner.ch/kleiner/boxart7.htm
37865:
37866: All 34 Articles of maXbox Pascal in Blaise Pascal Magazine!
37867: ----------------------------------------------------------
37868: Using Examples and Sources from Magazine CatalogX
37869: Art of Coding V.31/32  5/6 2013
37870: DLL for All V.33
37871: QR Code Coding V.34
37872: GEO Maps Nav V.35
37873: 3D Printing Lab V.36
37874: RegEx Report REX V.37/38
37875: Func Testing V.40
37876: Arduino Coding V.45/46
37877: Time is on my side V.48
37878: Compute Big Numbers V.52
37879: Code Metrics V.57
37880: Work with WineHQ V.60
37881: Work with Win WMI V.63
37882: Work with Win WMI II V.64
37883: Work MS CryptoAPI V.65
37884: The Perceptron! V.72
37885: No GUI Shell V.80
37886: Machine L. with CAI V.87/88
37887: Portable Pixmap PPM V.90
37888: Fundamentals Lib V.91
37889: RSS Feed of BBC News V.92
37890: RSS Feed Sentiment V.93
37891: JSON4Pascal V.94/95
37892: Python4Delphi 96
37893: Python4Delphi II 97
37894: Python4maXbox III 98
37895: Image Classification for Lazarus 99/100 !
37896: Faker SynDat 101
37897: Catapult, Coaster Simulator, VCL4Python 102
37898: Geocoding II 103/104
37899: CIFAR 10 Image Classifier 105
37900: Energy_API_agsi_Plot 106
37901: Cluster Classify 107
37902: Penguin Data Science SVG Plot 108
37903:
37904: Proposal for Boolean Logic Abreviation Symbol:
37905: ---------------------------------------------------
37906:
37907:   01 FALSE    //Contradiction
37908:
37909:   02 AND      //Conjunction x*y
37910:
37911:   03 INHIB    //Inhibition x*^y
37912:
37913:   04 PRAEP    //Praependence x
37914:
37915:   05 PRAE     //Praesection ^x*y
37916:
37917:   06 POST     //Postpendence y
37918:
37919:   07 XOR      //Exclusive OR x*^y+^x*y
37920:
37921:   08 OR       //Disjunction OR = x+y
37922:
37923:   09 NOR      //Rejection
37924:
37925:   10 AEQ      //Aequivalence x<-->y, ^x*^y+x*y
37926:
37927:   11 NEGY     //YNegation ^y
37928:
37929:   12 IMPY     //YImplication y-->x; x+^y
```

```
37930:
37931:   13 NEGX     //Xnegation ^x
37932:
37933:   14 IMPX     //XImplication x-->y; ^x+y
37934:
37935:   15 NAND     //Exclusion
37936:
37937:   16 TRUE     //TRUE Tautologic
37938: -------------------------------------------------------
37939:
37940: http://sourceforge.net/projects/maxbox/files/Examples/
37941: http://sourceforge.net/projects/maxbox/files/Examples/
37942:
37943: Help Online: http://www.softwareschule.ch/maxbox_functions.txt
37944:
37945: WebScript Examples: Get Web Script
37946: http://www.softwareschule.ch/examples/performer.txt;
37947: http://www.softwareschule.ch/examples/turtle.txt;
37948: http://www.softwareschule.ch/examples/SQLExport.txt;
37949: http://www.softwareschule.ch/examples/Richter.txt;
37950: http://www.softwareschule.ch/examples/checker.txt;
37951: http://www.softwareschule.ch/examples/demoscript.txt;
37952: http://www.softwareschule.ch/examples/ibzresult.txt;
37953: http://www.softwareschule.ch/examples/performindex.txt
37954: http://www.softwareschule.ch/examples/processlist.txt
37955: http://www.softwareschule.ch/examples/game.txt
37956: http://www.softwareschule.ch/examples/GEOGPS.txt
37957: http://www.softwareschule.ch/examples/turtle2.txt
37958: http://www.softwareschule.ch/examples/turtle3.txt
37959: http://www.softwareschule.ch/examples/asyncterminal.txt
37960: http://www.softwareschule.ch/examples/snowflake.txt
37961: http://www.softwareschule.ch/examples/arduinoled.txt
37962: http://www.softwareschule.ch/examples/moon2.txt
37963: http://www.softwareschule.ch/examples/cockpit.txt
37964: http://www.softwareschule.ch/examples/tartaruga.txt
37965: http://www.softwareschule.ch/examples/surprise.txt
37966: http://www.softwareschule.ch/examples/weatherapp2.txt
37967: http://www.softwareschule.ch/examples/weatherapp3.txt
37968: http://www.softwareschule.ch/examples/bigint.txt
37969: http://www.softwareschule.ch/examples/weatherapp.txt
37970: http://www.softwareschule.ch/examples/weatherapp.txt
37971: http://www.softwareschule.ch/examples/crypto.txt
37972: http://www.softwareschule.ch/examples/stack.htm
37973: http://www.softwareschule.ch/examples/fisher.htm
37974: http://www.softwareschule.ch/examples/arduinotemp.htm
37975: http://www.softwareschule.ch/examples/web.txt
37976: http://www.softwareschule.ch/examples/web3.txt
37977: http://www.softwareschule.ch/examples/web4.txt
37978: http://www.softwareschule.ch/examples/cryptocomplete.txt
37979: http://www.softwareschule.ch/examples/web4arduino.txt
37980: http://www.softwareschule.ch/examples/adafruit.txt
37981: http://www.softwareschule.ch/examples/stack.txt
37982: http://www.softwareschule.ch/examples/stack.htm
37983: http://www.softwareschule.ch/examples/billiard.txt
37984: http://www.softwareschule.ch/examples/systeminfo_tester.htm
37985: http://www.softwareschule.ch/examples/systeminfo.htm
37986: http://www.softwareschule.ch/examples/elevator.txt
37987: http://www.softwareschule.ch/examples/sha256.txt
37988: http://www.softwareschule.ch/examples/robo.txt
37989: http://www.softwareschule.ch/examples/robo2.txt
37990: http://www.softwareschule.ch/examples/font.txt
37991: http://www.softwareschule.ch/examples/spider.txt
37992: http://www.softwareschule.ch/examples/totient.txt
37993: http://www.softwareschule.ch/examples/811_mXpcatest_dmath_datascience.txt
37994: http://www.softwareschule.ch/examples/datascience.txt
37995: http://www.softwareschule.ch/examples/pi.txt
37996: http://www.softwareschule.ch/examples/excelexport.txt
37997: http://www.softwareschule.ch/examples/tokenizer.htm
37998: http://www.softwareschule.ch/examples/snake.txt
37999: http://www.softwareschule.ch/examples/asteroid.txt
38000: http://www.softwareschule.ch/examples/weatherapp46.txt
38001: http://www.softwareschule.ch/examples/cryptocomplete.txt
38002: http://www.softwareschule.ch/examples/web4arduino.txt
38003: http://www.softwareschule.ch/examples/wmi.txt
38004: http://www.softwareschule.ch/examples/wmi2.txt
38005: http://www.softwareschule.ch/examples/reverse.htm
38006: http://www.softwareschule.ch/examples/weatherapp4.txt
38007: http://www.softwareschule.ch/examples/tictac.txt
38008: http://www.softwareschule.ch/examples/tictac3.txt
38009: http://www.softwareschule.ch/examples/mapbox2.txt
38010: http://www.softwareschule.ch/examples/neuralnetwork.txt
38011: http://www.softwareschule.ch/examples/datascience.txt
38012: http://www.softwareschule.ch/examples/billiard.txt
38013: http://www.softwareschule.ch/examples/sentiment2.txt
38014: http://www.softwareschule.ch/examples/python.htm
38015: http://www.softwareschule.ch/examples/asteroid.txt
38016: http://www.softwareschule.ch/examples/penny.txt
38017: http://www.softwareschule.ch/examples/snake3.txt
38018: http://www.softwareschule.ch/examples/unittest.txt
```

```
38019: http://www.softwareschule.ch/examples/410_titanic_keras_predictor.txt
38020: http://www.softwareschule.ch/examples/410_titanic_keras_predictor.htm
38021: http://www.softwareschule.ch/examples/histogram.txt
38022: http://www.softwareschule.ch/examples/odometer.txt
38023: http://www.softwareschule.ch/examples/chartapi.txt
38024: http://www.softwareschule.ch/examples/geoloc.txt
38025: http://www.softwareschule.ch/examples/xml.txt
38026: http://www.softwareschule.ch/examples/4gewinnt.txt
38027: http://www.softwareschule.ch/examples/cgi.txt
38028: http://www.softwareschule.ch/examples/singlesamplepredict.py
38029: http://www.softwareschule.ch/examples/perform.txt
38030: http://www.softwareschule.ch/examples/statistic.txt
38031: http://www.softwareschule.ch/examples/picbox.txt
38032: http://www.softwareschule.ch/examples/memory2.txt
38033: http://www.softwareschule.ch/examples/getgeomap.txt
38034: http://www.softwareschule.ch/examples/getzip.txt
38035: http://www.softwareschule.ch/examples/getrsa.txt
38036: http://www.softwareschule.ch/examples/getrsa5.txt
38037: http://www.softwareschule.ch/examples/getrsa5.htm
38038: http://www.softwareschule.ch/examples/scramble.txt
38039: http://www.softwareschule.ch/examples/factor2.txt
38040: http://www.softwareschule.ch/examples/bug.txt
38041: http://www.softwareschule.ch/examples/smalldic.txt
38042: http://www.softwareschule.ch/examples/metadefender.txt
38043: http://www.softwareschule.ch/examples/pythonoma.txt
38044: http://www.softwareschule.ch/examples/992_detector21_wget_integrate2.htm
38045: http://www.softwareschule.ch/examples/perceptron.txt
38046: http://www.softwareschule.ch/examples/perceptron2.txt
38047: http://www.softwareschule.ch/examples/perceptron2.htm
38048: http://www.softwareschule.ch/examples/sphere.txt
38049: http://www.softwareschule.ch/examples/entropy.txt
38050: http://www.softwareschule.ch/examples/protein.txt
38051: http://www.softwareschule.ch/examples/wordwheel.txt
38052: http://www.softwareschule.ch/examples/soundcloud.txt
38053: http://www.softwareschule.ch/examples/geometry.txt
38054: http://www.softwareschule.ch/examples/time.txt
38055: http://www.softwareschule.ch/examples/sphere2.txt
38056: http://www.softwareschule.ch/examples/sphere2.htm
38057: http://www.softwareschule.ch/examples/unittests.txt
38058: http://www.softwareschule.ch/examples/newssentiment.txt
38059: http://www.softwareschule.ch/examples/newssentiment.htm
38060: http://www.softwareschule.ch/examples/bbcnews.txt
38061: http://www.softwareschule.ch/examples/mathtool.txt
38062: http://www.softwareschule.ch/examples/newssentiment2.txt
38063: http://www.softwareschule.ch/examples/covid2.txt
38064: http://www.softwareschule.ch/examples/covid3.txt
38065: http://www.softwareschule.ch/examples/covidapp3.txt
38066: http://www.softwareschule.ch/examples/signtoolbatch.txt
38067: http://www.softwareschule.ch/examples/openweather.txt
38068: http://www.softwareschule.ch/examples/weatherbox.txt
38069: http://www.softwareschule.ch/examples/wininet.txt
38070: http://www.softwareschule.ch/examples/digiclock.txt
38071: http://www.softwareschule.ch/examples/pydemo.txt
38072: http://www.softwareschule.ch/examples/pydemo2.txt
38073: http://www.softwareschule.ch/examples/pydemo3.txt
38074: http://www.softwareschule.ch/examples/pydemo13.txt
38075: http://www.softwareschule.ch/examples/pydemo31.txt
38076: http://www.softwareschule.ch/examples/catapult.txt
38077: http://www.softwareschule.ch/examples/pydemo33.txt
38078: http://www.softwareschule.ch/examples/syndat.png
38079: http://www.softwareschule.ch/examples/cryptosystem2.txt
38080: http://www.softwareschule.ch/examples/cryptosystem3.txt
38081: http://www.softwareschule.ch/examples/springdemo.txt
38082: http://www.softwareschule.ch/examples/coaster.txt
38083: http://www.softwareschule.ch/examples/pydemo38.txt
38084: http://www.softwareschule.ch/examples/pydemo40.txt
38085: http://www.softwareschule.ch/examples/sunpos.txt
38086: http://www.softwareschule.ch/examples/geocoding.txt
38087: http://www.softwareschule.ch/examples/sentiment3.txt
38088: http://www.softwareschule.ch/examples/sentiment4.txt
38089: http://www.softwareschule.ch/examples/qrcode.txt
38090: http://www.softwareschule.ch/examples/json3.txt
38091: http://www.softwareschule.ch/examples/json4.txt
38092: http://www.softwareschule.ch/examples/shellcomposer.txt
38093: http://www.softwareschule.ch/examples/4gewinnt2.txt
38094: http://www.softwareschule.ch/examples/space2.txt
38095: http://www.softwareschule.ch/examples/pydemo48.txt
38096: http://www.softwareschule.ch/examples/json7.txt
38097:
38098: https://maxbox4.wordpress.com">maxbox4.wordpress.com
38099: https://softwareschule.code.blog">softwareschule.code.blog
38100: https://my6.code.blog">my6.code.blog
38101: https://github.com/maxkleiner/maXbox4/blob/master/newstoday_sentiment.ipynb
38102: http://www.softwareschule.ch/download/exampleedition2016.zip
38103: https://github.com/maxkleiner/fundamentals5
38104: https://thenewstack.io/which-programming-languages-use-the-least-electricity/
38105: https://www.clevercomponents.com/articles/article052/
38106:
38107:
```

```
38108: Delphi Basics Run Time Library listing
38109: ******************************************************************************
38110: A
38111: Compiler Directive $A Determines whether data is aligned or packed
38112: Compiler Directive $Align Determines whether data is aligned or packed
38113: Compiler Directive $AppType Determines the application type : GUI or Console
38114: Proc SysUtils Abort Aborts the current processing with a silent exception
38115: Func System Abs Gives the absolute value of a number (-ve sign is removed)
38116: Directive Abstract Defines a class method only implemented in subclasses
38117: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
38118: Func System Addr Gives the address of a variable, Func or procedure
38119: Keyword And Boolean and or bitwise and of two arguments
38120: Type System AnsiChar A character type guaranteed to be 8 bits in size
38121: Func SysUtils AnsiCompareStr Compare two strings for equality
38122: Func SysUtils AnsiCompareText Compare two strings for equality, ignoring case
38123: Func StrUtils AnsiContainsStr Returns true if a string contains a substring
38124: Func StrUtils AnsiEndsStr Returns true if a string ends with a substring
38125: Func StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
38126: Func StrUtils AnsiLeftStr Extracts characters from the left of a string
38127: Func SysUtils AnsiLowerCase Change upper case characters in a string to lower case
38128: Func StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
38129: Func StrUtils AnsiMidStr Returns a substring from the middle characters of a string
38130: Func StrUtils AnsiPos Find the position of one string in another
38131: Func StrUtils AnsiReplaceStr Replaces a part of one string with another
38132: Func StrUtils AnsiReverseString Reverses the sequence of letters in a string
38133: Func StrUtils AnsiRightStr Extracts characters from the right of a string
38134: Func StrUtils AnsiStartsStr Returns true if a string starts with a substring
38135: Type System Ansistr A data type that holds a string of AnsiChars
38136: Func SysUtils AnsiUpperCase Change lower case characters in a string to upper case
38137: Proc System Append Open a text file to allow appending of text to the end
38138: Proc SysUtils AppendStr Concatenate one string onto the end of another
38139: Func Math ArcCos The Arc Cosine of a number, returned in radians
38140: Func Math ArcSin The Arc Sine of a number, returned in radians
38141: Func System ArcTan The Arc Tangent of a number, returned in radians
38142: Keyword Array A data type holding indexable collections of data
38143: Keyword As Used for casting object references
38144: Proc System Assign Assigns a file handle to a binary or text file
38145: Func System Assigned Returns true if a reference is not nil
38146: Proc System AssignFile Assigns a file handle to a binary or text file
38147: Proc Printers AssignPrn Treats the printer as a text file - an easy way of printing text
38148:
38149: B
38150: Compiler Directive $B Whether to short cut and and or operations
38151: Compiler Directive $BoolEval Whether to short cut and and or operations
38152: Proc SysUtils Beep Make a beep sound
38153: Keyword Begin Keyword that starts a statement block
38154: Func System BeginThread Begins a separate thread of code execution
38155: Proc System BlockRead Reads a block of data records from an untyped binary file
38156: Proc System BlockWrite Writes a block of data records to an untyped binary file
38157: Type System Boolean Allows just True and False values
38158: Func Classes Bounds Create a TRect value from top left and size values
38159: Proc System Break Forces a jump out of a single loop
38160: Type System Byte An Int type supporting values 0 to 255
38161:
38162: C
38163: Type System Cardinal The basic unsigned Int type
38164: Keyword Case A mechanism for acting upon different values of an Ordinal
38165: Func StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
38166: Func SysUtils ChangeFileExt Change the extension part of a file name
38167: Type System Char Variable type holding a single character
38168: Proc System ChDir Change the working drive plus path for a specified drive
38169: Func System Chr Convert an Int into a character
38170: Keyword Class Starts the declaration of a type of object class
38171: Proc System Close Closes an open file
38172: Proc System CloseFile Closes an open file
38173: Variable System CmdLine Holds the execution text used to start the current program
38174: Type System Comp A 64 bit signed Int
38175: Func SysUtils CompareStr Compare two strings to see which is greater than the other
38176: Func SysUtils CompareText Compare two strings for equality, ignoring case
38177: Func Math CompareValue Compare numeric values with a tolerance
38178: Func System Concat Concatenates one or more strings into one string
38179: Keyword Const Starts the definition of fixed data values
38180: Keyword Constructor Defines the method used to create an object from a class
38181: Proc System Continue Forces a jump to the next iteration of a loop
38182: Func ConvUtils Convert Convert one measurement value to another
38183: Func System Copy Create a copy of part of a string or an array
38184: Func System Cos The Cosine of a number
38185: Func SysUtils CreateDir Create a directory
38186: Type System Currency A floating point type with 4 decimals used for financial values
38187: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
38188: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
38189: Variable SysUtils CurrencyString The currency string used in currency display functions
38190: Func SysUtils CurrToStr Convert a currency value to a string
38191: Func SysUtils CurrToStrF Convert a currency value to a string with formatting
38192:
38193: D
38194: Compiler Directive $D Determines whether application debug information is built
38195: Compiler Directive $DebugInfo Determines whether application debug information is built
38196: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
```

```
38197: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
38198: Func SysUtils Date Gives the current date
38199: Variable SysUtils DateSeparator The character used to separate display date fields
38200: Func SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
38201: Func SysUtils DateTimeToStr Converts TDateTime date and time values to a string
38202: Proc SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
38203: Func SysUtils DateToStr Converts a TDateTime date value to a string
38204: Func DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
38205: Func DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
38206: Func DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
38207: Func SysUtils DayOfWeek Gives day of week index for a TDateTime value
38208: Func DateUtils DaysBetween Gives the whole number of days between 2 dates
38209: Func DateUtils DaysInAMonth Gives the number of days in a month
38210: Func DateUtils DaysInAYear Gives the number of days in a year
38211: Func DateUtils DaySpan Gives the fractional number of days between 2 dates
38212: Proc System Dec Decrement an ordinal variable
38213: Variable SysUtils DecimalSeparator The character used to display the decimal point
38214: Proc SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
38215: Proc DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
38216: Proc SysUtils DecodeTime Break a TDateTime value into individual time values
38217: Directive Default Defines default processing for a property
38218: Func Math DegToRad Convert a degrees value to radians
38219: Proc System Delete Delete a section of characters from a string
38220: Func SysUtils DeleteFile Delete a file specified by its file name
38221: Keyword Destructor Defines the method used to destroy an object
38222: Func SysUtils DirectoryExists Returns true if the given directory exists
38223: Func SysUtils DiskFree Gives the number of free bytes on a specified drive
38224: Func SysUtils DiskSize Gives the size in bytes of a specified drive
38225: Proc System Dispose Dispose of storage used by a pointer type variable
38226: Keyword Div Performs Int division, discarding the remainder
38227: Keyword Do Defines the start of some controlled action
38228: Type System Double A floating point type supporting about 15 digits of precision
38229: Keyword DownTo Prefixes an decremental for loop target value
38230: Func StrUtils DupeString Creates a string containing copies of a substring
38231: Directive Dynamic Allows a class method to be overriden in derived classes
38232:
38233: E
38234: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
38235: Compiler Directive $EndIf Terminates conditional code compilation
38236: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
38237: Keyword Else Starts false section of if, case and try statements
38238: Func SysUtils EncodeDate Build a TDateTime value from year, month and day values
38239: Func DateUtils EncodeDateTime Build a TDateTime value from day and time values
38240: Func SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
38241: Keyword End Keyword that terminates statement blocks
38242: Func DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
38243: Func DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
38244: Proc System EndThread Terminates a thread with an exit code
38245: Func System Eof Returns true if a file opened with Reset is at the end
38246: Func System Eoln Returns true if the current text file is pointing at a line end
38247: Proc System Erase Erase a file
38248: Variable System ErrorAddr Sets the error address when an application terminates
38249: Keyword Except Starts the error trapping clause of a Try statement
38250: Proc System Exclude Exclude a value in a set variable
38251: Proc System Exit Exit abruptly from a Func or procedure
38252: Variable System ExitCode Sets the return code when an application terminates
38253: Func System Exp Gives the exponent of a number
38254: Directive System Export Makes a Func or Proc in a DLL externally available
38255: Type System Extended The floating point type with the highest capacity and precision
38256: Func SysUtils ExtractFileDir Extracts the dir part of a full file name
38257: Func SysUtils ExtractFileDrive Extracts the drive part of a full file name
38258: Func SysUtils ExtractFileExt Extracts the extension part of a full file name
38259: Func SysUtils ExtractFileName Extracts the name part of a full file name
38260: Func SysUtils ExtractFilePath Extracts the path part of a full file name
38261:
38262: F
38263: Func StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
38264: Keyword File Defines a typed or untyped file
38265: Func SysUtils FileAge Get the last modified date/time of a file without opening it
38266: Func SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
38267: Func SysUtils FileExists Returns true if the given file exists
38268: Func SysUtils FileGetAttr Gets the attributes of a file
38269: Variable System FileMode Defines how Reset opens a binary file
38270: Func System FilePos Gives the file position in a binary or text file
38271: Func SysUtils FileSearch Search for a file in one or more directories
38272: Func SysUtils FileSetAttr Sets the attributes of a file
38273: Func SysUtils FileSetDate Set the last modified date and time of a file
38274: Func System FileSize Gives the size in records of an open file
38275: Proc System FillChar Fills out a section of storage with a fill character or byte value
38276: Keyword Finally Starts the unconditional code section of a Try statement
38277: Func SysUtils FindClose Closes a successful FindFirst file search
38278: Func SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
38279: Func SysUtils FindFirst Finds all files matching a file mask and attributes
38280: Func SysUtils FindNext Find the next file after a successful FindFirst
38281: Func SysUtils FloatToStr Convert a floating point value to a string
38282: Func SysUtils FloatToStrF Convert a floating point value to a string with formatting
38283: Proc System Flush Flushes buffered text file data to the file
38284: Keyword For Starts a loop that executes a finite number of times
38285: Func SysUtils ForceDirectories Create a new path of directories
```

```
38286: Func SysUtils Format Rich formatting of numbers and text into a string
38287: Func SysUtils FormatCurr Rich formatting of a currency value into a string
38288: Func SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
38289: Func SysUtils FormatFloat Rich formatting of a floating point number into a string
38290: Func System Frac The fractional part of a floating point number
38291: Proc SysUtils FreeAndNil Free memory for an object and set it to nil
38292: Proc System FreeMem Free memory storage used by a variable
38293: Keyword System Func Defines a subroutine that returns a value
38294:
38295: G
38296: Func SysUtils GetCurrentDir Get the current directory (drive plus directory)
38297: Proc System GetDir Get the default directory (drive plus path) for a specified drive
38298: Func System GetLastError Gives the error code of the last failing Windows API call
38299: Proc SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
38300: Func System GetMem Get a specified number of storage bytes
38301: Keyword Goto Forces a jump to a label, regardless of nesting
38302:
38303: H
38304: Compiler Directive $H Treat string types as Ansistr or ShortString
38305: Compiler Directive $Hints Determines whether Delphi shows compilation hints
38306: Proc System Halt Terminates the program with an optional dialog
38307: Func System Hi Returns the hi-order byte of a (2 byte) Int
38308: Func System High Returns the highest value of a type or variable
38309:
38310: I
38311: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
38312: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
38313: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
38314: Compiler Directive $IfOpt Tests for the state of a Compiler directive
38315: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
38316: Compiler Directive $IOChecks When on, an IO operation error throws an exception
38317: Keyword If Starts a conditional expression to determine what to do next
38318: Keyword Implementation Starts the implementation (code) section of a Unit
38319: Keyword In Used to test if a value is a member of a set
38320: Proc System Inc Increment an ordinal variable
38321: Func DateUtils IncDay Increments a TDateTime variable by + or - number of days
38322: Proc System Include Include a value in a set variable
38323: Func DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
38324: Func DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
38325: Func SysUtils IncMonth Increments a TDateTime variable by a number of months
38326: Func DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
38327: Func DateUtils IncYear Increments a TDateTime variable by a number of years
38328: Directive Index Principally defines indexed class data properties
38329: Constant Math Infinity Floating point value of infinite size
38330: Keyword Inherited Used to call the parent class constructor or destructor method
38331: Variable System Input Defines the standard input text file
38332: Func Dialogs InputBox Display a dialog that asks for user text input, with default
38333: Func Dialogs InputQuery Display a dialog that asks for user text input
38334: Proc System Insert Insert a string into another string
38335: Func System Int The Int part of a floating point number as a float
38336: Type System Int64 A 64 bit sized Int - the largest in Delphi
38337: Type System Int The basic Int type
38338: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
38339: Func SysUtils IntToHex Convert an Int into a hexadecimal string
38340: Func SysUtils IntToStr Convert an Int into a string
38341: Func System IOResult Holds the return code of the last I/O operation
38342: Keyword Is Tests whether an object is a certain class or ascendant
38343: Func Math IsInfinite Checks whether a floating point number is infinite
38344: Func SysUtils IsLeapYear Returns true if a given calendar year is a leap year
38345: Func System IsMultiThread Returns true if the code is running multiple threads
38346: Func Math IsNaN Checks to see if a floating point number holds a real number
38347:
38348: L
38349: Compiler Directive $L Determines what application debug information is built
38350: Compiler Directive $LocalSymbols Determines what application debug information is built
38351: Compiler Directive $LongStrings Treat string types as Ansistr or ShortString
38352: Func SysUtils LastDelimiter Find the last position of selected characters in a string
38353: Func System Length Return the number of elements in an array or string
38354: Func System Ln Gives the natural logarithm of a number
38355: Func System Lo Returns the low-order byte of a (2 byte) Int
38356: Func Math Log10 Gives the log to base 10 of a number
38357: Variable SysUtils LongDateFormat Long version of the date to string format
38358: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
38359: Type System LongInt An Int whose size is guaranteed to be 32 bits
38360: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
38361: Variable SysUtils LongTimeFormat Long version of the time to string format
38362: Type System LongWord A 32 bit unsigned Int
38363: Func System Low Returns the lowest value of a type or variable
38364: Func SysUtils LowerCase Change upper case characters in a string to lower case
38365:
38366: M
38367: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
38368: Func Math Max Gives the maximum of two Int values
38369: Constant System MaxInt The maximum value an Int can have
38370: Constant System MaxLongInt The maximum value an LongInt can have
38371: Func Math Mean Gives the average for a set of numbers
38372: Func Dialogs MessageDlg Displays a message, symbol, and selectable buttons
38373: Func Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
38374: Func Math Min Gives the minimum of two Int values
```

```
38375: Constant SysUtils MinsPerDay Gives the number of minutes in a day
38376: Proc System MkDir Make a directory
38377: Keyword Mod Performs Int division, returning the remainder
38378: Constant SysUtils MonthDays Gives the number of days in a month
38379: Func DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
38380: Proc System Move Copy bytes of data from a source to a destination
38381:
38382: N
38383: Constant Math NaN Not a real number
38384: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
38385: Proc System New Create a new pointer type variable
38386: Constant System Nil A pointer value that is defined as undetermined
38387: Keyword Not Boolean Not or bitwise not of one arguments
38388: Func SysUtils Now Gives the current date and time
38389: Variable Variants Null A variable that has no value
38390:
38391: O
38392: Compiler Directive $O Determines whether Delphi optimises code when compiling
38393: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
38394: Compiler Directive $OverFlowChecks Determines whether Delphi checks Int and enum bounds
38395: Keyword System Object Allows a subroutine data type to refer to an object method
38396: Func System Odd Tests whether an Int has an odd value
38397: Keyword Of Linking keyword used in many places
38398: Keyword On Defines exception handling in a Try Except clause
38399: Keyword Or Boolean or or bitwise or of two arguments
38400: Func System Ord Provides the Ordinal value of an Int, character or enum
38401: Directive Out Identifies a routine parameter for output only
38402: Variable System Output Defines the standard output text file
38403: Directive Overload Allows 2 or more routines to have the same name
38404: Directive Override Defines a method that replaces a virtual parent class method
38405:
38406: P
38407: Keyword Packed Compacts complex data types into minimal storage
38408: Type System PAnsiChar A pointer to an AnsiChar value
38409: Type System PAnsistr Pointer to an Ansistr value
38410: Func System ParamCount Gives the number of parameters passed to the current program
38411: Func System ParamStr Returns one of the parameters used to run the current program
38412: Type System PChar A pointer to an Char value
38413: Type System PCurrency Pointer to a Currency value
38414: Type System PDateTime Pointer to a TDateTime value
38415: Type System PExtended Pointer to a Extended floating point value
38416: Func System Pi The mathematical constant
38417: Type System PInt64 Pointer to an Int64 value
38418: Func Classes Point Generates a TPoint value from X and Y values
38419: Type System Pointer Defines a general use Pointer to any memory based data
38420: Func Classes PointsEqual Compares two TPoint values for equality
38421: Func System Pos Find the position of one string in another
38422: Func System Pred Decrement an ordinal variable
38423: Func Printers Printer Returns a reference to the global Printer object
38424: Directive Private Starts the section of private data and methods in a class
38425: Keyword System Proc Defines a subroutine that does not return a value
38426: Proc FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
38427: Keyword System Program Defines the start of an application
38428: Func Dialogs PromptForFileName Shows a dialog allowing the user to select a file
38429: Keyword System Property Defines controlled access to class fields
38430: Directive Protected Starts a section of class private data accesible to sub-classes
38431: Type System PShortString A pointer to an ShortString value
38432: Type System PString Pointer to a String value
38433: Func Types PtInRect Tests to see if a point lies within a rectangle
38434: Directive Public Starts an externally accessible section of a class
38435: Directive Published Starts a published externally accessible section of a class
38436: Type System PVariant Pointer to a Variant value
38437: Type System PWideChar Pointer to a WideChar
38438: Type System PWideString Pointer to a WideString value
38439:
38440: Q
38441: Compiler Directive $Q Determines whether Delphi checks Int and enum bounds
38442:
38443: R
38444: Compiler Directive $R Determines whether Delphi checks array bounds
38445: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
38446: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
38447: Compiler Directive $Resource Defines a resource file to be included in the application linking
38448: Func Math RadToDeg Converts a radian value to degrees
38449: Keyword Raise Raise an exception
38450: Func System Random Generate a random floating point or Int number
38451: Proc System Randomize Reposition the Random number generator next value
38452: Func Math RandomRange Generate a random Int number within a supplied range
38453: Variable System RandSeed Reposition the Random number generator next value
38454: Proc System Read Read data from a binary or text file
38455: Proc System ReadLn Read a complete line of data from a text file
38456: Type System Real A floating point type supporting about 15 digits of precision
38457: Type System Real48 The floating point type with the highest capacity and precision
38458: Proc System ReallocMem Reallocate an existing block of storage
38459: Func DateUtils RecodeDate Change only the date part of a TDateTime variable
38460: Func DateUtils RecodeTime Change only the time part of a TDateTime variable
38461: Keyword Record A structured data type - holding fields of data
38462: Func Classes Rect Create a TRect value from 2 points or 4 coordinates
38463: Func SysUtils RemoveDir Remove a directory
```

```
38464: Proc System Rename Rename a file
38465: Func SysUtils RenameFile Rename a file or directory
38466: Keyword Repeat Repeat statements until a ternmination condition is met
38467: Proc SysUtils ReplaceDate Change only the date part of a TDateTime variable
38468: Proc SysUtils ReplaceTime Change only the time part of a TDateTime variable
38469: Proc System Reset Open a text file for reading, or binary file for read/write
38470: Variable System Result A variable used to hold the return value from a function
38471: Proc System ReWrite Open a text or binary file for write access
38472: Proc System RmDir Remove a directory
38473: Func System Round Rounds a floating point number to an Int
38474: Proc System RunError Terminates the program with an error dialog
38475:
38476: S
38477: Constant SysUtils SecsPerDay Gives the number of seconds in a day
38478: Proc System Seek Move the pointer in a binary file to a new record position
38479: Func System SeekEof Skip to the end of the current line or file
38480: Func System SeekEoln Skip to the end of the current line or file
38481: Func FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
38482: Variable System Self Hidden parameter to a method - refers to the containing object
38483: Keyword Set Defines a set of up to 255 distinct values
38484: Func SysUtils SetCurrentDir Change the current directory
38485: Proc System SetLength Changes the size of a string, or the size(s) of an array
38486: Proc System SetString Copies characters from a buffer into a string
38487: Keyword Shl Shift an Int value left by a number of bits
38488: Variable SysUtils ShortDateFormat Compact version of the date to string format
38489: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
38490: Type System ShortInt An Int type supporting values -128 to 127
38491: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
38492: Type System ShortString Defines a string of up to 255 characters
38493: Variable SysUtils ShortTimeFormat Short version of the time to string format
38494: Proc Dialogs ShowMessage Display a string in a simple dialog with an OK button
38495: Proc Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
38496: Proc Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
38497: Keyword Shr Shift an Int value right by a number of bits
38498: Func System Sin The Sine of a number
38499: Type System Single The smallest capacity and precision floating point type
38500: Func System SizeOf Gives the storage byte size of a type or variable
38501: Func System Slice Creates a slice of an array as an Open Array parameter
38502: Type System SmallInt An Int type supporting values from -32768 to 32767
38503: Func System Sqr Gives the square of a number
38504: Func System Sqrt Gives the square root of a number
38505: Proc System Str Converts an Int or floating point number to a string
38506: Type System String A data type that holds a string of characters
38507: Func System StringOfChar Creates a string with one character repeated many times
38508: Func SysUtils StringReplace Replace one or more substrings found within a string
38509: Func System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
38510: Func SysUtils StrScan Searches for a specific character in a constant string
38511: Func SysUtils StrToCurr Convert a number string into a currency value
38512: Func SysUtils StrToDate Converts a date string into a TDateTime value
38513: Func SysUtils StrToDateTime Converts a date+time string into a TDateTime value
38514: Func SysUtils StrToFloat Convert a number string into a floating point value
38515: Func SysUtils StrToInt Convert an Int string into an Int value
38516: Func SysUtils StrToInt64 Convert an Int string into an Int64 value
38517: Func SysUtils StrToInt64Def Convert a string into an Int64 value with default
38518: Func SysUtils StrToIntDef Convert a string into an Int value with default
38519: Func SysUtils StrToTime Converts a time string into a TDateTime value
38520: Func StrUtils StuffString Replaces a part of one string with another
38521: Func System Succ Increment an ordinal variable
38522: Func Math Sum Return the sum of an array of floating point values
38523:
38524: T
38525: Func Math Tan The Tangent of a number
38526: Type Classes TBits An object that can hold an infinite number of Boolean values
38527: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
38528: Type ConvUtils TConvType Defines a measurement type as used by Convert
38529: Type System TDateTime Data type holding a date and time value
38530: Type System Text Defines a file as a text file
38531: Type System TextFile Declares a file type for storing lines of text
38532: Type SysUtils TFloatFormat Formats for use in floating point number display functions
38533: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
38534: Keyword Then Part of an if statement - starts the true clause
38535: Variable SysUtils ThousandSeparator The character used to display the thousands separator
38536: Keyword ThreadVar Defines variables that are given separate instances per thread
38537: Func SysUtils Time Gives the current time
38538: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
38539: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
38540: Variable SysUtils TimeSeparator The character used to separate display time fields
38541: Func SysUtils TimeToStr Converts a TDateTime time value to a string
38542: Type Classes TList General purpose container of a list of objects
38543: Keyword To Prefixes an incremental for loop target value
38544: Type System TObject The base class type that is ancestor to all other classes
38545: Func DateUtils Tomorrow Gives the date tomorrow
38546: Type Dialogs TOpenDialog Displays a file selection dialog
38547: Type Types TPoint Holds X and Y Int values
38548: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
38549: Type Types TRect Holds rectangle coordinate values
38550: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
38551: Func SysUtils Trim Removes leading and trailing blanks from a string
38552: Func SysUtils TrimLeft Removes leading blanks from a string
```

```
38553: Func SysUtils TrimRight Removes trailing blanks from a string
38554: Func System Trunc The Int part of a floating point number
38555: Proc System Truncate Truncates a file size - removes all data after the current position
38556: Keyword Try Starts code that has error trapping
38557: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
38558: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
38559: Type Classes TStringList Holds a variable length list of strings
38560: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
38561: Type System TThreadFunc Defines the Func to be called by BeginThread
38562: Var SysUtils TwoDigitYearCenturyWindow Sets century threshold for 2 digit year string conversions
38563: Keyword Type Defines a new category of variable or process
38564:
38565: U
38566: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
38567: Keyword Unit Defines the start of a unit file - a Delphi module
38568: Keyword Until Ends a Repeat control loop
38569: Func System UpCase Convert a Char value to upper case
38570: Func SysUtils UpperCase Change lower case characters in a string to upper case
38571: Keyword Uses Declares a list of Units to be imported
38572:
38573: V
38574: Proc System Val Converts number strings to Int and floating point values
38575: Keyword Var Starts the definition of a section of data variables
38576: Type System Variant A variable type that can hold changing data types
38577: Func Variants VarType Gives the current type of a Variant variable
38578: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
38579: Directive Virtual Allows a class method to be overriden in derived classes
38580:
38581: W
38582: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
38583: Keyword While Repeat statements whilst a continuation condition is met
38584: Type System WideChar Variable type holding a single International character
38585: Func System WideCharToString Copies a null terminated WideChar string to a normal string
38586: Type System WideString A data type that holds a string of WideChars
38587: Keyword With A means of simplifying references to structured variables
38588: Type System Word An Int type supporting values 0 to 65535
38589: Func SysUtils WrapText Add line feeds into a string to simulate word wrap
38590: Proc System Write Write data to a binary or text file
38591: Proc System WriteLn Write a complete line of data to a text file
38592:
38593: X
38594: Compiler Directive $X Controls some Pascal extension handling
38595: Keyword Xor Boolean Xor or bitwise Xor of two arguments
38596: Y
38597: Compiler Directive $Y Determines whether application symbol information is built
38598: Func DateUtils Yesterday Gives the date yesterday
38599: Z
38600: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
38601:
38602:
38603: Proc SIRegister_uPSUtils(CL: TPSPascalCompiler);
38604: begin     //'TPSBaseType','').SetString( Byte);
38605:  PSMainProcName','String').SetString( '!MAIN;
38606:  PSMainProcNameOrg','String').SetString( 'Main Proc;
38607: 'PSLowBuildSupport','LongInt').SetInt(12);
38608: 'PSCurrentBuildNo','LongInt').SetInt(23);
38609: 'PSCurrentversion','String').SetString('1.31;
38610: 'PSValidHeader','LongInt').SetInt(1397769801);
38611: 'PSAddrStackStart','LongInt').SetInt(1610612736);
38612: 'PSAddrNegativeStackStart','LongInt').SetInt(1073741824);
38613: 'btReturnAddress','LongInt').SetInt(0);
38614: 'btU8','LongInt').SetInt(1);
38615: 'btS8','LongInt').SetInt(2);
38616: 'btU16','LongInt').SetInt(3);
38617: 'btS16','LongInt').SetInt(4);
38618: 'btU32','LongInt').SetInt(5);
38619: 'btS32','LongInt').SetInt(6);
38620: 'btSingle','LongInt').SetInt(7);
38621: 'btDouble','LongInt').SetInt(8);
38622: 'btExtended','LongInt').SetInt(9);
38623: 'btString','LongInt').SetInt(10);
38624: 'btRecord','LongInt').SetInt(11);
38625: 'btArray','LongInt').SetInt(12);
38626: 'btPointer','LongInt').SetInt(13);
38627: 'btPChar','LongInt').SetInt(14);
38628: 'btResourcePointer','LongInt').SetInt(15);
38629: 'btVariant','LongInt').SetInt(16);
38630: 'btS64','LongInt').SetInt(17);
38631: 'btU64','LongInt').SetInt(30);
38632: 'btChar','LongInt').SetInt(18);
38633: 'btWideString','LongInt').SetInt( 19);
38634: 'btWideChar','LongInt').SetInt( 20);
38635: 'btProcPtr','LongInt').SetInt( 21);
38636: 'btStaticArray','LongInt').SetInt( 22);
38637: 'btSet','LongInt').SetInt( 23);
38638: 'btCurrency','LongInt').SetInt( 24);
38639: 'btClass','LongInt').SetInt( 25);
38640: 'btInterface','LongInt').SetInt( 26);
38641: 'btNotificationVariant','LongInt').SetInt( 27);
```

```
38642: 'btUnicodeString','LongInt').SetInt( 28);
38643: 'btType','LongInt').SetInt( 130);
38644: 'btEnum','LongInt').SetInt( 129);
38645: 'btExtClass','LongInt').SetInt( 131);
38646: 'CM_A','LongInt').SetInt( 0);
38647: 'CM_CA','LongInt').SetInt( 1);
38648: 'CM_P','LongInt').SetInt( 2);
38649: 'CM_PV','LongInt').SetInt( 3);
38650: 'CM_PO','LongInt').SetInt( 4);
38651: 'Cm_C','LongInt').SetInt( 5);
38652: 'Cm_G','LongInt').SetInt( 6);
38653: 'Cm_CG','LongInt').SetInt( 7);
38654: 'Cm_CNG','LongInt').SetInt( 8);
38655: 'Cm_R','LongInt').SetInt( 9);
38656: 'Cm_ST','LongInt').SetInt( 10);
38657: 'Cm_Pt','LongInt').SetInt( 11);
38658: 'CM_CO','LongInt').SetInt( 12);
38659: 'Cm_cv','LongInt').SetInt( 13);
38660: 'cm_sp','LongInt').SetInt( 14);
38661: 'cm_bn','LongInt').SetInt( 15);
38662: 'cm_vm','LongInt').SetInt( 16);
38663: 'cm_sf','LongInt').SetInt( 17);
38664: 'cm_fg','LongInt').SetInt( 18);
38665: 'cm_puexh','LongInt').SetInt( 19);
38666: 'cm_poexh','LongInt').SetInt( 20);
38667: 'cm_in','LongInt').SetInt( 21);
38668: 'cm_spc','LongInt').SetInt( 22);
38669: 'cm_inc','LongInt').SetInt( 23);
38670: 'cm_dec','LongInt').SetInt( 24);
38671: 'cm_nop','LongInt').SetInt( 255);
38672: 'Cm_PG','LongInt').SetInt( 25);
38673: 'Cm_P2G','LongInt').SetInt( 26);
38674: TypeS('TbtU8', 'Byte;
38675: TypeS('TbtS8', 'ShortInt;
38676: TypeS('TbtU16', 'Word;
38677: TypeS('TbtS16', 'SmallInt;
38678: TypeS('TbtU32', 'Cardinal;
38679: TypeS('TbtS32', 'Longint;
38680: TypeS('TbtSingle', 'Single;
38681: TypeS('TbtDouble', 'double;
38682: TypeS('TbtExtended', 'Extended;
38683: TypeS('tbtCurrency', 'Currency;
38684: TypeS('tbts64', 'int64;
38685: TypeS('Tbtu64', 'uint64;
38686: TypeS('TbtString', 'string;
38687: Func MakeHash( const s : TbtString) : Longint;
38688: // TbtString = {$IFDEF DELPHI2009UP}Ansistr{$ELSE}String{$ENDIF};
38689: //'PointerSize','LongInt').SetInt( IPointer ( 8 4 ));
38690: end:
38691:
38692: ------------------------------------------------------------------------------
38693: mapX:
38694: if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
38695:  writeln('cologne map found;
38696:  GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
38697:  writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
38698:  OpenMapX('church trier;
38699:  GetGeoCode(C_form,apath:Str; const data:Str; sfile:Bool):Str;
38700:  writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));}
38701:  >>>  //latitude: '50.94133705'  longitude: '6.95812076100766'
38702:  // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
38703:  writeln(GetGeoCoord('xml','church cefalu sicily',true))
38704:  CoordinateStr(Idx: Int; PosInSec: Double; PosLn: TPos):Str;
38705:  Func SendInput( cInputs : UINT; var pInputs : TInput; cbSize : Int) : UINT;
38706:  Func GetLastInputInfo( var plii : TLastInputInfo) : BOOL;
38707:  Proc JvErrorIntercept;
38708:  writeln(GetGeoInfo4('178.196.192.131', UrlGeoLookupInfo3));
38709:
38710:    {$IFDEF MSWINDOWS}
38711:    Proc Process;
38712:    var
38713:      Msg: TMsg;
38714:    begin
38715:      if ApplicationHasPriority then begin
38716:        Application.ProcessMessages;
38717:      end else begin
38718:        // This guarantees it will not ever call Application.Idle
38719:        if PeekMessage(Msg, 0, 0, 0, PM_NOREMOVE) then begin
38720:          Application.HandleMessage;
38721:        end;
38722:      end;
38723:    end;
38724:
38725:
38726: Ref:
38727:  1. writeln(SHA1(Exepath+'\maxbox4.exe'))
38728:  2. shdig: TSHA1Digest;
38729:     shdig:= GetSHA1OfFile(false,exepath+'\maxbox4.exe;
38730:             for i:= 0 to 19 do write(BytetoHex(shdig[i]));
```

```
38731:
38732:  3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox4.exe'),4));
38733:  4. writeln(sha1tohex(SHA1ofStr(loadStringofFile(exepath+'maxbox4.exe'))))
38734:
38735: I have to get only the numbers, without the hyphen and the dots, to my calcs worth.
38736:     sr := '123.456.789-00'
38737:     writeln(ReplaceRegExpr('\D',sr,'',true))
38738:
38739: Func GetLocalStream(C_form:Str; const lat,long:Str):boolean;
38740:  var encodedURL, UrlMapQuestAPI, bufstr:Str;
38741:      mStream: TMemoryStream;
38742:      idhttp: TIdHttp;// THTTPSend;
38743:      len: Int;
38744:  begin
38745:    UrlMapQuestAPI:= 'http://192.168.1.53:9000';
38746:    encodedURL:= UrlMapQuestAPI;
38747:    idHTTP:= TIdHTTP.Create(NIL)
38748:    mStream:= TMemoryStream.create;
38749:    try
38750:      //HttpGet(EncodedURL, mStream);  {WinInet}
38751:      idHTTP.Get1(EncodedURL, mStream)
38752:      mStream.Position:= 0;
38753:      writeln('stream size: '+inttostr(mStream.size));
38754:      //mStream.memory;
38755:      len:= mStream.Size - mStream.Position;
38756:      SetLength(bufstr, len);
38757:      mStream.readbuffer(bufstr, len)
38758:      writeln('debug stream local back: '+bufstr)
38759:    finally
38760:      encodedURL:= '';
38761:      mStream.Free;
38762:      idHTTP.Disconnect
38763:      idHTTP.Free;
38764:      result:= true;
38765:    end;
38766:  end;
38767:
38768: 39/40 PE imports of V maXbox 4.2.5.10
38769:
38770: 1[+] AVICAP32.DLL  2[+] AVICAP32.dll   3[+] GLU32.dll     40[++]dmath.dll
38771: 4[+] IMAGEHLP.DLL  6[+] MSVCRT.DLL     7[+] MSVFW32.DLL
38772: 8[+] OpenGL32.dll  9[+] SHFolder.dll  11[+] URLMON.DLL    10[+] SHfolder.dll
38773: 12[+] advapi32.dll 13[+] comctl32.dll 14[+] comdlg32.dll
38774: 15[+] gdi32.dll    16[+] imagehlp.dll 17[+] imm32.dll
38775: 18[+] iphlpapi.dll 19[+] kernel32.dll 20[+] mpr.dll
38776: 21[+] msacm32.dll  24[+] ole32.dll    25[+] oleacc.dll
38777: 26[+] oleaut32.dll 27[+] oledlg.dll   28[+] opengl32.dll
38778: 29[+] shell32.dll  30[+] shlwapi.dll  31[+] user32.dll
38779: 32[+] usp10.dll    33[+] version.dll  34[+] winhttp.dll
38780: 35[+] wininet.dll  36[+] winmm.dll    37[+] winspool.drv   5[+] KERNEL32.DLL
38781: 38[+] ws2_32.dll   39[+] wsock32.dll  22[+] msimg32.dll   23[+] netapi32.dll
38782:
38783: Magic literal PE32 executable for MS Windows (GUI) Intel 80386 32-bit
38784: ******************************************************************************
38785: Microsoft Windows SDK for Windows 7 and .NET Framework 4 Interface (UI Automation)
38786: Overall test results Number of tests 68 Verify
38787: Passed 67 Failed 1
38788: Unexpected Error 0
38789:
38790: UI Tests caused Unexpected Error: 0
38791: Tests Failed: 1
38792: Test Name : ControlType.Window:SetFocus.1
38793: Result Status: Failed
38794: Test Priority: Pri1
38795: Description: Precondition: Some controls are not persisted after they loose focus (ie, Floating edit in
       Excel), don't tests these
38796: !! Exception !!
38797: Message: Step 12 : Focus event was not fired and was expected to be fired
38798: Unexpected: false Known Bug: false Incorrect Configuration: false
38799:
38800: 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)
38801:
38802: Tests Passed:67/68
38803: Test Name:ControlType.Window:FocusedElement.1
38804: Test Name:ControlType.Window:GetRuntimeIdType.1
38805: Test Name:ControlType.Window:GetSupportedPatterns.1
38806: Test Name:ControlType.Window:GetSupportedProperties.1
38807: Test Name:ControlType.Window:GetType.1
38808: Test Name:ControlType.Window:HwndWindowRect.1.MSAA
38809: Test Name:ControlType.Window:HwndWindowRect.2.UIA
38810: Test Name:ControlType.Window:LocalizedControlType.1.8.1
```

```
38811: Test Name:ControlType.Window:ToString.1
38812: Test Name:ControlType.Window:AutomationIdProperty.1.7.1
38813: Test Name:ControlType.Window:AutomationIdProperty.1.7.8
38814: Test Name:ControlType.Window:BoundingRect.1
38815: Test Name:ControlType.Window:ClickablePointProperty.1
38816: Test Name:ControlType.Window:ControlTypeProperty.1
38817: Test Name:ControlType.Window:GetClickablePoint.2
38818: Test Name:ControlType.Window:GetHashCode.1
38819: Test Name:ControlType.Window:IsKeyboardFocusable.1.1.1
38820: Test Name:ControlType.Window:IsPasswordProperty.1.3.2
38821: Test Name:ControlType.Window:KeyboardFocusable.1.1.2
38822: Test Name:ControlType.Window:Navigation.1
38823: Test Name:ControlType.Window:Navigation.10
38824: Test Name:ControlType.Window:Navigation.11
38825: Test Name:ControlType.Window:Navigation.2
38826: Test Name:ControlType.Window:Navigation.3
38827: Test Name:ControlType.Window:Navigation.4
38828: Test Name:ControlType.Window:Navigation.5
38829: Test Name:ControlType.Window:Navigation.6
38830: Test Name:ControlType.Window:Navigation.7
38831: Test Name:ControlType.Window:Navigation.8
38832: Test Name:ControlType.Window:Navigation.9
38833: Test Name:ControlType.Window:Navigation.ControlFirstChild.1
38834: Test Name:ControlType.Window:Navigation.ControlLastChild.1
38835: Test Name:ControlType.Window:Navigation.ControlNextSibling.1
38836: Test Name:ControlType.Window:Navigation.ControlParent.1
38837: Test Name:ControlType.Window:Navigation.ControlPreviousSibling.1
38838: Test Name:ControlType.Window:SetFocus.10
38839: Test Name:ControlType.Window:SetFocus.3
38840: Test Name:ControlType.Window:AutomationElement.PropertyChange.Enabled.1
38841: Test Name:ControlType.Window:AutomationElement.PropertyChange.Name.1
38842: Test Name:ControlType.Window:GetCurrentPattern.1
38843: Test Name:ControlType.Window:GetCurrentPropertyValue.1
38844: Test Name:ControlType.Window:SetFocus.4
38845: Test Name:ControlType.Window:BulkAdd.1
38846: Test Name:ControlType.Window:BulkRemove.1
38847: Test Name:ControlType.Window:ChildAdd.1
38848: Test Name:ControlType.Window:ChildRemove.1
38849: Test Name:ControlType.Window:Invalidate.1
38850: Test Name:ControlType.Window:Reorder.1
38851: Test Name:ControlType.Window:TestContentView
38852: Test Name:ControlType.Window:TestControlPatterns
38853: Test Name:ControlType.Window:TestControlProperties
38854: Test Name:ControlType.Window:TestControlView
38855: Test Name:ControlType.Window:CanRotate.1.6
38856: Test Name:ControlType.Window:Move.2.1
38857: Test Name:ControlType.Window:Move.2.2
38858: Test Name:ControlType.Window:Move.2.8
38859: Test Name:ControlType.Window:Move.2.13
38860: Test Name:ControlType.Window:Move.2.17
38861: Test Name:ControlType.Window:Move.2.18
38862: Test Name:ControlType.Window:Move.2.19
38863: Test Name:ControlType.Window:Move.2.3
38864: Test Name:ControlType.Window:Move.2.4
38865: Test Name:ControlType.Window:Move.2.5
38866: Test Name:ControlType.Window:Window.MaximizableProperty.S.6.1
38867: Test Name:ControlType.Window:Window.MaximizableProperty.S.6.2
38868: Test Name:ControlType.Window:Window.MinimizableProperty.S.7.2
38869: Test Name:ControlType.Window:Window.ModalProperty.S.10.1
38870:
38871:  Func GetCachedFileFromURL(strUL:Str; var strLocalFile:Str):Bool;
38872:  Func IAddrToHostName(const IP:Str):Str;
38873:  Func GetIEHandle(WebBrowser: TWebbrowser; ClassName:Str): HWND;
38874:  Func GetTextFromHandle(WinHandle: THandle):Str;
38875:  Proc Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
38876:  Func FillWebForm(WebBrowser:TWebBrowser;FieldName:str;Value:str):Bool;
38877:  Proc WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode:Str);
38878:  Func NetSend(dest, Source, Msg:Str): Longint; overload;
38879:  Func RecordsetFromXML2(const XML:Str): variant;;
38880:  Func RecordsetToXML2(const Recordset: variant):Str;;
38881:  Func GetCharEncoding( alias :Str; var _name :Str) : Int;
38882:  Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
38883:  Func MicrosoftLangageCodeToISOCode( langcode : Int) :Str;
38884:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str = '';
38885:  Func RFC1123ToDateTime(Date:Str): TDateTime;
38886:  Func DateTimeToRFC1123(aDate: TDateTime):Str;
38887:  Proc CopyHTMLToClipBoard(const str:Str; const htmlStr:Str);;
38888:  Proc DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);;
38889:  Proc DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);;
38890:  Proc DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);;
38891:  Func checkSystem:Str;
38892:  Func getSystemReport:Str;
38893:
38894: //Commonly used Delphi WinAPI routines
38895: http://www.rosseeld.be/DRO/Delphi/Delphi%20WinAPI.htm
38896:
38897: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
38898:
38899: http://www.slideshare.net/maxkleiner1/codesign-2015
```

```
38900: http://basta_2015_speaker_max_kleiner
38901: http://www.arduino.cc/en/Tutorial/ASCIITable
38902: http://www.vwlowen.co.uk/arduino/usb-digital/pc-control.htm
38903: http://www.yunqa.de/delphi/doku.php/products/regex/syntax#quantifiers
38904: http://elib.uni-stuttgart.de/opus/volltexte/2008/3440/pdf/diss_kroell_hs.pdf
38905: http://www.softwareschule.ch/download/exampleedition2016.zip
38906:
38907: Published Doc maXbox Example Edition 2015/2016/2017/2018
38908:
38909:   \example_edition\01_Algorithm;
38910:   \example_edition\02_Graphics;
38911:   \example_edition\03_Games;
38912:   \example_edition\04_Multimedia;
38913:   \example_edition\05_Internet;
38914:   \example_edition\06_Communication;
38915:   \example_edition\07_Geographical;
38916:   \example_edition\08_Operating;
38917:   \example_edition\09_Database;
38918:   \example_edition\10_Science;
38919:   \example_edition\11_Embedded;
38920:   \example_edition\12_Security;
38921:   \example_edition\13_General;
38922:   \example_edition\14_Energy;
38923:   \example_edition\15_Transport;
38924:   \example_edition\16_Robotics;
38925:
38926: Ref:
38927:   if IsValidPeFile(exepath+'maxbox3.exe') then begin
38928:     x1:= ComputePEChecksum(exepath+'maxbox3.exe; // original filename
38929:     x2:= ComputePEChecksum(exepath+'maxbox3.exe;
38930:   end;
38931:   WriteLn('Checksum 1: '+ itoa(x1)+ #13#10+'Checksum 2: '+ itoa(x2));
38932:
38933:   if ConnectDrive('Z:','\\MAXBOX8\Users\Public', True,True) = NO_ERROR then
38934:           writeln('Net Share Z:\ Connected;
38935:      DisconnectNetDrive('Z:', True, True, True);
38936:
38937:     ComTerminal:= TCustomComTerminal.create(self);
38938:       EditComTerminal(comterminal);  //TComTrmSetForm  comterminal.Free;
38939:     LastSysErrorMessage :Str; LastSysErrorMessageA : Ansistr;
38940:
38941:   writeln(sha1tohex(SHA1ofStr(loadstringj(exepath+'maxbox4.exe'))))
38942:   writeln(sha1tohex(SHA1ofStr(loadstringoffile(exepath+'maxbox4.exe'))))
38943:   writeln(strtoHex1(SHA1ofStr(lsof(exepath+'maxbox4.exe'))))
38944:   writeln(sha1tohex(SHA1ofStr(filetoString(exepath+'maxbox4.exe'))))
38945:
38946:   SHA512 sr:= filetostring(Exepath+'maXbox4.exe')
38947:   writeln(uppercase(SHA512DigesttoHEX(CalcSHA512(sr))))
38948:   writeln(Base64EncodeStr((SHA512DigestAsString(CalcSHA512(sr)))));
38949: base64:w4ueRtnUZ641dsy4yIISYvhMB9r0c9kc8YO9o5FgkWQRbayHLaZay7tjnt1JmhV3vck2a7Mec7wfK8PbHacrbQ==
38950:   PE Checksum 1: 27767976 PE Checksum 2: 27767976
38951:
38952: // REST API Example
38953: procedure TOnlineRandomizerGetResultsFromWeb(qUrl: String);
38954: var FResultStream: TMemoryStream;
38955:     Flist: TStringList;
38956:     FResultList: TStringList;
38957:     FSettings:TFormatSettings;
38958: begin
38959:   FResultList:= TSTringList.Create;
38960:   //FSettings:= TFormatSettings.Create;
38961:   //FSettings.DecimalSeparator := '.';
38962:   FResultStream:= TMemoryStream.Create;
38963:   FList:= TStringList.Create;
38964:   //httpObj.Get1(qUrl,FResultStream);
38965:   HTTPget(qURL, FResultStream)
38966:   FResultStream.Position:= 0;
38967:   FList.LoadFromStream(FResultStream);
38968:   FResultStream.Free;
38969:   FResultList.AddStrings(FList);
38970:   writeln('online random: '+flots(StrToFloat(FResultList[3])));
38971:   writeln('online random: '+flots(StrToFloat(FResultList[4])));
38972:   writeln('online random: '+flots(StrToFloat(FResultList[5])));
38973:   FResultList.Free;
38974:   FList.Free;
38975: end;
38976:
38977: //var FDecUrlPattern:String;
38978: procedure TOnlineRandomizerGetRandomDecimals(dPrecission, dCount: Integer);
38979: var qUrl, FDecUrlPattern: String;
38980: begin
38981:   FDecUrlPattern:= 'https://www.random.org/decimal-fractions/'+
38982:                    '?dec=%d&num=%d&col=1&format=plain&rnd=new';
38983:   qUrl:= Format(FDecUrlPattern,[dPrecission, dCount]);
38984:   TOnlineRandomizerGetResultsFromWeb(qUrl);
38985: end;
38986:
38987: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
38988: ------------------------------------------------------------------------------
```

```
38989: ////////////////////////////////////////////////////////////////////////////////
38990: --------------------------------------------------------------------------------
38991: --------------------------------------------------------------------------------
38992: https://www.metadefender.com/#!/results/file/933baa9823514320806c9533482a3b89/regular/analysis
38993: https://www.virustotal.
       com/en/file/b80b0bfef22c6b4be3dbc4af984ca897144895f3c1e162f3ad7895d14fb4e667/analysis/1476479319/
38994: Name Virtual address Virtual size Raw size Entropy MD5
38995: .text 4096 21923324 21923328 6.61 0449d5754ddb069630808bfc6bb356ef
38996: .itext 21929984 52108 52224 6.64 0226e6c46fcf9cc23805e75784f48b4b
38997: .data 21983232 407520 407552 5.25 91d5dac4714aa4119ea18fb5c4ed250d
38998: .bss 22392832 390424 0 0.00 d41d8cd98f00b204e9800998ecf8427e
38999: .idata 22786048 59722 59904 5.53 40cc92b8f087c512630bf807d4d20fe1
39000: .edata 22847488 77 512 0.89 0fcb1fd6f1dde231aa401cffaaa8cd63
39001: .tls 22851584 484 0 0.00 d41d8cd98f00b204e9800998ecf8427e
39002: .rdata 22855680 24 512 0.28 916275c8032d420cb22a05c2086b7e93
39003: .reloc 22859776 1378116 1378304 6.76 51ab0f1c39aa5fc52a7bc82557df173e
39004: .rsrc 24240128 4132864 4132864 5.37 c49d04317b7a6aadc424538ff5ede5c0
39005: CodeSize 21975552 InitializedDataSize 5979648
39006: TimeStamp 2019-08-06 08:06:39
39007: EntryPoint 22518720
39008: OriginalFileName maxbox4_6.exe
39009: MIMEType application/octet-stream
39010:
39011: Article concerning maXbox and mapbox at PHP Magazine 1.2017
39012: https://entwickler.de/php-magazin/php-magazin-1-17-301136.html
39013: WineHQ Update:
39014: https://appdb.winehq.org/objectManager.php?sClass=application&iId=17937
39015: http://www.slideshare.net/maxkleiner1/maxbox-starter46-work-with-wine
39016: https://www.academia.edu/30401550/maXbox_Starter_46_Work_with_WineHQ
39017: https://appdb.winehq.org/appimage.php?iId=50376
39018: https://www.scribd.com/document/333983581/maXbox-starter46-work-with-WineHQ
39019: http://www.softwareschule.ch/download/maxbox_starter46.pdf
39020:
39021: Update for Windows 10 Version 1511 for x64-based Systems (KB3152599)
39022: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3200970)
39023: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3201845)
39024: Update for Windows 10 Version 1607 for x64-based Systems (KB3211320)
39025: Intel Corporation - Display - 11/10/2016 12:00:00 AM - 20.19.15.4549
39026: 2017-10 Cumulative Update for Win 10 Version 1703 for x64-based Systems (KB4041676)
39027: 2017-12 Cumulative Update for Win 10 Version 1703 for x64-based Systems (KB4053580)
39028: Feature update to Windows 10, version 1709
39029: Feature update to Windows 10, version 1803 amd64 2019-04
39030: Feature update to Windows 10, version 1903
39031: Feature update to Windows 10, version 1909
39032: November 12, 2019—KB4524570 (OS Builds 18362.476 and 18363.476)
39033: Version: 1903-OS Build 18362.535 and 1909-OS Build 18363.535
39034: Done Adding Additional Store Successfully signed: maXbox4.exe
39035: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maXbox4.exe
39036: Verifying: maXbox4.exe
39037: Successfully verified: maXbox4.exe
39038: Number of files successfully Verified: 1
39039: Number of warnings: 0
39040: Number of errors: 0C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39041: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39042: https://maxbox4.wordpress.com/
39043: http://www.softwareschule.ch/teelicht.jpg
39044:
39045: Verifying: maXbox4.exe   4.7.1.80
39046: Signature Index: 0 (Primary Signature)
39047: Hash of file (sha1): 9D181955E579DCFF6920CEEC32749F0E43F2C157
39048: Signing Certificate Chain:
39049:     Issued to: maXboxCertAuth
39050:     Issued by: maXboxCertAuth
39051:     Expires:   Sun Jan 01 00:59:59 2040
39052:     SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
39053:         Issued to: maXbox4signer
39054:         Issued by: maXboxCertAuth
39055:         Expires:   Sun Jan 01 00:59:59 2040
39056:         SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
39057:             Issued to: maXbox4exe
39058:             Issued by: maXbox4signer
39059:             Expires:   Sun Jan 01 00:59:59 2040
39060:             SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
39061: The signature is timestamped: Tue Dec 03 10:51:06 2019
39062: Timestamp Verified by:
39063:     Issued to: Starfield Root Certificate Authority - G2
39064:     Issued by: Starfield Root Certificate Authority - G2
39065:     Expires:   Fri Jan 01 00:59:59 2038
39066:     SHA1 hash: B51C067CEE2B0C3DF855AB2D92F4FE39D4E70F0E
39067:         Issued to: Starfield Secure Certificate Authority - G2
39068:         Issued by: Starfield Root Certificate Authority - G2
39069:         Expires:   Sat May 03 08:00:00 2031
39070:         SHA1 hash: 7EDC376DCFD45E6DDF082C160DF6AC21835B95D4
39071:             Issued to: Starfield Timestamp Authority - G2
39072:             Issued by: Starfield Secure Certificate Authority - G2
39073:             Expires:   Tue Sep 17 08:00:00 2024
39074:             SHA1 hash: E8551398FF530A9278FD9818E448CB333F67924D
39075:
39076: Successfully verified: maXbox4.exe
```

```
39077: Number of files successfully Verified: 1
39078: Number of warnings: 0
39079: Number of errors: 0
39080: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39081:
39082: ****************************************************************
39083: Release Notes maXbox 4.7.2.82 April 2020 mX47
39084: ****************************************************************
39085: 1254 unit uPSI_MaskEdit.pas    FCL
39086: 1255 unit uPSI_SimpleRSSTypes; BlueHippo
39087: 1256 unit uPSI_SimpleRSS;      BlueHippo
39088: 1257 unit uPSI_psULib.pas      Prometheus
39089: 1258 unit uPSI_psUFinancial;   Prometheus
39090: 1259 uPSI_PsAPI_2.pas          mX4
39091: 1260 uPSI_PersistSettings_2    mX4
39092: Totals of Func Calls: 32339
39093: for 4.7.2.82
39094: https://sourceforge.net/projects/prometheuslibra/
39095:
39096: CDS Func CreateBlobStream(Field: TField; Mode: TBlobStreamMode): TStream;
39097:     Func BookmarkValid(Bookmark: TBookmark):Bool; ');
39098:     Func CompareBookmarks(Bookmark1, Bookmark2: TBookmark): Integer;
39099:
39100: Proc ModifyFontsFor(ctrl: TWinControl);
39101: Proc ModifyFontsFor(ctrl: TWinControl; afontname:Str);
39102: Proc GetAccessTimeOut(var bTimeOut: Bool; var bFeedBack:Boolean;var iTimeOutTime:Int);
39103: Func GetParentProcessName():Str;
39104: writeln('ParentProcessName: '+GetProcessName(GetParentProcessID(getprocessID)));
39105: Func wFillRect(hDC: HDC; const lprc: TRect; hbr: HBRUSH): Integer; stdcall;
39106: Func wFrameRect(hDC: HDC; const lprc: TRect; hbr: HBRUSH):Integer; stdcall;
39107: Proc GetIpAddresses(Results: TStrings); overload;
39108: Proc GetIpAddresses(Results: TStrings; const HostName: Ansistr);
39109: S.RegisterDelphiFunction(@GetIPAddresses, 'GetIPs', cdRegister);
39110: RegisterMethod(@TStringGrid.CellRect, 'CellRect');
39111: RegisterMethod(@TStringGrid.MouseToCell, 'MouseToCell');
39112: Proc SendKeyVar(AppName:Str; key: Variant);
39113: http://www.scalabium.com/faq/dct0069.htm
39114: Proc GetComponentNames(lst: TStrings);
39115: Proc AngleTextOut(ACanvas: TCanvas; Angle, X, Y: Integer; Str:str);
39116: Proc PrintText2Printer(s:Str);
39117: Proc UASwapEndian(var UC: TUArrDW);
39118: Func SliceByteArray(const B: array of Byte; Start, Len: Integer): TBytes;
39119: Proc TailOfFile(aFilename:str; aTailContainer: TStrings; MaxNumLines:integer=5);
39120: Func CloneByteArray(const B: array of Byte): TBytes;
39121: Func PopByteArray(var A: TBytes): Byte;
39122: Proc PushByteArray(const B: Byte; var A: TBytes);
39123: Proc AppendByteArray(var B1:TBytes; const B2: array of Byte);
39124: Func ShiftByteArray(var A: TBytes): Byte;
39125:
39126: ****************************************************************
39127: Release Notes maXbox 4.7.4.64 June 2020 mX47
39128: ****************************************************************
39129: add Tutor 57 - 72
39130: add NoGUI Shell Tutorial 71 and 70 Units
39131: 1307 unit uPSI_statmach,   {StateMachine}
39132: 1308 uPSI_uTPLb_RSA_Primitives,
39133: 1309 unit uPSI_UMatrix,    //for Tensorflow dll
39134: 1310 uPSI_DXUtil,
39135: 1311 uPSI_crlfParser,
39136: 1312 unit uPSI_DCPbase64;
39137: 1313 unit uPSI_FlyFilesUtils;
39138: 1314 uPSI_PJConsoleApp.pas
39139: 1315 uPSI_PJStreamWrapper.pas
39140: 1316 uPSI_LatLonDist,    //DFF
39141: 1317 uPSI_cHash2.pas     //Fundamentals SHA512
39142: 1318 uPSI_ZLib2.pas      //compressor
39143: 1319 unit uPSI_commDriver
39144: 1320 unit uPSI_PXLNetComs.pas   //PXL
39145: 1321 unit uPSI_PXLTiming.pas    //PXL
39146: 1322 uPSI_Odometer.pas
39147: 1323 unit uPSI_UIntList2;
39148: 1324 uPSI_UIntegerpartition.pas
39149: 1325 unit uPSI_idPHPRunner.pas  //prepare for PHP4D
39150: 1326 unit uPSI_idCGIRunner.pas
39151: 1327 uPSI_DrBobCGI,             //4.7.1.20
39152: 1328 uPSI_OverbyteIcsLogger,
39153: 1329 uPSI_OverbyteIcsNntpCli,  testset
39154: 1330 uPSI_OverbyteIcsCharsetUtils,
39155: 1331 uPSI_OverbyteIcsMimeUtils,
39156: 1332 uPSI_OverbyteIcsUrl(CL: TPSPascalCompiler);
39157: 1333 uPSI_uWebSocket.pas
39158: 1334 uPSI_KhFunction.pas
39159: 1335 uPSI_ALOpenOffice.pas
39160: 1336 unit uPSI_ALLibPhoneNumber
39161: 1337 unit uPSI_ALPhpRunner2;
39162: 1338 unit uPSI_ALWebSpider2;
39163: 1339 unit uPSI_ALFcnHTML2; // RunJavaScript2
39164: 1340 unit uPSI_ALExecute2.pas
39165: 1341 uPSI_ALIsapiHTTP.pas
```

```
39166: 1342 uPSI_ALOpenOffice_Routines
39167: 1343 unit uPSI_uUsb;
39168: 1344 uPSI_uWebcam.pas
39169: 1345 uPSI_PersistSettings.pas //fixing
39170: 1346 uPSI_uTPLb_MemoryStreamPool.pas
39171: 1347 uPSI_uTPLb_Signatory.pas
39172: 1348 unit uPSI_uTPLb_Constants.pas  //TurboPower
39173: 1349 uPSI_uTPLb_Random.pas
39174: 1350 unit uPSI_uTPLb_PointerArithmetic;
39175: 1351 unit uPSI_EwbCoreTools.pas
39176: 1352 unit uPSI_EwbUrl.pas
39177: 1353 unit uPSI_SendMail_For_Ewb.pas
39178: 1354 unit uPSI_MaskEdit.pas    FCL
39179: 1355 unit uPSI_SimpleRSSTypes; BlueHippo
39180: 1356 unit uPSI_SimpleRSS;      BlueHippo
39181: 1357 unit uPSI_psULib.pas      Prometheus
39182: 1358 unit uPSI_psUFinancial;   Prometheus
39183: 1359 uPSI_PsAPI_2.pas          mX4
39184: 1360 uPSI_PersistSettings_2    mX4
39185: 1361 uPSI_rfc1213util2.pas     IP
39186: 1362 uPSI_JTools.pas           JCL
39187: 1363 unit uPSI_neuralbit.pas   CAI
39188: 1364 unit uPSI_neuralab.pas    CAI
39189: 1365 unit uPSI_winsvc2.pas     TEK
39190: 1366 unit uPSI_wmiserv2.pas    TEK
39191: 1367 uPSI_neuralcache.pas      CAI
39192: 1368 uPSI_neuralbyteprediction CAI
39193: 1369 unit uPSI_USolarSystem;   glscene.org
39194: 1370 uPSI_USearchAnagrams.pas  DFF
39195: 1371 uPSI_JsonsUtilsEx.pas     Randolph
39196: 1372 unit uPSI_Jsons.pas       Randolph
39197: 1373 unit uPSI_HashUnit;       DFF
39198: 1374 uPSI_U_Invertedtext.pas   DFF
39199: 1275 unit uPSI_Bricks;         Dendron
39200: 1376 unit uPSI_lifeblocks.pas  Dendron
39201:
39202: Bugfix: Pagecontrol,TTabcontrol with change event TTabChangingEvent,TGetSiteInfoEvent,
39203:   Json extension with objects and arrays
39204:   TCheckListBox.Style  Property TListBoxStyle Enumeration
39205:   glsincos11 glsincos0 fix uPSI_UMatrix; Gaussian_Elimination fix  array[1..30] of Extended
39206:   more ico and bmp resouces, fix Textmetric type and strtochars() alias StrtoCharSet()
39207:   shortcut Ctrl+Alt+M to jump to output (memo2) console
39208:
39209: Func MercScaling(Long0,lat0,long1,lat1:extended;x0,y0,x1,y1:integer) :TMercScalingRec');
39210: Func LonglatToPlotPt(Long,lat:Extended; Scalerec:TMercScalingrec):TPoint');
39211: Func PlotPtToLonglat(PlotPt:TPoint; Scalerec:TMercScalingrec):TRealPoint');
39212: Func SphericalEarthDistance(lat1,lon1,lat2,lon2:extended;Units:integer):extended');
39213: Proc PrintUsingShell(psFileName :str);
39214: Func LocalExecute32(FileName:Str;Wait:bool;Visibility:int;lWaitFor:Card=INFINITE):int;
39215:
39216: TidCGIRunner component allows to execute CGI scripts using
39217: Indy TidHTTPServer component.
39218: // Project:     1307 State Machine
39219: // Module:      statmach.pas
39220: // Description: Visual Finite State Machine.
39221: // Version:     2.2a,  Release: 6 preprocessor, compiler, interpreter
39222: Func VarArrayToStr(const vArray: variant):Str;
39223: Func GetWMIObject(const objectName:Str): IDispatch; //create the Wmi instance
39224: Func GetAntiVirusProductInfo: TStringlist;
39225: Proc GetWaveOutDevices(DeviceNames: TStrings);
39226: Proc GetMIDIOutDevices(Devices: TStrings);
39227: Proc SetDecimalSeparator2(Ch: Char);
39228: Totals of Func Calls: 32356
39229: Proc ResizeBitmap(Bitmap:TBitmap;Width,Height:Integer;Background: TColor);
39230: Func ProcessCount(const ExeName:str): Integer;
39231: Func MaxWidthOfStrings(const Strings:Classes.TStrings;const Font:Graphics.TFont):Integer;
39232: Proc ScreenShotMonitor(var Bitmap:TBitmap;const MonitorNum:Integer;const DrawCursor:Bool;const
       Quality:TPixelFormat);
39233:
39234: https://github.com/maxkleiner/maXbox/blob/master/logisticregression2.ipynb
39235:
39236: ---- mX4 bigbitbox code_cleared_checked  IMPORTS DLL LIB ----
39237: AVICAP32.DLL  01 [+] AVICAP32.DLL
39238: AVICAP32.dll  02 [+] AVICAP32.dll
39239: GLU32.dll     03 [+] GLU32.dll
39240: IMAGEHLP.DLL  04 [+] IMAGEHLP.DLL
39241: KERNEL32.DLL  05 [+] KERNEL32.DLL
39242: MSVCRT.DLL    06 [+] MSVCRT.DLL
39243: MSVFW32.DLL   07 [+] MSVFW32.DLL
39244: OpenGL32.dll  08 [+] OpenGL32.dll
39245: SHFolder.dll  09 [+] SHFolder.dll
39246: SHfolder.dll  10 [+] SHfolder.dll
39247: URLMON.DLL    11 [+] URLMON.DLL
39248: advapi32.dll  12 [+] advapi32.dll
39249: comctl32.dll  13 [+] comctl32.dll
39250: comdlg32.dll  14 [+] comdlg32.dll
39251: gdi32.dll     15 [+] gdi32.dll
39252: imagehlp.dll  16 [+] imagehlp.dll
39253: imm32.dll     17 [+] imm32.dll
```

```
39254: iphlpapi.dll  18 [+] iphlpapi.dll
39255: kernel32.dll  19 [+] kernel32.dll
39256: mpr.dll       20 [+] mpr.dll
39257: msacm32.dll   21 [+] msacm32.dll
39258: msimg32.dll   22 [+] msimg32.dll
39259: netapi32.dll  23 [+] netapi32.dll
39260: ole32.dll     24 [+] ole32.dll
39261: oleacc.dll    25 [+] oleacc.dll
39262: oleaut32.dll  26 [+] oleaut32.dll
39263: oledlg.dll    27 [+] oledlg.dll
39264: opengl32.dll  28 [+] opengl32.dll
39265: shell32.dll   29 [+] shell32.dll
39266: shlwapi.dll   30 [+] shlwapi.dll
39267: user32.dll    31 [+] user32.dll
39268: usp10.dll     32 [+] usp10.dll
39269: version.dll   33 [+] version.dll
39270: winhttp.dll   34 [+] winhttp.dll
39271: wininet.dll   35 [+] wininet.dll
39272: winmm.dll     36 [+] winmm.dll
39273: winspool.drv  37 [+] winspool.drv
39274: ws2_32.dll    38 [+] ws2_32.dll
39275: wsock32.dll   39 [+] wsock32.dll
39276: fannfloat.dll 40 [++]fannfloat.dll
39277: dmath.dll     41 [++]dmath.dll
39278: midas.dll     42 [++]midas.dll
39279:
39280: [+] AVICAP32.DLL    [+] AVICAP32.dll  [+] GLU32.dll      [+] IMAGEHLP.DLL
39281: [+] KERNEL32.DLL    [+] MSVCRT.DLL    [+] MSVFW32.DLL    [+] OpenGL32.dll
39282: [+] SHFolder.dll    [+] SHfolder.dll  [+] URLMON.DLL     [+] advapi32.dll
39283: [+] comctl32.dll    [+] comdlg32.dll  [+] gdi32.dll      [+] imagehlp.dll
39284: [+] imm32.dll       [+] iphlpapi.dll  [+] kernel32.dll   [+] mpr.dll
39285: [+] msacm32.dll     [+] msimg32.dll   [+] netapi32.dll   [+] ole32.dll
39286: [+] oleacc.dll      [+] oleaut32.dll  [+] oledlg.dll     [+] opengl32.dll
39287: [+] shell32.dll     [+] shlwapi.dll   [+] user32.dll     [+] usp10.dll
39288: [+] version.dll     [+] winhttp.dll   [+] wininet.dll    [+] winmm.dll [+] midas.dll
39289: [+] winspool.drv    [+] ws2_32.dll    [+] wsock32.dll    [+] dmath.dll [+] fannfloat.dll
39290:
39291: Exports   CreateIncome      ExePath         GetASCII                GetBiosVendor
39292:           GetMemoryInfo     GetOSName       GetProcessorName        GetScriptName2
39293:           GetScriptPath2    IsCOMConnected  IsInternet              RemainingBatteryPercent
39294:           UpTime            getBigPI        getHostIP               isSound
39295:
39296: memo1.CodeFolding.FoldRegions.Add begin','end''try','end repeat, until''case','end {$IFDEF','{$ENDIF}};
39297: memo1.InitCodeFolding;  true, '{','}
39298: writeln(GETDOSOutput('cmd.exe /c wmic cpu get name','C:\'));
39299: >>> Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
39300: V 4.7.4.64
39301: Totals of Func Calls: 32633
39302: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
39303: CRC32: 3EB27A87:  28.2 MB (29,608,248) bytes
39304: V 4.7.5.20
39305: Totals of Func Calls: 33282
39306: SHA1: of 4.7.5.20 D82EAD01C58738887661428F94B207DB1D8FAEB5
39307: CRC32: 203C82F0  29.5 MB (31,012,768 bytes
39308: V 4.7.5.80
39309: Totals of Calls: 33848
39310: SHA1: of 4.7.5.80 3E38A48072D4F828A4BE4A52320F092FE50AE9C3
39311: CRC32: 198E756B:  31344456 bytes
39312: ▯▯▯ mX4 executed: 27/07/2021 19:05:10  Runtime:0:12:3.247  Memload: 39% use
39313: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool sign /f "maxbox4exe
39314: .pfx" /p "password" /tr http://timestamp.comodoca.com /td SHA256 maXbox4.exe
39315: C:\maXbox\EKON_BASTA\\Windows Kits\10\bin\x64>signtool verify /v /pa maXbox4.exe
39316: For SHA256 the Parameter name has to be /tr. The /t is only valid for SHA1.
39317:  ExecuteShell(OpenSSL_Path+'openssl.exe',
39318: 'pkcs12 -in'+OpenSSL_Path+'./certs/maXboxCertAuth3.pfx -out'+OpenSSL_Path+'./certs/maXboxCertAuth3.pfx.pem
       -nodes')
39319: Verifying: maXbox4.exe 4.7.6.10 IV  Compilation Timestamp 2022-01-14 13:20:37
39320: Entry Point 24467336  Contained Sections 10
39321:
39322: ****************************************************************
39323: Release Notes maXbox 4.7.6.10 VIII August 2022 mX476
39324: ****************************************************************
39325: Add 48 Units + 16 Tutorials
39326:
39327: 1441 unit uPSI_neuralgeneric.pas; CAI
39328: 1442 unit uPSI_neuralthread.pas; CAI
39329: 1443 unit uPSI_uSysTools; TuO
39330: 1444 unit upsi_neuralsets; mX4
39331: 1445 unit uPSI_uWinNT.pas mX4
39332: 1446 unit uPSI_URungeKutta4.pas ICS
39333: 1447 unit uPSI_UrlConIcs.pas ICS
39334: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
39335: 1449 unit uPSI_Numedit2 mX4
39336: 1450 unit uPSI_PsAPI_3.pas mX4
39337: 1451 unit uPSI_SeSHA256.pas
39338: 1452 unit IdHashMessageDigest_max2;
39339: 1453 unit uPSI_BlocksUnit.pas
39340: 1454 unit uPSI_DelticsCommandLine.pas
39341: 1455 unit uPSI_DelticsStrUtils;
```

```
39342: 1456 unit uPSI_DelticsBitField;
39343: 1457 unit uPSI_DelticsSysUtils;
39344: 1458 unit uPSI_ALIniFiles2.pas
39345: 1459 unit uPSI_StarCalc2.pas
39346: 1460 unit uPSI_IdHashMessageDigest2.pas
39347: 1461 unit uPSI_U_Splines;
39348: 1462 unit uPSI_U_CoasterB.pas;
39349: 1463 U_SpringMass2.pas
39350: 1464 uPSI_MARSCoreUtils;
39351: 1465 unit uPSI_clJsonParser.pas
39352: 1466 unit uPSI_SynHighlighterPython.pas
39353: 1467 unit uPSI_DudsCommonDelphi;
39354: 1468 unit uPSI_AINNNeuron;
39355: 1469 unit uPSI_PJConsoleApp2;
39356: 1470 unit uPSI_PJPipeFilters2;
39357: 1471 unit uPSI_uHTMLBuilder;
39358: 1472 unit uPSI_PJPipe2;
39359: 1473 uPSI_WinApiDownload,
39360: 1474 uPSI_pxQRcode,     //beta
39361: 1475 unit uPSI_neuralplanbuilder2
39362: 1476 unit uPSI_DelphiZXingQRCode;
39363: 1477 unit uPSI_RestJsonUtils;
39364: 1478 unit UtilsTimeCode;
39365: 1479 unit uPSC_classes2.pas; //TList
39366: 1480 unit uPSC_std2.pas
39367: 1481 unit uPSI_maxIniFiles.pas
39368: 1482 unit uROPSImports.pas
39369: 1483 unit uROPSServerLink.pas
39370: 1484 unit uPSI_KLibUtils;
39371: 1485 unit uPSI_PathFunc2;  //inno setup
39372: 1486 unit KLibVC_Redist.pas;
39373: 1487 unit HTTPApp2.pas;
39374: 1488 unit uPSI_XCollection2;
39375:
39376: Total of Function Calls: 35597
39377: SHA1: 4.7.6.10 AA16F6BCD100F0AD16A39CD9741E78A96DF5F7F1
39378: CRC32: 412DE996: 31.7 MB (33,331,992 bytes)
39379: Amount of Functions: 21002
39380: Amount of Procedures: 12504
39381: Amount of Constructors: 1980
39382: Amount of Destructors: 14
39383: Totals of Calls: 35500
39384: SHA1: of 4.7.6.10 AEF0EE239713A3E142EE3480F86C30B9F0B872FD
39385: CRC32: 2D7A3208:  33300808 bytes
39386: ▯▯▯ mX4 executed: 19/03/2022 09:21:28  Runtime: 0:13:36.211  Memload: 42% use
39387: Amount of Functions: 21220
39388: Amount of Procedures: 12616
39389: Amount of Constructors: 1992
39390: Amount of Destructors: 14
39391: Totals of Calls: 35842
39392: SHA1: of 4.7.6.10 B8590FA3326B735E14D9687076B302461D8D49D9
39393: CRC32: CCBD8C56:  33472792 bytes
39394: ▯▯▯ mX4 executed: 26/08/2022 09:53:21  Runtime: 0:14:24.171  Memload: 42% use
39395: ▯▯▯
39396: Total of Function Calls: 35842
39397: SHA1: 4.7.6.10 B8590FA3326B735E14D9687076B302461D8D49D9
39398: CRC32: CCBD8C56: 31.9 MB (33,472,792 bytes)
39399: recomplied
39400: 25/08/2022  22:41           483,539 fMain.dcu
39401: 25/08/2022  22:42            71,769 ALHttpClient2.dcu
39402: 25/08/2022  15:48            48,080 ALMultiPartParser.dcu
39403: 25/08/2022  22:11            20,366 ALWininetHttpClient2.dcu
39404: 25/08/2022  22:42            13,520 BlocksUnit.dcu
39405: 25/08/2022  23:24            30,875 MathsLib.dcu
39406: 25/08/2022  22:42             6,566 PXLTiming.dcu
39407: 25/08/2022  22:42             7,431 PythonAction.dcu
39408: 25/08/2022  22:42           279,159 PythonEngine.dcu
39409: 25/08/2022  22:42             6,446 SeSHA256.dcu
39410: 25/08/2022  20:14            45,539 UBigIntsV4.dcu
39411: 25/08/2022  22:42            66,215 uPSI_ALHttpClient2.dcu
39412: 25/08/2022  14:03            39,724 uPSI_ALHttpCommon.dcu
39413: 25/08/2022  23:22             7,652 uPSI_ALWininetHttpClient2.dcu
39414: 25/08/2022  22:42            28,590 uPSI_AzuliaUtils.dcu
39415: 25/08/2022  22:42            20,494 uPSI_BlocksUnit.dcu
39416: 25/08/2022  22:42             8,645 uPSI_PXLTiming.dcu
39417: 25/08/2022  22:42           175,123 uPSI_PythonEngine.dcu
39418: 25/08/2022  20:14            19,746 uPSI_UBigIntsV4.dcu
39419: 25/08/2022  22:42            46,239 VarPyth.dcu
39420: 25/08/2022  14:03            77,150 OverbyteIcsUtils.dcu
39421: ▯▯▯
39422: Recompiled: dir /s /o N *.dcu | find "03/01/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
39423: 03/01/2022  14:28  479,283 fMain.dcu
39424: 03/01/2022  14:28   13,506 BlocksUnit.dcu
39425: 03/01/2022  13:05   30,110 clJsonParser.dcu
39426: 03/01/2022  15:22   30,878 MathsLib.dcu
39427: 03/01/2022  14:28    6,566 PXLTiming.dcu
39428: 03/01/2022  14:28    7,417 PythonAction.dcu
39429: 03/01/2022  14:28  279,035 PythonEngine.dcu
39430: 03/01/2022  14:28    6,446 SeSHA256.dcu
```

```
39431: 03/01/2022  14:28   20,494 uPSI_BlocksUnit.dcu
39432: 03/01/2022  13:50   21,242 uPSI_clJsonParser.dcu
39433: 03/01/2022  14:11   59,984 uPSI_OverbyteIcsUtils.dcu
39434: 03/01/2022  14:28    8,645 uPSI_PXLTiming.dcu
39435: 03/01/2022  14:28  175,131 uPSI_PythonEngine.dcu
39436: 03/01/2022  10:50   37,866 uPSI_U_Splines.dcu
39437: 03/01/2022  14:50   30,632 uPSI_uLkJSON.dcu
39438: 03/01/2022  11:45   44,566 uPSI_VectorLists.dcu
39439: 03/01/2022  14:28   46,239 VarPyth.dcu
39440: 03/01/2022  14:11   40,901 OverbyteIcsMimeUtils.dcu
39441: 03/01/2022  14:28   77,151 OverbyteIcsUtils.dcu
39442: 03/01/2022  15:21  108,526 uPSI_UtilsMax4.dcu
39443: 02/01/2022  11:52   11,355 U_Splines.dcu
39444: 02/01/2022  11:27   24,281 uPSI_neuraldatasets.dcu
39445: 02/01/2022  12:15   77,381 uPSI_neuralvolume.dcu
39446: 01/01/2022  22:44   48,480 U_CoasterB.dcu
39447: 01/01/2022  22:44   34,098 uPSI_U_CoasterB.dcu
39448:
39449: mX4.7.6.10 VI
39450: 16/02/2022  21:29  50,625 uPSI_StrUtils.dcu
39451: 16/02/2022  21:36  59,984 uPSI_OverbyteIcsUtils.dcu
39452: 16/02/2022  15:49  16,601 uPSI_PJConsoleApp.dcu
39453: 16/02/2022  15:27   6,527 uPSI_PJPipe.dcu
39454: 16/02/2022  18:25 105,750 uPSI_PsAPI.dcu
39455: 16/02/2022  21:36  40,891 OverbyteIcsMimeUtils.dcu
39456:
39457: 17/02/2022  11:02 479,639 fMain.dcu
39458: 17/02/2022  09:55  30,889 uPSUtils.dcu
39459: 17/02/2022  10:56   3,723 AINNNeuron.dcu
39460: 17/02/2022  11:02  13,524 BlocksUnit.dcu
39461: 17/02/2022  11:27   9,594 DudsCommonDelphi.dcu
39462: 17/02/2022  11:34  30,868 MathsLib.dcu
39463: 17/02/2022  11:02   6,568 PXLTiming.dcu
39464: 17/02/2022  11:02   7,434 PythonAction.dcu
39465: 17/02/2022  11:02 278,950 PythonEngine.dcu
39466: 17/02/2022  11:02   6,446 SeSHA256.dcu
39467: 17/02/2022  11:02   5,744 uPSI_AINNNeuron.dcu
39468: 17/02/2022  11:02  20,494 uPSI_BlocksUnit.dcu
39469: 17/02/2022  11:23   5,442 uPSI_DudsCommonDelphi.dcu
39470: 17/02/2022  10:07  11,793 uPSI_PJEnvVars.dcu
39471: 17/02/2022  11:02   8,642 uPSI_PXLTiming.dcu
39472: 17/02/2022  11:02 175,123 uPSI_PythonEngine.dcu
39473: 17/02/2022  11:02  46,244 VarPyth.dcu
39474: 17/02/2022  11:02  77,152 OverbyteIcsUtils.dcu
39475:
39476: mX4.7.6.10 VII
39477: 17/03/2022  14:08    194,307 IFSI_WinForm1puzzle.dcu
39478: 17/03/2022  20:30     46,854 uPSC_classes.dcu
39479: 17/03/2022  19:48      5,226 uPSC_std.dcu
39480: 17/03/2022  18:49     53,416 uPSR_classes.dcu
39481: 17/03/2022  20:30     19,528 uPSI_SynEditHighlighter.dcu
39482: 17/03/2022  16:03     11,784 uPSI_SynEditMiscProcs.dcu
39483: 17/03/2022  21:15    173,341 uPSI_TeEngine.dcu
39484: 17/03/2022  21:16     30,854 MathsLib.dcu
39485: 17/03/2022  20:13     77,627 uPSI_neuralvolume.dcu
39486: 17/03/2022  13:10     13,188 uPSI_UcomboV2.dcu
39487: 17/03/2022  13:26     14,813 uPSI_UTGraphSearch.dcu
39488: 17/03/2022  13:22     21,041 UTGraphSearch.dcu
39489: 17/03/2022  08:35    <DIR>          n
39490:
39491: 18/03/2022  18:38      5,234 uPSC_std.dcu
39492: 18/03/2022  18:38     30,854 MathsLib.dcu
39493: 18/03/2022  18:33     77,624 uPSI_neuralvolume.dcu
39494:
39495: mX4.7.6.10 VIII
39496: Recompiled Units:
39497: c:\maXbox>dir /s /o N *.dcu | find "01/08/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
39498: 28/07/2022  18:44           211,349 IFSI_SysUtils_max.dcu
39499: 28/07/2022  15:56            46,996 uPSC_classes.dcu
39500: 28/07/2022  15:23            49,036 uPSC_DB.dcu
39501: 28/07/2022  16:02            53,529 uPSR_classes.dcu
39502: 28/07/2022  15:23            89,402 uPSR_DB.dcu
39503: 28/07/2022  22:37            11,941 uPSI_SynEditMiscProcs.dcu
39504: 28/07/2022  22:37            26,896 uPSI_StrUtil.dcu
39505: 28/07/2022  14:51             9,910 uPSI_IdThreadSafe.dcu
39506: 31/07/2022  22:43             7,253 KLibTypes.dcu
39507: 31/07/2022  21:46            59,910 uPSI_OverbyteIcsUtils.dcu
39508: 31/07/2022  21:38            10,411 uPSI_XCollection.dcu
39509: 31/07/2022  21:38            40,887 OverbyteIcsMimeUtils.dcu
39510: 01/08/2022  12:28           482,562 fMain.dcu
39511: 01/08/2022  13:37            66,516 uPSI_HTTPApp.dcu
39512: 01/08/2022  12:28            13,517 BlocksUnit.dcu
39513: 01/08/2022  14:19            38,401 KLibUtils.dcu
39514: 01/08/2022  14:22            30,856 MathsLib.dcu
39515: 01/08/2022  12:28             6,566 PXLTiming.dcu
39516: 01/08/2022  12:28             7,430 PythonAction.dcu
39517: 01/08/2022  12:28           278,932 PythonEngine.dcu
39518: 01/08/2022  12:28             6,446 SeSHA256.dcu
39519: 01/08/2022  12:28            20,494 uPSI_BlocksUnit.dcu
```

```
39520: 01/08/2022  14:21            21,506 uPSI_KLibUtils.dcu
39521: 01/08/2022  12:22             7,927 uPSI_PathFunc.dcu
39522: 01/08/2022  12:28             8,645 uPSI_PXLTiming.dcu
39523: 01/08/2022  12:28           175,131 uPSI_PythonEngine.dcu
39524: 01/08/2022  12:28            46,238 VarPyth.dcu
39525: 01/08/2022  12:28            77,151 OverbyteIcsUtils.dcu
39526: 03/08/2022  18:25           482,909 fMain.dcu
39527: 03/08/2022  18:24            64,415 AzuliaUtils.dcu
39528: 03/08/2022  18:25            13,523 BlocksUnit.dcu
39529: 03/08/2022  14:27             3,112 KlibConstants.dcu
39530: 03/08/2022  15:15            41,680 KLibWindows.dcu
39531: 03/08/2022  18:36            30,869 MathsLib.dcu
39532: 03/08/2022  18:25             6,568 PXLTiming.dcu
39533: 03/08/2022  18:25             7,434 PythonAction.dcu
39534: 03/08/2022  18:25           278,919 PythonEngine.dcu
39535: 03/08/2022  18:25             6,446 SeSHA256.dcu
39536: 03/08/2022  14:21            22,204 uPSI_ALHttpClient.dcu
39537: 03/08/2022  18:24            28,585 uPSI_AzuliaUtils.dcu
39538: 03/08/2022  18:25            20,494 uPSI_BlocksUnit.dcu
39539: 03/08/2022  16:33            17,512 uPSI_KLibWindows.dcu
39540: 03/08/2022  18:15            59,933 uPSI_OverbyteIcsUtils.dcu
39541: 03/08/2022  18:25             8,644 uPSI_PXLTiming.dcu
39542: 03/08/2022  18:25           175,123 uPSI_PythonEngine.dcu
39543: 03/08/2022  16:35           108,026 uPSI_VectorGeometry.dcu
39544: 03/08/2022  18:25            46,235 VarPyth.dcu
39545: 03/08/2022  18:15            40,883 OverbyteIcsMimeUtils.dcu
39546: 03/08/2022  18:25            77,145 OverbyteIcsUtils.dcu
39547:
39548: Target Machine Intel 386 or later processors and compatible processors
39549: Compilation Timestamp 2022-08-03 16:37:05 UTC
39550: Entry Point 24656140
39551: Contained Sections 10
39552: Target Machine Intel 386 or later processors and compatible processors
39553: Compilation Timestamp 2022-08-29 09:10:00 UTC
39554: Entry Point 24725780
39555: Contained Sections 10
39556:
39557: mX4.7.6.10 X
39558: Recompiled Units:
39559: C:\maXbox>dir /s /o N *.dcu | find "23/09/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
39560: 23/09/2022  16:15           483,806 fMain.dcu
39561: 23/09/2022  10:00           194,750 IFSI_WinForm1puzzle.dcu
39562: 23/09/2022  12:43            81,932 uPSI_Chart.dcu
39563: 23/09/2022  11:39            97,061 uPSI_TeCanvas.dcu
39564: 23/09/2022  11:39           177,404 uPSI_TeEngine.dcu
39565: 23/09/2022  11:20            73,888 uPSI_TeeProcs.dcu
39566: 23/09/2022  16:16            71,782 ALHttpClient2.dcu
39567: 23/09/2022  16:16            13,522 BlocksUnit.dcu
39568: 23/09/2022  17:45            30,862 MathsLib.dcu
39569: 23/09/2022  16:16             6,568 PXLTiming.dcu
39570: 23/09/2022  16:16             7,434 PythonAction.dcu
39571: 23/09/2022  16:16           278,867 PythonEngine.dcu
39572: 23/09/2022  16:06            10,517 RestUtils.dcu
39573: 23/09/2022  16:16             6,446 SeSHA256.dcu
39574: 23/09/2022  16:16            66,632 uPSI_ALHttpClient2.dcu
39575: 23/09/2022  17:45            31,846 uPSI_AzuliaUtils.dcu
39576: 23/09/2022  16:16            20,494 uPSI_BlocksUnit.dcu
39577: 23/09/2022  16:16             8,642 uPSI_PXLTiming.dcu
39578: 23/09/2022  16:16           175,123 uPSI_PythonEngine.dcu
39579: 23/09/2022  16:15            11,240 uPSI_RestUtils.dcu
39580: 23/09/2022  16:16            46,235 VarPyth.dcu
39581: 23/09/2022  15:11            63,422 uPSI_Series.dcu
39582:
39583: Total of Function Calls: 35958
39584: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
39585: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
39586: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
39587: Entry Point 24783132 - Contained Sections 10
39588:
39589: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maXbox4.exe
39590: Verifying: maXbox4.exe
39591: Signature Index: 0 (Primary Signature)
39592: Hash of file (sha1): 5E556D92930A5ED88CBDAE8E5A174EE8D3161F56
39593:
39594: Signing Certificate Chain:
39595:     Issued to: maXboxCertAuth
39596:     Issued by: maXboxCertAuth
39597:     Expires:   Sun Jan 01 01:59:59 2040
39598:     SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
39599:
39600:         Issued to: maXbox4signer
39601:         Issued by: maXboxCertAuth
39602:         Expires:   Sun Jan 01 01:59:59 2040
39603:         SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
39604:
39605:             Issued to: maXbox4exe
39606:             Issued by: maXbox4signer
39607:             Expires:   Sun Jan 01 01:59:59 2040
39608:             SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
```

```
39609:
39610: The signature is timestamped: Mon Oct 17 11:48:48 2022
39611: Timestamp Verified by:
39612:     Issued to: USERTrust RSA Certification Authority
39613:     Issued by: USERTrust RSA Certification Authority
39614:     Expires:   Tue Jan 19 01:59:59 2038
39615:     SHA1 hash: 2B8F1B57330DBBA2D07A6C51F70EE90DDAB9AD8E
39616:         Issued to: Sectigo RSA Time Stamping CA
39617:         Issued by: USERTrust RSA Certification Authority
39618:         Expires:   Tue Jan 19 01:59:59 2038
39619:         SHA1 hash: 02D65B95E28370C1570095FA88F923DD937FAD8F
39620:             Issued to: Sectigo RSA Time Stamping Signer #3
39621:             Issued by: Sectigo RSA Time Stamping CA
39622:             Expires:   Thu Aug 11 01:59:59 2033
39623:             SHA1 hash: AB34013AAC4097319F081AF0B318E183F80F7881
39624:
39625: Successfully verified: maXbox4.exe
39626:
39627: Number of files successfully Verified: 1
39628: Number of warnings: 0
39629: Number of errors: 0
39630:
39631: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39632:
39633: MD5 863f4a600090421779a0d4855f31593c
39634: SHA-1 b2b44fc20da22111ef81553b94b0b903b9d7fa7a
39635: SHA-256 66e02bfc560c5082c2bfcb5e4b8e2a7b19fa1f60386474097625d55c7cc6b472
39636: Vhash 0370a6666d5c0d555c051031e0900e5003116012z33034z17013109054843115z10
39637: Authentihash 4999cd522416fd5fc367b54d2d00b253177a463c174fea96ece121a50e639e16
39638: Imphash da442fa7c70f4ee112e5300f1dacb062
39639: SSDEEP 393216:lWhOYZrD7vZw7VJ1T8pGxysz1Pg8DeDAmcAR9+xR7R/j4ENWDQXZd5eGlJFUVTKv:0OYZrD7cvNbzmcs9I9Rn5PdGfod
39640: TLSH T113778C62B1C0D632D05705788C0FD6E85A6A3D206DB4949B79F6BF8D2E311A17F362CB
39641: File type Win32 EXE
39642: Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
39643: TrID Windows ActiveX control (60.1%)   InstallShield setup (22.2%)   Win64 Executable (generic) (5.4%)
       DOS Borland compiled Executable (generic) (5.1%)   Win32 Executable (generic) (2.3%)
39644: DetectItEasy PE32   Compiler: Borland Delphi (2006)   Linker: Turbo Linker (2.25*,Delphi) [GUI32,signed]
39645: File size 32.44 MB (34014488 bytes)
39646:
39647: Verifying: maXbox4.exe
39648: Signature Index: 0 (Primary Signature)
39649: Hash of file (sha1): 221468297F0D9C645950C99EFB8F92C12E1C9A39
39650: Signing Certificate Chain:
39651:     Issued to: maXboxCertAuth
39652:     Issued by: maXboxCertAuth
39653:     Expires:   Sun Jan 01 00:59:59 2040
39654:     SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
39655:         Issued to: maXbox4signer
39656:         Issued by: maXboxCertAuth
39657:         Expires:   Sun Jan 01 00:59:59 2040
39658:         SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
39659:             Issued to: maXbox4exe
39660:             Issued by: maXbox4signer
39661:             Expires:   Sun Jan 01 00:59:59 2040
39662:             SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
39663:
39664: The signature is timestamped: Thu Dec 08 11:12:55 2022
39665: Timestamp Verified by:
39666:     Issued to: USERTrust RSA Certification Authority
39667:     Issued by: USERTrust RSA Certification Authority
39668:     Expires:   Tue Jan 19 00:59:59 2038
39669:     SHA1 hash: 2B8F1B57330DBBA2D07A6C51F70EE90DDAB9AD8E
39670:         Issued to: Sectigo RSA Time Stamping CA
39671:         Issued by: USERTrust RSA Certification Authority
39672:         Expires:   Tue Jan 19 00:59:59 2038
39673:         SHA1 hash: 02D65B95E28370C1570095FA88F923DD937FAD8F
39674:             Issued to: Sectigo RSA Time Stamping Signer #3
39675:             Issued by: Sectigo RSA Time Stamping CA
39676:             Expires:   Thu Aug 11 00:59:59 2033
39677:             SHA1 hash: AB34013AAC4097319F081AF0B318E183F80F7881
39678: Successfully verified: maXbox4.exe
39679: Number of files successfully Verified: 1
39680: Number of warnings: 0
39681: Number of errors: 0
39682: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
39683:
39684: Borland Conf 1999 Video: https://www.youtube.com/watch?v=m1Gn4echQIE
39685: http://www.softwareschule.ch/examples/classifier_compare2confusion2.py.htm
39686: 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.
39687: https://www.udemy.com/course/learn-coding-from-the-scratch/learn/
39688: - Valuable information-Clear explanations Engaging delivery-Helpful practice activities
39689: - Accurate course description - Knowledgeable instructor
39690:
39691: This Version is dedicated to Katherine Goble Johnson, NASA mathematician
39692: --- The girl who asked questions ---
39693:
```

https://github.com/joaopauloschuler/neural-api
https://sourceforge.net/projects/cai/
http://www.formatio-reticularis.de/code.html
http://cyberunits.sourceforge.net/
https://torry.net/files/tools/developers/scripters/maxbox4.zip
https://softwareschule.code.blog/
https://maxbox4.wordpress.com/
https://my6.code.blog/2020/01/17/maxbox4machine/
https://www.filecroco.com/download-maxbox/
https://towardsdatascience.com/how-to-implement-machine-learning-for-predictive-maintenance-4633cdbe4860
https://www.udemy.com/course/learn-coding-from-the-scratch/learn/
https://data.world/maxbox
https://data.world/maxbox/maxbox
https://maxbox4.files.wordpress.com/2020/08/twinwizard-copy.jpeg?w=688&h=&zoom=2
https://maxbox4.files.wordpress.com/2020/07/ice4_interlaken_max.jpg?w=768&h=&zoom=2
https://github.com/maxkleiner/maXbox4/blob/master/objectdetector3.ipynb
https://colab.research.google.com/github/maxkleiner/maXbox4/blob/master/objectdetector3.ipynb
https://my6.code.blog/
https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
https://kiosk.entwickler.de/windows-developer-magazin/windows-developer-magazin-10-20/erklaerbare-modelle-schaffen-vertrauen/
https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
https://repl.it/@MaxKleiner/machinelearning4#main.py
https://maxbox4.wordpress.com/2020/05/15/ekon-24/
https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb
https://github.com/maxkleiner/maXbox4/blob/master/sentimenttree2.ipynb
https://towardsdatascience.com/6-amateur-mistakes-ive-made-working-with-train-test-splits-916fabb421bb
https://repl.it/@maxbox/machinelearning4#main.py
https://maxbox.sourceforge.io/
https://linuxschweizag.wordpress.com/
https://maxkleiner1.medium.com/
https://ideone.com/hQ3zAd
https://appdb.winehq.org/objectManager.php?sClass=version&iId=39678
https://bitbucket.org/maxbox4/maxbox4/downloads/
https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
https://maxbox4.blogspot.com/
https://wiki.freepascal.org/Python4Delphi
https://www.clevercomponents.com/articles/article052/
https://entwickler-konferenz.de/delphi-innovations-fundamentals/vcl4python/

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.
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

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.
Association Rules uncover the relationship between two **or** more attributes. **It is** mainly **in** the form **of**- **If** antecedent than consequent.
Picture Exhibition 2021
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"

Doc :
AssignFile versus TStringlist

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**.
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*

```
                 ___  _____.
  _____ _____   \   \/  /\_ |__   _______ ___
 /     \\__  \  \     /  | __ \ /  _  \  \/  /
|  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?!::::::::     `.
!......YBBBBBBBBBBBBBBBBBBBRPFT?!::::::;:!^"`;:::       `.
!........YBBBBBBBBBBBRPFT?!::::::::::^''...::::::;         iBBbo.
`..........YBRPFT?!::::::::::::::::::::::::;iof68bo.      WBBBBbo.
  `..........::::::::::::::::::::::::;iof688888888888b.     `YBBBP^'
    `.......::::::::::::::::;iof68888888888888888888b.     `
      `.....::::::::::;iof688888888888888888888888888888b.
        `....:::;iof688888888888888888888888888888888899fT!
          `..::!8888888888888888888888888888888899fT||!^"'
            ` !!988888888888888888888888888899fT||!^"'
                `!!8888888888888888899fT||!^"'
                  `!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.

Hybrid Analysis <noreply@hybrid-analysis.com>

7:27 PM (1 hour ago)

to
Congratulations! Your analysis **is** done **and** available at: https:*//www.hybrid-analysis.com/sample/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3?environmentId=160*

--- Falcon Sandbox Analysis Overview ---

https:*//www.hybrid-analysis.com/sample/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3*

--- Falcon Sandbox Analysis Summary ---

File Name: maXbox4.exe
Analysis State: SUCCESS
Threat Verdict: no specific threat
Threat Score: n/a/100
AV Detection Ratio: n/a
AV Family Name:
Time **of** analysis: 2022-12-15 18:16:53
File Size (bytes): 34057496
File Type: PE32 executable (GUI) Intel 80386, **for** MS Windows
Contacted Domains: none
Contacted Hosts: none
Environment: Windows 10 64 bit (ID: 160)

```
                    ----
                   /~@@~\,
 _______ . _\_\___/\ __ /\___|_|_ . _______
/ ____  |=|      \  <_+>  /       |=|  ____ \
~|    |\|=|======\\_______//======|=|/|    |~
 |_   |    \       |       |       /    |    |
  \==-|     \      |  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.
      ./?HMMMMMMMMMMM"*"""            &MHb.
     /'|MMMMMMMMMMM'              -    `*MHM\
    /  |MMM'MMHHM''                     .MMMHb
   |   9HMMP   .Hq,                     TMM'MMH
  /     |MM\,H-""&&6\__                 `MMMMMMb
 |       `""HH#,        \              - MMMMMMM|
 |           `HoodHMM###.                `9MMMMMH
 |              .MMMMMMMM##\               `*"?HM
 |          ..  ,HMMMMMMMMMMMMo\.               |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());
```

Feature update **to** Windows 10, version 22H2
Target Machine Intel 386 **or** later processors **and** compatible processors
Compilation Timestamp 2022-12-15 14:45:59 UTC Signing time 15 Dec 2022 15:50:51
Entry Point 24930856 - Contained Sections 10
SHA1: 4.7.6.20 BF042A1DC1C479C706100BB83F8DEE351554F74B

https://maxbox4.wordpress.com/2022/11/17/data-science-story/

```
39933:
39934: Teach and Train it with the Jupyter Notebook:
39935: Tested on
39936:    Delphi 10.4 Community Edition Update 2
39937:    Lazarus V 2.2.4 for Windows 64 bit
39938:    maXbox4 Version 4.7.6.20
39939:    PyScripter 4.1.1.0 x64
39940:    SVGImage32Package270.bpl V 3.9.4
39941:
39942: MD5 214cfe8c9c60dd12bb65b9b30e51b1c2
39943: SHA-1 bf042a1dc1c479c706100bb83f8dee351554f74b
39944: SHA-256 102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3
39945: Vhash 0370a6666d5c0d551c051031e0900e5003116012z33034z17013109054843115z10
39946: Authentihash a83b3186965f1e3001f870d90522ce98650e4a0435d12f02d7db385f5d940555
39947: Imphash da442fa7c70f4ee112e5300f1dacb062
39948: SSDEEP
       393216:bWouExdbycZbvhe19EGxdtzUROg/YC4+eOIBpzbloJfHMXjZLkxbS5r/vrEbCK9u:zuExdby8c6AC+pzbloJ0TTr325mfow
39949: TLSH T10D778C62B1C0D632D05705788C0FD6E85A6A3D206DB4949B79F6BF8E2E311617F362CB
39950: File type Win32 EXE
39951: Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
39952: TrID Windows ActiveX control (60.1%)   InstallShield setup (22.2%)   Win64 Executable (generic) (5.4%)
       DOS Borland compiled Executable (generic) (5.1%)   Win32 Executable (generic) (2.3%)
39953: DetectItEasy PE32   Compiler: Borland Delphi (2006)   Linker: Turbo Linker (2.25*,Delphi) [GUI32,signed]
39954: File size 32.48 MB (34057496 bytes)
39955:
39956: https://github.com/maxkleiner/Bayes_theorem/blob/master/clusterclass4penguins.ipynb
39957: https://github.com/maxkleiner/maXbox4/blob/master/clusterclass4penguins.ipynb
39958: https://github.com/maxkleiner/maXbox4/releases/download/V4.2.4.80/maXbox4.exe
39959: https://my6.code.blog/2022/09/02/webpostdata/
39960:
39961: Songs for maXbox
39962: 1. Beat the Blues /2. Play Chess /3. Cary Copper (White magic) /4. Eat Garlic (NowAge)
39963: 5. Exerce de Meditation /6. Practise running (Japan) /7. Bett von Nord nach Süd - Zeitraum
39964: 8. Olemos Bruja /9. Listen white noise (Lord of Darkness)
39965:
39966: https://sourceforge.net/projects/maxbox/files/maXbox_Archive/Improver/Improver_4_Edition_2021.zip/download
39967:
39968:
```

maXbox 4exe

Digital unterschrieben von maXbox4exe
Datum: 2022.12.16 09:03:46 +01'00'