

```
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:34,04 MB(35,697,944) V4.7.6.50 June.2023 EKON26/BASTA/JAX/IBZ/SWS/PASCON
9: *****Now the Funclist*****
10: Funclist Func : 22257; Source Compiled: 2'844'308 lines
11: *****Now the Proclist*****
12: Proclist Proc Size is: 13051_
13: *****Now Constructors*****
14: Constructlist Constructor Size is: 2052_
15: def head: max: maXbox10: 15/06/2023 11:42:52
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: 373372! Constructor, Func and Procedure
22: AExtraxt of EXE Functions of maXbox4.exe, locs of file = 42720
23: ASize of EXE: 35,697,944 bytes CRC32: 38562FA8
24: SHA1: of maXbox4.exe (4.7.6.50) D047DBD5412C3E4A436089018B9C7FACF17A2EB5
25: SHA-256 193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7
26:
27: https://www.hybrid-analysis.com/sample/193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7
28: https://www.virustotal.com/gui/file/193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7/details
29: https://www.hybrid-analysis.com/sample/cf3b5e722fc42ad6e4d795623b3a88c94523abb28e84bd77a9d644418d379c97
30: https://www.virustotal.com/gui/file/cf3b5e722fc42ad6e4d795623b3a88c94523abb28e84bd77a9d644418d379c97/detection
31: https://www.hybrid-analysis.com/sample/fa0f30abf34292e91070a5bd4682040eb6af79a8f6c7f55111c9692153120988
32: https://www.virustotal.com/gui/file/fa0f30abf34292e91070a5bd4682040eb6af79a8f6c7f55111c9692153120988/detection
33: https://www.hybrid-analysis.com/sample/55a86c35b1d497c3a5d860da04734c791d5affa68b7e5c5461b7032d6a325c14
34: https://www.virustotal.com/gui/file/55a86c35b1d497c3a5d860da04734c791d5affa68b7e5c5461b7032d6a325c14/detection
35: https://www.hybrid-analysis.com/sample/882ddadb9b330318bfff73d8db66d1a33be575be4cedec3960374bfeb7c49c968
36: https://www.virustotal.com/gui/file/882ddadb9b330318bfff73d8db66d1a33be575be4cedec3960374bfeb7c49c968/detection
37: https://www.virustotal.com/gui/file/42c6b4dc932473f10a67708c141807acf3f832f7ba66005b495d41bf1653b019/detection
38: https://www.hybrid-analysis.com/sample/1658eb368e4e8621c51d8e568de40a12be00ad02c63dfe7a3b655603661dc4f9
39: https://www.virustotal.com/gui/file/1658eb368e4e8621c51d8e568de40a12be00ad02c63dfe7a3b655603661dc4f9/detection
40: https://www.hybrid-analysis.com/sample/4dfbada6765e47c72b7c2496f831419b3461fa7c4ac6e05e2a941b501e10e022
41: https://metadefender.opswat.com/results/file/bzIzMDUzMXUxb2NqQ2VfVTNmWF3c0JwNGF/regular/multiscan
42: https://www.virustotal.com/gui/file/4dfbada6765e47c72b7c2496f831419b3461fa7c4ac6e05e2a941b501e10e022/detection
43: https://www.virustotal.com/gui/file/168f85cff6ecdffcc4e7614758bbdb333a38f4896f94b7056f3e53af8fd15a66b/detection
44: https://www.virustotal.com/gui/file/222ee48c3409cbf176ffe9037d7958952e6754549c66d8397e927ad68ee3be17/detection
45: https://www.hybrid-analysis.com/sample/c9f275808708fa8bbe56f816a6ae90f2438dd8a7085ecf8e4bbcff1e1747571
46: https://www.virustotal.com/gui/file/c9f275808708fa8bbe56f816a6ae90f2438dd8a7085ecf8e4bbcff1e1747571/detection
47: https://www.hybrid-analysis.com/sample/ca5b41a709e61c1174d5105d1edc2759e01bd3c4a57a30b54dad568e104cac77
48: https://www.virustotal.com/gui/file/ca5b41a709e61c1174d5105d1edc2759e01bd3c4a57a30b54dad568e104cac77/details
49: https://www.virustotal.com/gui/file/9347258d5985a2fe88a1df45c3cd99747babbbaa034de3e1725139684a7b6e1c7/detection
50: https://www.hybrid-analysis.com/sample/97943841d5908ea0846dc6763b14941350a0c1aa5d2b6248bfc79ffad7a0314e
51: https://www.virustotal.com/gui/file/97943841d5908ea0846dc6763b14941350a0c1aa5d2b6248bfc79ffad7a0314e/detection
52: https://www.hybrid-analysis.com/sample/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3
53: https://www.virustotal.com/gui/file/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3/detection
54: https://www.hybrid-analysis.com/sample/66e02bfc560c5082c2bfc5e4b8e2a7b19fal60386474097625d55c7cc6b472
55: https://www.virustotal.com/gui/file/66e02bfc560c5082c2bfc5e4b8e2a7b19fal60386474097625d55c7cc6b472/detection
56: https://www.virustotal.com/gui/file/2acfa55e4bb3ae08a46cf2a691c8f62a6404d84db3ae6fb4bb3f7042149bfb73/detection
57: https://www.virustotal.com/gui/file/212f8fda437e877c7f89f771ce40e57814cfb4380ee010c090df78d8dbbc4d9f/detection
58: https://www.virustotal.com/gui/file/4f4219830b728537ef6c4ddec92c2fbd8f6658a2f03c3981fdfcc833ae8c1e/detection
59: https://www.virustotal.com/gui/file/c6e8c0861b0a05cb26ebdaf6985983eb8f6b4d92fb99ad2fa62d2ba50bed/detection
60: https://www.virustotal.com/gui/file/5d231d7b91a8e1d79ee2b1f19279a0dd2685c7c99974b0da9df749347fc58/detection
61: https://www.virustotal.com/gui/file/09eb95c4cc3b7fe60772405b170af4705085da89d190213228c65313c8733499/details
62: https://www.virustotal.com/gui/file/e35b4b6cde46d1c0c8a0c30bc125d76d6738e9d0541523214c12d0aa1ba91ca9/details
63: https://www.virustotal.com/gui/file/1a2ed2b64f9b7d68e1a16158d06742ee1f1cde94864a50ce29f08bfadaf402a44/detection
64: https://www.virustotal.com/gui/file/ec6aeb784bdf2dec91834378ef46be698cc0d0edda9e6488c7fcafac7728/detection
65: https://www.virustotal.com/gui/file/9b4ce8468d70b11a0b111009ccf9b4a9e1daf6f9ec53680bf301256e5f69518a/detection
66: https://www.virustotal.com/gui/file/db0c72990863b4a08341410b51d76a00632519c0f1df9cc4d63a40c28f2c9778/detection
```

```

67: https://www.virustotal.com/gui/file/52a07df33308c2ded182c50194a4c731f9921461ba1cf26625005ec0db672/detection
68: https://www.virustotal.com/gui/file/e99be9470f26f69169cc81dfbd8984c7a27dd3fb4b16941cb4028e5dbba66957/detection
69: //////////////////////////////////////
70:
71: Func Metric of Script: 256_findfunctions476_of_EXE_80_1.txt
72: Func *****Now the Funclist*****
73: Func GetResStringChecked(Ident:Str; const Args: array of const):str
74: Func ( Index : Longint) : Integer
75: Func (Command: Word; Data: Longint; var CallHelp:Bool):Bool
76: Func _CheckAutoResult( ResultCode : HRESULT) : HRESULT
77: Func _T(Name: tbtString): Variant;
78: Func ABNFToText(const AText :Str) :Str
79: Func Abs(e : Extended) : Extended;
80: Func AbsInt(e : Extended) : Integer;
81: Func AbsInt( const B : integer) : integer;
82: Func AbsFloat( const B : double) : extended;
83: Func Ackermann( const A, B : Integer) : Integer
84: Func AcquireLayoutLock :Bool
85: Func ActionByName( const AName :Str) : TWebActionItem
86: Func ACTIVEBUFFER : PCHAR
87: Func Add:TAggregate
88: Func Add:TCollectionItem
89: Func Add:TColumn
90: Func Add:TComboExItem
91: Func Add:TCookie
92: Func Add:TCoolBand
93: Func Add:TFavoriteLinkItem
94: Func Add:TFileTypeItem
95: Func Add:THeaderSection
96: Func Add:THTMLTableColumn
97: Func Add:TIdEmailAddressItem
98: Func Add:TIdMessagePart
99: Func Add:TidUserAccount
100: Func Add:TListColumn
101: Func Add:TListItem
102: Func Add:TStatusPanel
103: Func Add:TTaskDialogBaseButtonItem
104: Func Add:TWebActionItem
105: Func Add:TWorkArea
106: Func Add(AClass : TClass) : Integer
107: Func Add(AComponent : TComponent) : Integer
108: Func Add(AItem, AData : Integer) : Integer
109: Func Add(AItem, AData : Pointer) : Pointer
110: Func Add(AItem, AData : TObject) : TObject
111: Func Add(AObject : TObject) : Integer
112: Func Add(const Access, Count :Card; const Offset : Int64) : Integer
113: Func Add(const S : WideString) : Int
114: Func Add(Image, Mask : TBitmap) : Int
115: Func Add(Index : LongInt; const Text :Str) : LongInt
116: Func Add(Sibling : TTreeNode; const S :Str) : TTreeNode
117: Func Add(const S:Str): Int
118: Func Add(S:Str): Int;
119: Func AddAt(const Access, Count:Card;const Offset: Int64; const Address:Pointer): Int
120: Func ADDCHILD : TFIELDDEF
121: Func AddChild( Index : LongInt; const Text :Str) : LongInt
122: Func AddChild( Parent : TTreeNode; const S :Str) : TTreeNode
123: Func AddChildFirst( Parent : TTreeNode; const S :Str) : TTreeNode
124: Func AddChildObject(Index:LongInt;const Text:str;const Data:Pointer:LongInt
125: Func AddChildObject( Parent:TTreeNode;const S:str;Ptr:Pointer) : TTreeNode
126: Func AddChildObjectFirst(Parent:TTreeNode;const S:str;Ptr:Pointer):TTreeNode
127: Func ADDFIELDDEF : TFIELDDEF
128: Func AddFileExtIfNecessary( AFileName, AExt :Str) :Str
129: Func AddFirst( Sibling : TTreeNode; const S :Str) : TTreeNode
130: Func AddIcon( Image : TIcon) : Int
131: Func AddImage( Value : TCustomImageList; Index : Int) : Int
132: Func ADDINDEXDEF : TINDEXDEF
133: Func AddItem(const Caption:str;const ImageIdx,SelectImageIdx,OverlayImagIdx,
Indent: Int;Data:Ptr):TComboExItem
134: Func AddItem( Item : THeaderSection; Index : Int) : THeaderSection
135: Func AddItem( Item : TListItem; Index : Int) : TListItem
136: Func AddItem( Item : TStatusPanel; Index : Int) : TStatusPanel
137: Func AddMapping( const FieldName :Str) :Bool
138: Func AddMasked( Image : TBitmap; MaskColor : TColor) : Int
139: Func AddModuleClass( AClass : TComponentClass) : TComponent
140: Func AddModuleName( const AClass :Str) : TComponent
141: Func AddNode(Node,Relative:TTreeNode;const S:str;Ptr:Pointer;Method:TNodeAttachMode):TTreeNode
142: Func AddObject( const S : WideString; AObject : TObject) : Int
143: Func AddObject(Index:LongInt;const Text:str; const Data:Pointer): LongInt
144: Func AddObject( Sibling : TTreeNode; const S :Str;Ptr: Pointer): TTreeNode
145: Func AddObject(S:str;AObject:TObject):Int
146: Func AddObjectFirst(Sibling:TTreeNode;const S string;Ptr:Pointer):TTreeNode
147: Func AddParameter : TParameter
148: Func AddParamsSQLForDetail(Params:TParams;SQL:WideStr;Native:Bool;QuoteChar:WideString):WideString
149: Func Addr64ToAddr32(const Value: TAddr64): TAddr32;
150: Func Addr32ToAddr64(const Value: TAddr32): TAddr64;
151: Func AdjustLineBreaksS(const S:Str):Str TTextLineBreakStyle', '(tlbsLF, tlbsCRLF)
152: Func AdjustLineBreaks(const S:Str; Style: TTextLineBreakStyle):Str;
153: Func AllData :Str

```

```
154: Func AllocMemCount: Int;
155: Func AllocMemSize: Int;
156: Func AllocPatternBitmap( BkColor, FgColor : TColor ) : TBitmap
157: Func AllowRegKeyForEveryone( Key : HKEY; Path :Str) :Bool
158: Func AlphaComponent( const Color32 : TColor32) : Int
159: Func AlphaSort :Bool
160: Func AlphaSort( ARecurse :Bool) :Bool
161: Func AnsiCat( const x, y : Ansistr) : Ansistr
162: Func AnsiCompareFileName( S1, S2 :Str) : Int
163: Func AnsiCompareFileName(const S1:Str; const S2:Str): Int)
164: Func AnsiCompareStr( S1, S2 :Str) : Int
165: Func AnsiCompareStr(const S1:Str; const S2:Str): Int;
166: Func AnsiCompareText( S1, S2 :Str) : Int
167: Func AnsiCompareText(const S1:Str; const S2:Str): Int;
168: Func AnsiContainsStr( const AText, ASubText :Str) :Bool
169: Func AnsiContainsText( const AText, ASubText :Str) :Bool
170: Func AnsiCopy( const src : Ansistr; index, count : Int) : Ansistr
171: Func AnsiDequotedStr( S :Str; AQuote : Char) :Str
172: Func AnsiEndsStr( const ASubText, AText :Str) :Bool
173: Func AnsiEndsText( const ASubText, AText :Str) :Bool
174: Func AnsiExtractQuotedStr( var Src : PChar; Quote : Char) :Str
175: Func AnsiExtractQuotedStr(var Src: PChar; Quote: Char):Str)
176: Func AnsiIndexStr( const AText :Str; const AValues : array of string) : Int
177: Func AnsiIndexText(const AText:str; const AValues:array of string):Int
178: Func AnsiLastChar( S :Str) : PChar
179: Func AnsiLastChar(const S:Str): PChar)
180: Func AnsiLeftStr( const AText : Ansistr; const ACount : Int) : Ansistr
181: Func AnsiLowerCase( S :Str) :Str
182: Func AnsiLowercase(s :Str) :Str;
183: Func AnsiLowerCaseFileName( S :Str) :Str
184: Func AnsiMatchStr( const AText:str; const AValues:array of string):Bool
185: Func AnsiMatchText(const AText:str; const AValues: array of string):Bool
186: Func AnsiMidStr(const AText: Ansistr; const AStart, ACount: Int): Ansistr
187: Func AnsiPos( const src, sub : Ansistr) : Int
188: Func AnsiPos( Substr, S :Str) : Int
189: Func AnsiPos(const Substr:Str; const S:Str): Int;
190: Func AnsiQuotedStr( S :Str; Quote : Char) :Str
191: Func AnsiReplaceStr( const AText, AFromText, AToText :Str) :Str
192: Func AnsiReplaceText( const AText, AFromText, AToText :Str) :Str
193: Func AnsiResemblesText( const AText, AOther :Str) :Bool
194: Func AnsiReverseString( const AText : Ansistr) : Ansistr
195: Func AnsiRightStr( const AText : Ansistr; const ACount : Int) : Ansistr
196: Func AnsiSameCaption(const Text1:Str; const Text2:Str):Bool)
197: Func AnsiSameStr( S1, S2 :Str) :Bool
198: Func AnsiSameStr(const S1:Str; const S2:Str):Bool)
199: Func AnsiSameText( const S1, S2 :Str) :Bool
200: Func AnsiSameText( S1, S2 :Str) :Bool
201: Func AnsiSameText(const S1:Str; const S2:Str):Bool)
202: Func AnsiStartsStr( const ASubText, AText :Str) :Bool
203: Func AnsiStartsText( const ASubText, AText :Str) :Bool
204: Func AnsiStrComp( S1, S2 : PChar) : Int
205: Func AnsiStrComp(S1: PChar; S2: PChar): Int)
206: Func AnsiStrIComp( S1, S2 : PChar) : Int
207: Func AnsiStrIComp(S1: PChar; S2: PChar): Int)
208: Func AnsiStrLastChar( P : PChar) : PChar
209: Func AnsiStrLastChar(P: PChar): PChar)
210: Func AnsiStrLComp( S1, S2 : PChar; MaxLen :Card) : Int
211: Func AnsiStrLIComp( S1, S2 : PChar; MaxLen :Card) : Int
212: Func AnsiStrLower( Str : PChar) : PChar
213: Func AnsiStrPos( Str, SubStr : PChar) : PChar
214: Func AnsiStrPos(Str: PChar; SubStr: PChar): PChar)
215: Func AnsiStrScan(Str: PChar; Chr: Char): PChar)
216: Func AnsiStrUpper( Str : PChar) : PChar
217: Func AnsiToUtf8( const S :Str) : UTF8String
218: Func AnsiToUtf8Ex( const S :Str; const cp : Int) : UTF8String
219: Func AnsiUpperCase( S :Str) :Str
220: Func AnsiUpperCase(s :Str) :Str;
221: Func AnsiUpperCaseFileName( S :Str) :Str
222: Func ApplyUpdates(const Delta:OleVariant;MaxErrors:Int;out ErrorCount:Int): OleVariant
223: Func ApplyUpdates(const Delta:OleVariant;MaxErrors: Int;out ErrorCount:Int): OleVariant;
224: Func ApplyUpdates( MaxErrors : Int) : Int
225: Func ApplyUpdates1(const Delta:OleVar;MaxErrs:Int;out ErrCount:Int;var OwnerData:OleVar):OleVariant;
226: Func ArcCos(const X : Extended) : Extended
227: Func ArcCosh(const X : Extended) : Extended
228: Func ArcCot(const X : Extended) : Extended
229: Func ArcCotH(const X : Extended) : Extended
230: Func ArcCsc(const X : Extended) : Extended
231: Func ArcCscH(const X : Extended) : Extended
232: Func ArcSec(const X : Extended) : Extended
233: Func ArcSecH(const X : Extended) : Extended
234: Func ArcSin(const X : Extended) : Extended
235: Func ArcSinh(const X : Extended) : Extended
236: Func ArcTan(const X : Extended) : Extended
237: Func ArcTan2(const Y, X : Extended) : Extended
238: Func ArithmeticMean(const X : TDynDoubleArray) : Float
239: Func ArrayLength: Int;
240: Func AsHex( const AValue : T4x4LongWordRecord):Str
241: Func AsHex( const AValue : T5x4LongWordRecord):Str
242: Func ASNDLen( var Start : Int; const Buffer :Str) : Int
```

```

243: Func ASNDecOIDItem( var Start : Int; const Buffer :Str) : Int
244: Func ASNEncInt( Value : Int) :Str
245: Func ASNEncLen( Len : Int) :Str
246: Func ASNEncOIDItem( Value : Int) :Str
247: Func ASNEncUInt( Value : Int) :Str
248: Func ASNItem(var Start: Int; const Buffer:str; var ValueType:Int):str
249: Func ASNObject( const Data :Str; ASNTyp : Int) :Str
250: Func Assigned(I: Longint):Bool;
251: Func AspectRatio(aWidth, aHeight: Int):Str;
252: Func AsWideString( Field : TField) : WideString
253: Func AtLeast( ACount : Int) :Bool
254: Func AttemptToUseSharedMemoryManager :Bool
255: Func Authenticate :Bool
256: Func AuthenticateUser( const AUsername, APassWord :Str) :Bool
257: Func Authentication :Str
258: Func BatchMove( ASource : TBDEDataSet; AMode : TBatchMode) : Longint
259: Func BcdCompare( const bcd1, bcd2 : TBcd) : Int
260: Func BcdFromBytes( const AValue : TBytes) : TBcd
261: Func BcdPrecision( const Bcd : TBcd) : Word
262: Func BcdScale( const Bcd : TBcd) : Word
263: Func BcdToBytes( const Value : TBcd) : TBytes
264: Func BCDToCurr( const BCD : TBcd; var Curr : Currency) :Bool
265: Func BcdToDouble( const Bcd : TBcd) : Double
266: Func BcdToInt( const Bcd : TBcd; Truncate :Bool) : Int
267: Func BcdToStr( const Bcd : TBcd) :Str;
268: Func BcdToStrF(const Bcd:TBcd;Format:TFloatFormat;const Precision,Digits:Int):str
269: Func beep2(dwFreq, dwDuration: Int):Bool;
270: Func BeginPeriod( const Period :Card) :Bool
271: Func BeginTrans : Int
272: Func BeginTransaction : TDBXTransaction;
273: Func BeginTransaction1( Isolation : TDBXIsolation) : TDBXTransaction;
274: Func BigMulu(aone, atwo:Str):Str;
275: Func BigNumber(aone, atwo:Str):Str;
276: Func BigExp(aone,atwo:Str):Str;
277: Func BigPow(aone, atwo: Int):Str;
278: Func BigMul(aone,atwo:Str):Str;
279: Func BigAdd(aone,atwo:Str):Str;
280: Func BigSub(aone,atwo:Str):Str;
281: Func BigLog(atwo:String; sig: integer):String;
282: Func BigFactorial(aone:Str):Str;
283: Func BinaryToDouble( ABinary :Str; DefValue : Double): Double
284: Func BinomialCoeff( N, R :Card) : Float
285: Func BinomialCoefficient(n, k: Int):Str;
286: Func BinStrToInt( const ABinary :Str) : Int
287: Func BinToByte(Binary:Str): Byte;
288: Func BinToHex2(Binary:Str):Str;
289: Func BinToInt(Binary:Str): Int;
290: Func BinToChar(St:Str): Char;
291: Func BinToStr(ans:Str):Str;
292: Func BitBlt(hdcDest:HDC;nXDest,nYDest,nWidth,nHeigh:Int;hdcSrc:HDC;nXSrc,nYSrc:Int;dwRop:DWORD):Bool;
293: Func BitmapsAreIdentical( ABitmap1, ABitmap2 : TBitmap) :Bool
294: Func BitsHighest( X: Byte) : Int;
295: Func BitsHighest1(X: ShortInt) : Int;
296: Func BitsHighest2(X: SmallInt) : Int;
297: Func BitsHighest3(X: Word) : Int;
298: Func BitsHighest4(X: Int) : Int;
299: Func BitsHighest5(X:Card) : Int;
300: Func BitsHighest6(X: Int64) : Int;
301: Func BitsLowest( X: Byte) : Int;
302: Func BitsLowest1(X: Shortint) : Int;
303: Func BitsLowest2(X: Smallint) : Int;
304: Func BitsLowest3(X: Word) : Int;
305: Func BitsLowest4(X:Card) : Int;
306: Func BitsLowest5(X: Int) : Int;
307: Func BitsLowest6(X: Int64) : Int;
308: Func BitsNeeded( const X : Byte) : Int;
309: Func BitsNeeded1(const X : Word) : Int;
310: Func BitsNeeded2(const X : Int) : Int;
311: Func BitsNeeded3(const X : Int64) : Int;
312: Func BlueComponent(const Color32 : TColor32) : Int
313: Func BooleanToInt( const Pb :Bool) : Int
314: Func BoolToStr(B:Bool; UseBoolStrs:Bool):Str)
315: Func BoolToStr1(value :Bool) :Str;
316: Func booltoint( aBool :Bool) : LongInt
317: Func inttobool( aInt : LongInt) :Bool
318: Func Bounds( ALeft, ATop, AWidth, AHeight : Int) : TRect
319: Func Bounds(ALeft, ATop, AWidth, AHeight: Int): TRect)
320: Func BreakApart(BaseString,BreakString:str;StringList: TStrings): TStrings
321: Func BrightColor( const Color : TColor; const Pct : Single) : TColor
322: Func BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
323: Func BufferRequest( Length : Int) : TStream
324: Func BuildFileList( const Path:Str;const Attr:Int;const List:TStrings):Bool
325: Func Buttons: PTaskDialogButton
326: Func BytesPerScanline( PixelsPerScanline,BitsPerPixel,Alignment:Longint):Longint
327: Func BytesToCardinal( const AValue : TIdBytes; const AIndex : Int) :Card
328: Func BytesToChar( const AValue : TIdBytes; const AIndex : Int) : Char
329: Func BytesToInt64( const AValue : TIdBytes; const AIndex : Int) : Int64
330: Func BytesToInt( const AValue : TIdBytes; const AIndex : Int) : Int
331: Func BytesToIPv6( const AValue : TIdBytes; const AIndex : Int) : TIdIPv6Address

```



```

332: Func BytesToShort( const AValue : TIdBytes; const AIndex : Int) : Short
333: Func BytesToString(ABytes:TIdBytes; AStartIndex:Int; AMaxCount:Int):Str;
334: Func BytesToStr(const Value: TBytes):Str;
335: Func BytesToWord( const AValue : TIdBytes; const AIndex : Int) : Word
336: Func ByteToBin(Int: Byte):Str;
337: Func ByteToCharIndex( S :Str; Index : Int) : Int
338: Func ByteToCharIndex(const S:Str; Index: Int): Int)
339: Func ByteToCharLen( S :Str; MaxLen : Int) : Int
340: Func ByteToCharLen(const S:Str; MaxLen: Int): Int)
341: Func ByteToHex( const AByte : Byte) :Str
342: Func ByteToOctal( const AByte : Byte) :Str
343: Func ByteType( S :Str; Index : Int) : TMbcsByteType
344: Func ByteType(const S:Str; Index: Int): TMbcsByteType)
345: Func CalcTitleRect( Col : TColumn; ARow : Int; var MasterCol:TColumn):TRect
346: Func CalculateDFAFingerprint( oStates : TList) : Int
347: Func CallTerminateProcs:Bool)
348: Func CANFOCUS:BOOLEAN
349: Func CanLoad( const Ext :Str) :Bool
350: Func CanParse( AWebRequest : TWebRequest) :Bool
351: Func CanSave( const Ext :Str) :Bool
352: Func CanStart( cChar : char) :Bool
353: Func CaptureScreen : TBitmap;
354: Func CaptureScreen1( Rec : TRect) : TBitmap;
355: Func CardinalToFourChar( ACardinal : LongWord) :Str
356: Func CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Bool):Bool
357: Func CastSoapToVariant1(SoapInfo:PTypeInfo;const SoapData:WideString):Variant;
358: Func Ceil( const X : Extended) : Int
359: Func Ceil16( X : Int) : Int
360: Func Ceil4( X : Int) : Int
361: Func Ceil8( X : Int) : Int
362: Func Ceiling( const X : Extended) : Int
363: Func CellRect( ACol, ARow : Longint) : TRect
364: Func CelsiusToFahrenheit( const AValue : Double) : Double
365: Func CenterPoint( const Rect : TRect) : TPoint
366: Func CenterPoint(const Rect: TRect): TPoint)
367: Func ChangeFileExt( FileName, Extension :Str) :Str
368: Func ChangeFileExt(const FileName:Str; const Extension:Str):Str)
369: Func CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) :Bool
370: Func CharInSet3( C : Char; const CharSet : TSysCharSet) :Bool
371: Function CharInSet4(C: Char; const CharSet: CharSet):Bool;
372: Func CharInSet( const Ch : Char; const testSet: TSysCharSet):Bool
373: Func CharIsInEOF( const AString :Str; ACharPos : Int) :Bool
374: Func CharIsInSet(const AString:str;const ACharPos:Int;const ASet:str):Bool
375: Func CharLength( S :Str; Index : Int) : Int
376: Func CharRange( const AMin, AMax : Char) :Str
377: Func CharSetToIdent(Charset: Longint; var Ident:Str):Bool)
378: Func CharToBin(vChr: Char):Str;
379: Func CharNext(lpsz: PChar): PChar; stdcall;
380: Func CharToByteIndex( S :Str; Index : Int) : Int
381: Func CharToByteIndex(const S:Str; Index: Int): Int)
382: Func CharToByteLen( S :Str; MaxLen : Int) : Int
383: Func CharToByteLen(const S:Str; MaxLen: Int): Int)
384: Func CharToHex(const APrefix :Str; const cc : Char) : shortstring;
385: Func CharToHexStr(Value: char):Str;
386: Func CharToOem(ins, outs: PChar):boolean;
387: Func CharToUnicode(Value: Char):Str;
388: Func CheckMenuDropdown :Bool
389: Func CheckMessages : longint
390: Func CheckBox:Str;
391: Func CheckOpen( Status : DBIResult) :Bool
392: Func CheckPassword( const APassword :Str) :Bool
393: Func CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt):SmallInt
394: Func CheckCrc32( var X : array of Byte; N : Int; Crc :Card) : Int;
395: Func CheckSynchronize(Timeout: Int):Bool
396: Func CheckWin32Version( AMajor : Int; AMinor : Int) :Bool
397: Func CheckCom(AComNumber: Int): Int;;
398: Func CheckLPT1:Str;;
399: Func ChrA(const a: byte): char;
400: Func ClassIDToProgID(const ClassID: TGUID):Str;
401: Func ClassNameIs(const Name:Str):Bool
402: Func ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
403: Func ClearBit1(const Value : Shortint; const Bit : TBitRange) : Shortint;
404: Func ClearBit2(const Value : Smallint; const Bit : TBitRange) : Smallint;
405: Func ClearBit3(const Value : Word; const Bit : TBitRange) : Word;
406: Func ClearBit4(const Value : Int; const Bit : TBitRange) : Int;
407: Func ClearBit5(const Value :Card; const Bit : TBitRange) :Card;
408: Func ClearBit6(const Value : Int64; const Bit : TBitRange) : Int64;
409: Func CLIENTTOSCREEN(POINT:TPOINT):TPOINT
410: Func Clipboard : TClipboard
411: Func ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
412: Func ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
413: Func ClipLine( var X1, Y1, X2, Y2 : Int; const ClipRect : TRect) :Bool;
414: Func ClipLineToRect(var P1,P2:TFloatPoint;const Rect:TFloatRect) :Bool
415: Func Clone( out stm : IStream) : HResult
416: Func CloneConnection : TSQLConnection
417: Func CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
418: Func CLOSEQUERY:BOOLEAN
419: Func CloseVolume( var Volume : THandle) :Bool
420: Func CloseHandle(Handle: Int): Int; stdcall;

```

```

421: Func CPlApplet(hwndCPl:THandle;uMsg:DWORD;lParam1,lParam2:Longint):Longint
422: Func CmdLine: PChar;
423: Func CmdShow: Int;
424: // type TPos = (tLat, tLon); TShowFmt = (sfNautical, sfStatute, sfMetric);
425: Func CoordinateStr(Idx: Int; PosInSec: Double; PosLn: TPos):Str;
426: Func Color32(const R, G, B : Byte; const A : Byte) : TColor32;
427: Func Color32(WinColor : TColor) : TColor32;
428: Func Color321(const Index : Byte; const Palette : TPalette32) : TColor32;
429: Func ColorAdjustLuma( clrRGB : TColor; n : Int; fScale : Boolean) : TColor
430: Func ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
431: Func ColorToHTML( const Color : TColor) :Str
432: Func ColorToIdent(Color: Longint; var Ident:Str):Bool)
433: Func ColorToRGB(color: TColor): Longint
434: Func ColorToString(Color: TColor):Str)
435: Func ColorToWebColorName( Color : TColor) :Str
436: Func ColorToWebColorStr( Color : TColor) :Str
437: Func ColumnAtDepth( Col : TColumn; ADepth : Int) : TColumn
438: Func Combination(npr, ncr: Int): extended;
439: Func CombinationInt(npr, ncr: Int): Int64;
440: Func CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
441: Func CommaAdd( const AStr1, AStr2 :Str) :Str
442: Func CommercialRound( const X : Extended) : Int64
443: Func Commit( grfCommitFlags : Longint) : HRESULT
444: Func Compare( const NameExt :Str) :Bool
445: Func CompareDate(const A, B: TDateTime): TValueRelationship;
446: Func CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Int
447: Func CompareFiles(const FN1,FN2:str;Breathe:TNotifyEvent;BreathingSender:TObject):bool
448: Func CompareMemoryStreams( S1, S2 : TMemoryStream) :Bool
449: Func CompareStr( S1, S2 :Str) : Int
450: Func CompareStr(const S1:Str; const S2:Str): Int)
451: Func CompareString(const S1:Str; const S2:Str): Int)
452: Func CompareText( S1, S2 :Str) : Int
453: Func CompareText(const S1:Str; const S2:Str): Int)
454: Func CompareTextLike(cWildStr,cStr:str;const cWildChar:char;lCaseSensitive:bool):bool
455: Func CompareTime(const A, B: TDateTime): TValueRelationship;
456: Func CompareValueE(const A,B:Extended;Epsilon:Extended= 0):TValueRelationship; overload;
457: Func CompareValueD(const A, B:Double; Epsilon:Double = 0):TValueRelationship; overload;
458: Func CompareValues(const A, B: Single; Epsilon: Single = 0): TValueRelationship; overload;
459: Func CompareValueI(const A, B: Int): TValueRelationship; overload;
460: Func CompareValueI64(const A, B: Int64): TValueRelationship; overload;
461: Func CompatibleConversionType(const AType:TConvType; const AFamily : TConvFamily):Boolean
462: Func CompatibleConversionTypes( const AFrom, ATo : TConvType) :Bool
463: Func ComponentTypeToString( const ComponentType : DWORD) :Str
464: Func ComposeDateTime(Date,Time : TDateTime) : TDateTime;;
465: Func ComponentToStringProc(Component: TComponent):Str;
466: Func StringToComponentProc(Value:Str): TComponent;
467: Func CompToCurrency( Value : Comp) : Currency
468: Func CompToDouble( Value : Comp) : Double
469: Func ComputeFileCRC32(const FileName :Str) : Int;
470: Func ComputeSHA256(astr:Str; amode: char):Str //mode F:File, S:str
471: Func ComputeSHA512(astr:Str; amode: char):Str //mode F:File, S:str
472: Func ComPortSelect: Int; // Search for the first available port
473: Func Concat(s:Str):Str
474: Func ConnectAndGetAll :Str
475: Func Connected :Bool
476: Func constrain(x, a, b: Int): Int;
477: Func ConstraintCallBack(Req:DsInfoReq;var ADataSources:DataSources): DBIResult
478: Func ConstraintsDisabled :Bool
479: Func CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
480: Func ContainsState( oState : TniRegularExpressionState) :Bool
481: Func ContainsStr( const AText, ASubText :Str) :Bool
482: Func ContainsText( const AText, ASubText :Str) :Bool
483: Func ContainsTransition(oTransition:TniRegularExpressionTransition):boolean
484: Func Content:Str
485: Func ContentFromStream( Stream : TStream) :Str
486: Func ContentFromString( const S :Str) :Str
487: Func CONTROLSDISABLED:Bool
488: Func Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
489: Func Convert1(const AValue:Double;const AFrom1,AFrom2,ATo1,ATo2:TConvType):Double;
490: Func ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
491: Func ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Int) : Int
492: Func ConvertTo( const AValue : Double; const ATo : TConvType) : Double
493: Func ConvertWriteStream( Stream : TStream; Buffer: PChar; BufSize : Int) : Int
494: Func ConvFamilyToDescription( const AFamily : TConvFamily) :Str
495: Func ConvTypeToDescription( const AType : TConvType) :Str
496: Func ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
497: Func ConvTypeToFamily( const AType : TConvType) : TConvFamily;
498: Func ConvAdd(const AVal:Dbl;const AType1:TConvType;const AVal2:Dbl;const AType2,
AResultType:TConvType):Double
499: Func ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
AType2:TConvType):TValueRelationship
500: Func ConvDec( const AValue:Double;const AType,AAmountType:TConvType):Double;
501: Func ConvDec1(const AValue:Dbl;const AType:TConvType;const AAmount:Dble;const
AAmountType:TConvType):Double;
502: Func ConvDiff(const AVall:Dbl;const AType1:TConvType;const AVal2:Dble;const AType2,
AResuType:TConvType):Double
503: Func ConvInc( const AValue:Double; const AType,AAmountType:TConvType):Double;
504: Func ConvIncl(const AValue:Dbl;const AType:TConvType;const AAmount:Double;const
AAmountType:TConvType):Double;

```

```

505: Func ConvSameValue(const AValue1:Dbl;const AType1:TConvType;const AValue2:Dbl;const AType2:TConvType):Bool
506: Func ConvToStr( const AValue : Double; const AType : TConvType) :Str
507: Func ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmount : Double; const
    AAmountType : TConvType) :Bool
508: Func ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType;const AAmount:Double;const
    AAmountType: TConvType) :Bool
509: Func Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
510: Func CopyFile( Source, Dest :Str; CanOverwrite :Bool) : Bool
511: Func CopyFileEx( Source, Dest :Str; Flags : FILEOP_FLAGS) : Bool
512: Func CopyFileTo( const Source, Destination:Str) :Bool
513: Func CopyFrom(Source:TStream;Count:Int64):LongInt
514: Func CopyPalette( Palette : HPALETTE) : HPALETTE
515: Func CopyTo( Length : Int) :Str
516: Func CopyTo(stm:IStream;cb:Largeint;out cbRead:Largeint;out cbWritten:Largeint):HResult
517: Func CopyToEOF :Str
518: Func CopyToEOL :Str
519: Func Cos(e : Extended) : Extended;
520: Func Cosecant( const X : Extended) : Extended
521: Func Cot( const X : Extended) : Extended
522: Func Cotan( const X : Extended) : Extended
523: Func CotH( const X : Extended) : Extended
524: Func Count : Int
525: Func CountBitsCleared( X:Byte) : Int;
526: Func CountBitsCleared1(X:Shortint) : Int;
527: Func CountBitsCleared2(X:Smallint) : Int;
528: Func CountBitsCleared3(X:Word) : Int;
529: Func CountBitsCleared4(X:Int) : Int;
530: Func CountBitsCleared5(X:Card) : Int;
531: Func CountBitsCleared6(X:Int64) : Int;
532: Func CountBitsSet( X:Byte) : Int;
533: Func CountBitsSet1(X:Word) : Int;
534: Func CountBitsSet2(X:Smallint) : Int;
535: Func CountBitsSet3(X:ShortInt) : Int;
536: Func CountBitsSet4(X:Int) : Int;
537: Func CountBitsSet5(X:Card) : Int;
538: Func CountBitsSet6(X:Int64) : Int;
539: Func countDirfiles(const apath:Str): Int;
540: Func CountGenerations(Ancestor,Descendent: TClass): Int
541: Func Coversine( X : Float) : Float
542: Func CRC32(const fileName:Str): LongWord;
543: Func CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
544: Func CreateColumns : TDBGridColumns
545: Func CreateDataLink : TGridDataLink
546: Func CreateDir( Dir :Str) :Bool
547: Func CreateDir(const Dir:Str):Bool
548: Func CreateDOSProcessRedirected(const CommandLine,InputFile,OutputFile:str): Bool
549: Func CreateEnvironmentBlock(const Options:TEnvironmentOptions;const AdditionalVars:TStrings):PChar
550: Func CREATEFIELD(OWNER:TCOMPONENT,PARENTFIELD:TOBJECTFIELD;const FIELDNAME:str;CREATECHILDREN:BOOL):TFIELD
551: Func CreateGlobber( sFilespec :Str) : TniRegularExpression
552: Func CreateGrayMappedBmp( Handle : HBITMAP) : HBITMAP
553: Func CreateGrayMappedRes( Instance : THandle; ResName : PChar) : HBITMAP
554: Func CreateGUID(out Guid: TGUID): HResult
555: Func CreateInstance( const unkOuter: IUnknown;const iid:TGUID; out obj):HResult
556: Func CreateMappedBmp(Handle : HBITMAP; const OldColors,NewColors:array of TColor):HBITMAP
557: Func CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
558: Func CreateMessageDialog(const Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons):TForm;
559: Func CreateMessageDialog1(const Msg:str;DlgType:TMsgDlgType;Btns:TMsgDlgBtns;DefaultBtn:TMsgDlgBtn):TForm;
560: Func CreateOleObject(const ClassName:Str): IDispatch;
561: Func CREATEPARAM(FLDTYPE:TFIELDTYPE;const PARAMNAME:str; PARAMTYPE:TPARAMTYPE): TPARAM
562: Func CreateParameter(const
    Name:WideString;DataType:TDataType;Direction:TParameterDirection;Size:Int;Value: OleVariant):TParameter
563: Func CreateLocate( DataSet : TDataSet) : TJvLocateObject
564: Func CreateMappedBmp(Handle: HBITMAP;const OldColors,NewColors:array of TColor) :HBITMAP
565: Func CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
566: Func CreateMutex2(lpMutexAttributes:TObject;bInitialOwner:BOOL;lpName:PChar) : THandle;
567: Func CreateSemaphore2(lpSemaphoreAttributes:TObject;lInitialCount,
    lMaximumCount:Longint;lpName:PChar):THandle;
568: //Ex.: vMutex := CreateMutex2(nil, True, MutexName);
569: Func IPAddrToName // ShowMessage(IPAddrToName(LocalIp));
570: Func CreateRecordBuffer( Length : Int) : TRecordBuffer
571: Func CreateValueBuffer( Length : Int) : TValueBuffer
572: Func CreatePopUpCalculator(AOwner:TComponent;ABiDiMode:TBiDiMode):TWinControl
573: Func CreateRecordBuffer( Length : Int) : TRecordBuffer
574: Func CreateRotatedFont( Font : TFont; Angle : Int) : HFONT
575: Func CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
576: Func CreateValueBuffer( Length : Int) : TValueBuffer
577: Func CreateHexDump( AOwner : TWinControl) : THexDump
578: Func Csc( const X : Extended) : Extended
579: Func CscH( const X : Extended) : Extended
580: Func currencyDecimals: Byte
581: Func currencyFormat: Byte
582: Func currencyString:Str
583: Func CurrentProcessId : TidPID
584: Func CurrentReadBuffer :Str
585: Func CurrentThreadId : TidPID
586: Func CurrentYear : Word
587: Func CurrToBCD(const Curr:Currency;var BCD:TBcd;Precision:Int;Decimals:Int):Bool
588: Func CurrToStr( Value : Currency) :Str;
589: Func CurrToStrF(Value:Currency;FormatSettings:TFormatSettings;Digits:Int):str;

```

```

590: Func CurrToStrFS(Value:Currency;Format:TFloatFormat;Digits:Int;const FormatSettings:TFormatSettings):str;
591: Func CursorToIdent(cursor: Longint; var Ident:Str):Bool;
592: Func CursorToString(cursor: TCursor):Str;
593: Func CustomSort( SortProc : TLVCompare; lParam : Longint) :Bool
594: Func CustomSort(SortProc:TTVCompare;Data:Longint;ARecurse :Bool):Boolean
595: Func CycleToDeg(const Cycles : Extended) : Extended
596: Func CycleToGrad(const Cycles : Extended) : Extended
597: Func CycleToRad(const Cycles : Extended) : Extended
598: Func D2H( N : Longint; A : Byte) :Str
599: Func DarkColor(const Color : TColor; const Pct : Single) : TColor
600: Func DarkColorChannel(const Channel: Byte; const Pct : Single) : Byte
601: Func DataLinkDir :Str
602: Func DataRequest( Data : OleVariant) : OleVariant
603: Func DataRequest( Input : OleVariant) : OleVariant
604: Func DataToRawColumn( ACol : Int) : Int
605: Func Date : TDateTime
606: Func Date: TDateTime;
607: Func DateIsNull(const pdtValue : TDateTime; const pdtKind : TdtKind):Boolean
608: Func DateOf(const AValue : TDateTime) : TDateTime
609: Func DateSeparator: char;
610: Func DateTimeGMTToHttpStr(const GMTValue : TDateTime) :Str
611: Func DateTimeToFileDate( DateTime : TDateTime) : Int
612: Func DateTimeToFileDate(DateTime: TDateTime): Int;
613: Func DateTimeToGmtOffsetStr( ADateTime : TDateTime; SubGMT :Bool):Str
614: Func DateTimeToInternetStr(const Value:TDateTime;const AIsGMT:Boolean) String
615: Func DateTimeToJulianDate(const AValue : TDateTime) : Double
616: Func DateTimeToModifiedJulianDate(const AValue : TDateTime) : Double
617: Func DateTimeToStr( DateTime : TDateTime) :Str;
618: Func DateTimeToStr2(DateTime: TDateTime;FormatSettings:TFormatSettings):str;
619: Func DateTimeToTimeStamp(DateTime: TDateTime) : TTimeStamp
620: Func DateTimeToUnix(const AValue : TDateTime) : Int64
621: Func DateTimeToUnix(D: TDateTime): Int64;
622: Func DateToStr( DateTime : TDateTime) :Str;
623: Func DateToStr(const DateTime: TDateTime):Str;
624: Func DateToStr(D: TDateTime):Str;
625: Func DateToStr2(DateTime:TDateTime;FormatSettings:TFormatSettings):str;
626: Func DayOf(const AValue : TDateTime) : Word
627: Func DayOfTheMonth(const AValue : TDateTime) : Word
628: Func DayOfTheMonth(const AValue: TDateTime): Word;
629: Func DayOfTheWeek(const AValue : TDateTime) : Word
630: Func DayOfTheYear(const AValue : TDateTime) : Word
631: Func DayOfTheYear(const AValue: TDateTime): Word;
632: Func DayOfWeek( DateTime : TDateTime) : Word
633: Func DayOfWeek(const DateTime: TDateTime): Word;
634: Func DayOfWeekStr( DateTime : TDateTime) :Str
635: Func DaysBetween(const ANow, AThen : TDateTime) : Int
636: Func DaysInAMonth(const AYear, AMonth : Word) : Word
637: Func DaysInAYear(const AYear : Word) : Word
638: Func DaysInMonth(const AValue : TDateTime) : Word
639: Func DaysInYear(const AValue : TDateTime) : Word
640: Func DaySpan(const ANow, AThen : TDateTime) : Double
641: Func DBUseRightToLeftAlignment( AControl : TControl; AField : TField) :Bool
642: Func DecimalSeparator: char;
643: Func DecLimit(var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;
644: Func DecLimit1(var B:Shortint;const Limit:Shortint const Decr:Shortint):Shortint;
645: Func DecLimit2(var B:Smallint;const Limit:Smallint;const Decr:Smallint):Smallint;
646: Func DecLimit3(var B : Word; const Limit : Word; const Decr : Word) : Word;
647: Func DecLimit4(var B : Int; const Limit : Int; const Decr : Int) : Int;
648: Func DecLimit5(var B Cardinal;const Limit Cardinal;const Decr:Card):Card;
649: Func DecLimit6(var B:Int64;const Limit : Int64; const Decr : Int64) : Int64;
650: Func DecLimitClamp(var B:Byte; const Limit : Byte; const Decr : Byte) : Byte;
651: Func DecLimitClamp1(var B:Shortint;const Limit:Shortint;const Decr:Shortint): Shortint;
652: Func DecLimitClamp2(var B:Smallint;const Limit: Smallint;const Decr: Smallint): Smallint;
653: Func DecLimitClamp3(var B:Word;const Limit: Word; const Decr : Word) : Word;
654: Func DecLimitClamp4(var B : Int; const Limit : Int; const Decr : Int) : Int;
655: Func DecLimitClamp5(var B:Card;const Limit:Card; const Decr:Card) Cardinal;
656: Func DecLimitClamp6(var B: Int64; const Limit: Int64; const Decr: Int64) : Int64;
657: Func DecodeDateFully(DateTime: TDateTime; var Year,Month,Day,DOW:Word):Bool
658: Func DecodeSoundexInt( AValue : Int) :Str
659: Func DecodeSoundexWord( AValue : Word) :Str
660: Func DefaultAlignment : TAlignment
661: Func DefaultCaption :Str
662: Func DefaultColor : TColor
663: Func DefaultFont : TFont
664: Func DefaultImeMode : TImeMode
665: Func DefaultImeName : TImeName
666: Func DefaultReadOnly :Bool
667: Func DefaultWidth : Int
668: Func DegMinSecToFloat(const Degs, Mins, Secs : Float) : Float
669: Func DegToCycle(const Degrees : Extended) : Extended
670: Func DegToGrad(const Degrees : Extended) : Extended
671: Func DegToGrad(const Value : Extended) : Extended;
672: Func DegToGrad1(const Value : Double) : Double;
673: Func DegToGrad2(const Value : Single) : Single;
674: Func DegToRad(const Degrees : Extended) : Extended
675: Func DegToRad(const Value : Extended) : Extended;
676: Func DegToRad1(const Value : Double) : Double;
677: Func DegToRad2(const Value : Single) : Single;
678: Func DelChar(const pStr :Str; const pChar : Char) :Str

```

```

679: Func DelEnvironmentVar( const Name :Str) :Bool
680: Func Delete( const MsgNum : Int) :Bool
681: Func DeleteDirectory(const DirectoryName:str;MoveToRecycleBin:Boolean):Bool
682: Func DeleteFile(const FileName:Str):Bool
683: Func DeleteFileEx( FileName :Str; Flags : FILEOP_FLAGS) :Bool
684: Func DelimiterPosn( const sString :Str; const sDelimiters:Str): Int;
685: Func DelimiterPosn1(const sString:str;const sDelimiters:str;out cDelimiter:char):Int;
686: Func DelSpace( const pStr :Str) :Str
687: Func DelString( const pStr, pDelStr :Str) :Str
688: Func DelTree( const Path :Str) :Bool
689: Func Depth: Int
690: Func Description :Str
691: Func DescriptionsAvailable :Bool
692: Func DescriptionToConvFamily(const ADescription:str;out AFamily:TConvFamily): Boolean
693: Func DescriptionToConvType(const ADescription:str;out AType:TConvType): Bool;
694: Func DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:str;out AType:TConvType): Bool;
695: Func DetectUTF8Encoding( const s : UTF8String) : TEncodeType
696: Func DialogsToPixelsX( const Dialogs : Word) : Word
697: Func DialogsToPixelsY( const Dialogs : Word) : Word
698: Func Digits( const X :Card) : Int
699: Func DirectoryExists( const Name :Str) :Bool
700: Func DirectoryExists( Directory :Str) :Bool
701: Func DiskFree( Drive : Byte) : Int64
702: Func DiskFree(Drive: Byte): Int64
703: Func DiskInDrive( Drive : Char) :Bool
704: Func DiskSize( Drive : Byte) : Int64
705: Func DiskSize(Drive: Byte): Int64
706: Func DISPATCHCOMMAND( ACOMMAND : WORD):BOOLEAN
707: Func DispatchEnabled :Bool
708: Func DispatchMask : TMask
709: Func DispatchMethodType : TMethodType
710: Func DISPATCHPOPUP( AHANDLE : HMENU) :Bool
711: Func DispatchRequest(Sender:TObject;Request:TWebRequest;Response:TWebResponse): Boolean
712: Func DisplayCase( const S :Str) :Str
713: Func DisplayRect( Code : TDisplayCode) : TRect
714: Func DisplayRect( TextOnly :Bool) : TRect
715: Func DisplayStream( Stream : TStream) :Str
716: TBufferCoord', 'record Char : Int; Line : Int; end
717: TDisplayCoord', 'record Column : Int; Row : Int; end
718: Func DisplayCoord( AColumn, ARow : Int) : TDisplayCoord
719: Func BufferCoord( AChar, ALine : Int) : TBufferCoord
720: Func DomainName( const AHost :Str) :Str
721: Func DownloadFile(SourceFile, DestFile:Str):Bool; //fast!
722: Func DownloadFileOpen(SourceFile, DestFile:Str):Bool; //open process
723: Func DosPathToUnixPath( const Path :Str) :Str
724: Func DottedLineTo( const Canvas : TCanvas; const X, Y : Int) :Bool;
725: Func DoubleDecliningBalance(const Cost,Salvage:Extended;Life,Period:Int):Extended
726: Func DoubleToBcd( const AValue : Double) : TBcd;
727: Func DoubleToHex( const D : Double) :Str
728: Func DoUpdates :Bool
729: Func Dragging:Bool;
730: Func DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
731: Func DrawAnimatedRects(hwnd: HWND;idAni Int;const lprcFrom,lprcTo: TRect): BOOL
732: Func DrawEdge( hdc : HDC; var grc : TRect; edge : UINT; grfFlags : UINT) : BOOL
733: Func DrawFrameControl(DC:HDC; const Rect:TRect; uType, uState: UINT) : BOOL
734: {Works like InputQuery but displays 2edits.If PasswordChar<>#0,second edits PasswordChar is set}
735: Func DualInputQuery(const ACapt,Prpt1,Prpt2:str;var AVall,AVal2:str;PasswrChar:Char=#0):Bool;
736: Func DupeString( const AText :Str; ACount : Int) :Str
737: Func Edit :Bool
738: Func EditCaption :Bool
739: Func EditText :Bool
740: Func EditFolderList( Folders : TStrings) :Bool
741: Func EditQueryParams(DataSet:TDataSet;List: TParams; AHelpContext:THelpContext): Bool
742: Func Elapsed( const Update :Bool):Card
743: Func EnableProcessPrivilege(const Enable:Boolean; const Privilege:str):Bool
744: Func EnableThreadPrivilege(const Enable:Boolean;const Privilege:str): Bool
745: Func EncodeDate( Year, Month, Day : Word) : TDateTime
746: Func EncodeDate(Year, Month, Day : Word) : TDateTime;
747: Func EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
748: Func EncodeDateMonthWeek(const AYear,AMonth,AWeekOfMonth ADayOfWeek:Word): TDateTime
749: Func EncodeDateTimes(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSec:Word): TDateTime
750: Func EncodeDateWeek(const AYear,AWeekOfYear:Word;const ADayOfWeek:Word): TDateTime
751: Func EncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek Word): TDateTime
752: Func EncodeString( s :Str) :Str
753: Func DecodeString( s :Str) :Str
754: Func EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
755: Func EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
756: Func EndIP :Str
757: Func EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
758: Func EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
759: Func EndOfAMonth( const AYear, AMonth : Word) : TDateTime
760: Func EndOfAWeek(const AYear,AWeekOfYear:Word; const ADayOfWeek:Word):TDateTime
761: Func EndOfAYear( const AYear : Word) : TDateTime
762: Func EndOfTheDay( const AValue : TDateTime) : TDateTime
763: Func EndOfTheMonth( const AValue : TDateTime) : TDateTime
764: Func EndOfTheWeek( const AValue : TDateTime) : TDateTime
765: Func EndOfTheYear( const AValue : TDateTime) : TDateTime
766: Func EndPeriod( const Period :Card) :Bool
767: Func EndsStr( const ASubText, AText :Str) :Bool

```



```

768: Func EndsText ( const ASubText, AText :Str) :Bool
769: Func EnsureMsgIDBrackets ( const AMsgID :Str) :Str
770: Func EnsureRange ( const AValue, AMin, AMax : Int) : Int;
771: Func EnsureRange1 ( const AValue, AMin, AMax : Int64) : Int64;
772: Func EnsureRange2 ( const AValue, AMin, AMax : Double) : Double;
773: Func EOF:Bool
774: Func EOLn:Bool
775: Func EqualRect ( const R1, R2 : TRect) :Bool
776: Func EqualRect (const R1, R2: TRect):Bool)
777: Func Equals ( Strings : TWideStrings) :Bool
778: Func Equals (Strings: TStrings):Bool;
779: Func EqualState (oState: TniRegularExpressionState) :Bool
780: Func ErrOutput: Text)
781: Func ExceptionParam:Str;
782: Func ExceptionPos:Card;
783: Func ExceptionProc:Card;
784: Func ExceptionToString(er: TIFException; Param:Str):Str;
785: Func ExceptionType: TIFException;
786: Func ExcludeTrailingBackslash ( S :Str) :Str
787: Func ExcludeTrailingBackslash(const S:Str):Str)
788: Func ExcludeTrailingPathDelimiter ( const APath :Str) :Str
789: Func ExcludeTrailingPathDelimiter ( S :Str) :Str
790: Func ExcludeTrailingPathDelimiter(const S:Str):Str)
791: Func ExecConsoleApp(const AppName,Parameters:str;AppOutput:TStrings):bool;
792: Func ExecProc : Int
793: Func ExecSQL : Int
794: Func ExecSQL ( ExecDirect :Bool) : Int
795: Func Execute: _Recordset;
796: Func Execute:Bool
797: Func Execute:Bool;
798: Func Execute(const SQL:str;Params:TParams;Cache:Boolean;cursor:phDBICur):Int
799: Func Execute(const SQL: WideString;Params:TParams;ResultSet: TPSResult):Int
800: Func Execute ( ParentWnd : HWND) :Bool
801: Func Executel(constCommText:WideString;const CType:TCommType;const ExecuteOpts:TExecuteOpts):_Recordset;
802: Func Executel ( const Parameters : OleVariant) : _Recordset;
803: Func Executel ( ParentWnd : HWND) :Bool;
804: Func Execute2 ( var RecordsAffected:Int;const Parameters:OleVariant):_Recordset;
805: Func ExecuteAction ( Action : TBasicAction) :Bool
806: Func ExecuteDirect ( const SQL : WideString) : Int
807: Func ExecuteFile (const FileName:str;const Params:str;const DefDir:str;ShowCmd:Int) :THandle
808: Proc ExecuteThread2 (afunc:TThreadFunction2;thrOK:bool);AddTypeS ('TThreadFunction2', 'procedure
809: Func CreateThread2 (ThreadFunc: TThreadFunction2) : THandle
810: Func ExeFileIsRunning (ExeFile:Str):Bool;
811: Func ExePath:Str;
812: Func ScriptPath:Str;
813: Func ScriptName:Str;
814: Func ExePathName:Str;
815: Func Exists ( AItem : Pointer) :Bool
816: Func ExitWindows ( ExitCode :Card) :Bool
817: Func Exit2 ( ExitCode :Card) :Bool
818: Func Exp (x: Extended): Extended;
819: Func ExpandEnvironmentVar ( var Value :Str) :Bool
820: Func ExpandFileName ( FileName :Str) :Str
821: Func ExpandFileName (const FileName:Str):Str)
822: Func ExpandUNCFileName ( FileName :Str) :Str
823: Func ExpandUNCFileName (const FileName:Str):Str)
824: Func ExpJ ( const X : Float) : Float;
825: Func Exsecans ( X : Float) : Float
826: Func Extract ( const AByteCount : Int) :Str
827: Func Extract ( Item : TClass) : TClass
828: Func Extract ( Item : TComponent) : TComponent
829: Func Extract ( Item : TObject) : TObject
830: Func ExtractFileDir ( FileName :Str) :Str
831: Func ExtractFileDir (const FileName:Str):Str)
832: Func ExtractFileDrive ( FileName :Str) :Str
833: Func ExtractFileDrive (const FileName:Str):Str)
834: Func ExtractFileExt ( FileName :Str) :Str
835: Func ExtractFileExt (const FileName:Str):Str)
836: Func ExtractFileExtNoDot ( const FileName :Str) :Str
837: Func ExtractFileExtNoDotUpper ( const FileName :Str) :Str
838: Func ExtractFileName ( FileName :Str) :Str
839: Func ExtractFileName (const filename:Str):str;
840: Func ExtractFilePath ( FileName :Str) :Str
841: Func ExtractFilePath (const filename:Str):str;
842: Func ExtractRelativePath ( BaseName, DestName :Str) :Str
843: Func ExtractRelativePath (const BaseName:Str; const DestName:Str):Str)
844: Func ExtractShortPathName ( FileName :Str) :Str
845: Func ExtractShortPathName (const FileName:Str):Str)
846: Func ExtractStrings (Separators,WhiteSpace:TSysCharSet;Content:PChar;Strings:TStrings):Int
847: Func ExtractStrings (Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings):Int)
848: Func Fact (numb: Int): Extended;
849: Func FactInt (numb: Int): int64;
850: Func Factorial ( const N : Int) : Extended
851: Func FahrenheitToCelsius ( const AValue : Double) : Double
852: Func FalseBoolStrs: array of string
853: Func Fetch (var AInput:str;const ADelim:str;const ADelete:Bool;const ACaseSensitive:Bool):str
854: Func FetchCaseInsensitive (var AInput:str;const ADelim:str;const ADelete:Bool):str
855: Func Fibo (numb: Int): Extended;
856: Func FiboInt (numb: Int): Int64;

```

```

857: Func Fibonacci( const N : Int ) : Int
858: Func FIELDBYNAME(const FIELDNAME :Str) : TFIELD
859: Func FIELDBYNAME(const FIELDNAME : WIDESTRING) : TFIELD
860: Func FIELDBYNAME(const NAME :Str) : TFIELD
861: Func FIELDBYNAME(const NAME :Str) : TFIELDDEF
862: Func FIELDBYNUMBER( FIELDNO : Int) : TFIELD
863: Func FileAge( FileName :Str) : Int
864: Func FileAge(const FileName:Str): Int
865: Func FileCompareText( const A, B :Str) : Int
866: Func FileContains(const FileName:str;Text:str;CaseSensitive:Bool;ExceptionOnError:Bool):Bool
867: Func FileCreate(FileName:Str): Int;
868: //FileCreate2(FileName:str;Rights:Int):Int;
869: Func FileCreate(const FileName:Str): Int)
870: Func FileCreateTemp( var Prefix :Str) : THandle
871: Func FileDateToDateTime( FileDate : Int) : TDateTime
872: Func FileDateToDateTime(FileDate: Int): TDateTime;
873: Func FileExists( const FileName :Str) :Bool
874: Func FileExists( FileName :Str) :Bool
875: Func fileExists(const FileName:Str):Bool;
876: Func FileGetAttr( FileName :Str) : Int
877: Func FileGetAttr(const FileName:Str): Int)
878: Func FileGetDate( Handle : Int) : Int
879: Func FileGetDate(handle: Int): Int
880: Func FileGetDisplayName( const FileName :Str) :Str
881: Func FileGetSize( const FileName :Str) : Int
882: Func FileGetTempName( const Prefix :Str) :Str
883: Func FileGetTypeNames( const FileName :Str) :Str
884: Func FileIsReadOnly( FileName :Str) :Bool
885: Func FileLoad( ResType:TResType;const Name:str;MaskColor:TColor):Boolean
886: Func FileOpen( FileName :Str; Mode : LongWord) : Int
887: Func FileOpen(const FileName:Str; mode:Int): Int)
888: Func FileRead(handle: Int; Buffer: PChar; count: LongWord): Int
889: Func FileSearch( Name, DirList :Str) :Str
890: Func FileSearch(const Name, dirList:Str):Str)
891: Func FileSeek( Handle : Int; Offset : Int64; Origin : Int) : Int64;
892: Func FileSeek( Handle, Offset, Origin : Int) : Int;
893: Func FileSeek(handle, offset, origin: Int): Int
894: Func FileSetAttr( FileName :Str; Attr : Int) : Int
895: Func FileSetAttr(const FileName:Str; Attr: Int): Int)
896: Func FileSetDate(FileName :Str; Age : Int) : Int;
897: Func FileSetDate(handle: Int; age: Int): Int
898: Func FileSetDate2(FileHandle : Int; Age : Int) : Int;
899: Func FileSetDateH( Handle : Int; Age : Int) : Int;
900: Func FileSetReadOnly( FileName :Str; ReadOnly :Bool) :Bool
901: Func FileSize( const FileName :Str) : int64
902: Func FileSizeByName( const AFilename :Str) : Longint
903: Func FileWrite(Handle: Int; const Buffer: pChar; Count: LongWord): Int)
904: Func FilterSpecArray : TComdlgFilterSpecArray
905: Func FIND(ACAPTION :Str) : TMENUITEM
906: Func Find(AItem : Pointer; out AData : Pointer) :Bool
907: Func FIND(const ANAME :Str) : TNAMEDITEM
908: Func Find(const DisplayName :Str) : TAggregate
909: Func Find(const Item : TBookmarkStr; var Index : Int) :Bool
910: Func FIND(const NAME :Str) : TFIELD
911: Func FIND(const NAME :Str) : TFIELDDEF
912: Func FIND(const NAME :Str) : TINDEXDEF
913: Func Find(const S : WideString; var Index : Int) :Bool
914: Func Find(S:str;var Index:Int):Boolean
915: Func FindAuthClass( AuthName :Str) : TIdAuthenticationClass
916: Func FindBand( AControl : TControl) : TCoolBand
917: Func FindBoundary( AContentType :Str) :Str
918: Func FindButton( AModalResult : TModalResult) : TTaskDialogBaseButtonItem
919: Func FindCaption(StartIndex: Int; Value: str; Partial, Inclusive, Wrap :Bool):TListItem
920: Func FindCdLineSwitch( Switch :Str; IgnoreCase :Bool) :Bool;
921: Func FindCloseW(FindFile: Int): LongBool; stdcall;
922: Func FindCmdLineSwitch(Switch:str;Chars:TSysCharSet;IgnoreCase:Boolean):Bool;
923: Func FindCmndLineSwitch( Switch :Str):Bool;
924: Func FindComponent(ANAME:Str): TComponent
925: Func FindComponent(vlabel:Str): TComponent;
926: Func FindComponent2(vlabel:Str): TComponent;
927: Func FindControl(Handle: HWnd): TWinControl;
928: Func FindData( StartIndex : Int; Value : Pointer; Inclusive, Wrap :Bool):TListItem
929: Func FindDatabase( const DatabaseName :Str) : TDatabase
930: Func FindDragTarget(const Pos: TPoint; AllowDisabled:Bool): TControl;
931: Func FINDFIELD( const FIELDNAME :Str) : TFIELD
932: Func FINDFIELD( const FIELDNAME : WideString) : TFIELD
933: Func FindFirst2(const Path:Str; Attr: Int; var F: TSearchRec):Int)
934: Func FindNext2(var F: TSearchRec): Int)
935: Proc FindClose2(var F: TSearchRec)
936: Func FINDFIRST :Bool
937: TJvmSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
938: sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms,sfRecent,sfSendTo,
sfStartMenu, stStartup, sfTemplates);
939: FFolder: array [TJvmSpecialFolder] of Int =
940: (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
941: CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
942: CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
943: CSIDL_STARTUP, CSIDL_TEMPLATES);
944: Func FindFilesDlg(StartIn:str;SpecialFolder:TJvmSpecialFolder;UseFolder:Bool):Bool);

```

```

945: Func Findfirst(const filepath:Str; attr: Int): Int;
946: Func FindFirst2(const Path:Str; Attr: Int; var F: TSearchRec): Int)
947: Func FindFirstNotOf( AFind, AText :Str) : Int
948: Func FindFirstOf( AFind, AText :Str) : Int
949: Func FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
950: Func FINDINDEXFORFIELDS( const FIELDS :Str) : TINDEXDEF
951: Func FindInstanceOf( AClass : TClass; AExact :Bool; AStartAt : Int) : Int
952: Func FINDITEM( VALUE : Int; KIND : TFINDITEMKIND) : TMENUITEM
953: Func FindItemId( Id : Int) : TCollectionItem
954: Func FindKey( const KeyValues : array of const) :Bool
955: Func FINDLAST:BOOLEAN
956: Func FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
957: Func FindModuleClass( AClass : TComponentClass) : TComponent
958: Func FindModuleName( const AClass :Str) : TComponent
959: Func FINDNEXT :Bool
960: Func FindNext: Int;
961: Func FindNext2(var F: TSearchRec): Int)
962: Func FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible :Bool): TTabSheet
963: Func FindNextToSelect : TTreeNode
964: Func FINDPARAM( const VALUE :Str) : TPARAM
965: Func FindParam( const Value : WideString) : TParameter
966: Func FINDPRIOR :Bool
967: Func FindResource( ModuleHandle:HMODULE; ResourceName, ResourceType:PChar): TResourceHandle
968: Func FindSession( const SessionName :Str) : TSession
969: Func FindStringResource( Ident: Int):Str)
970: Func FindText(const SearchStr:str; StartPos, Length: Int; Options: TSearchTypes): Int
971: Func FindUnusedFileName( const FileName, FileExt, Suffix : Ansistr) : Ansistr
972: Func FindVCLWindow(const Pos: TPoint) : TWinControl;
973: Func FindWindow( C1, C2: PChar): Longint;
974: Func FindInPaths(const fileName, paths:Str): Str;
975: Func Finger :Str
976: Func First: TClass
977: Func First: TComponent
978: Func First: TObject
979: Func FirstDelimiter( const delimiters :Str; const Str :Str) : Int;
980: Func FirstDelimiter1(const delimiters:WideString; const Str : WideString) : Int;
981: Func FirstInstance( const ATitle :Str) :Bool
982: Func FloatPoint( const X, Y : Float) : TFloatPoint;
983: Func FloatPoint1( const P : TPoint) : TFloatPoint;
984: Func FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) :Bool
985: Func FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
986: Func FloatRect1( const Rect : TRect) : TFloatRect;
987: Func FloatsEqual( const X, Y : Float) :Bool
988: Func FloatToBin(const D: Double):Str; //doubletohex -> hex to bin! in buffer
989: Func FloatToCurr( Value : Extended) : Currency
990: Func FloatToDateTime( Value : Extended) : TDateTime
991: Func FloatToStr( Value : Extended) :Str;
992: Func FloatToStr(e : Extended) :Str;
993: Func FloatToStrF( Value:Extended; Format:TFloatFormat; Precision,Digits: Int):str;
994: Func FloatToStrF( Value:Extended; Format:TFloatFormat; Precision: Int; Digits: Int):Str)
995: Func FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Int; FormatSettings :
TFormatSettings) :Str;
996: Func FloatToStrFS( Value:Extended; Format:TFloatFormat; Precision,
Digits: Int; FormatSettings:TFormatSettings) :str;
997: Func FloatToText( BufferArg:PChar; const Value:Extended; ValueType:TFloatValue; Format:TFloatFormat; Precision,
Digits: Int):Int)
998: Func Floor( const X : Extended) : Int
999: Func FloorInt( Value : Int; StepSize : Int) : Int
1000: Func FloorJ( const X : Extended) : Int
1001: Func Flush( const Count :Card) :Bool
1002: Func Flush(var t: Text): Int
1003: Func FmtLoadStr(Ident: Int; const Args: array of const):Str)
1004: Func FOCUSED:BOOLEAN
1005: Func ForceBackslash( const PathName :Str) :Str
1006: Func ForceDirectories( const Dir :Str) :Bool
1007: Func ForceDirectories( Dir :Str) :Bool
1008: Func ForceDirectories( Name :Str) :Bool
1009: Func ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
1010: Func ForceInRange( A, Min, Max : Int) : Int
1011: Func ForceInRangeR( const A, Min, Max : Double) : Double
1012: Func ForEach( AProc : TBucketProc; AInfo : Pointer) :Bool;
1013: Func ForEach1( AEvent : TBucketEvent) :Bool;
1014: Func ForegroundTask:Bool
1015: Func Format(const Format:Str; const Args: array of const):Str;
1016: Func FormatBcd( const Format :Str; Bcd : TBcd) :Str
1017: Func FormatBigInt(s:Str):Str;
1018: Func FormatByteSize(const bytes: int64):Str;
1019: Func FormatBuf(var Buffer:PChar; BufLen:Card; const Format:str; FmtLen:Card; const Args:array of const):Card
1020: Func FormatCount(iCount: Int; const sSingular:str; const sPlural:str) :str
1021: Func FormatCurr( Format :Str; Value : Currency) :Str;
1022: Func FormatCurr(const Format:Str; Value: Currency):Str)
1023: Func FormatDateTime( Format :Str; DateTime : TDateTime) :Str;
1024: Func FormatDateTime(const fmt:Str; D: TDateTime):Str;
1025: Func FormatDateTime2(const Formatting:str; DateTime:TDateTime; const FormatSettings: TFormatSettings):str;
1026: Func FormatFloat( Format :Str; Value : Extended) :Str;
1027: Func FormatFloat(const Format:Str; Value: Extended):Str)
1028: Func FormatFloat( Format :Str; Value : Extended) :Str;
1029: Func FormatFloat2(Format:str; Value:Extended; FormatSettings:TFormatSettings):str;
1030: Func FormatF('%2f - %.12f ', [notes[it], (ln(notes[it+1])/notes[it])*1200]/ln(2)]); //with println - writeln

```

```

1031: Func FormatCurr( Format :Str; Value : Currency) :Str;
1032: Func FormatCurr2(Format:str;Value:Currency;FormatSettings:TFormatSettings):str;
1033: Func Format2(const Frmat:str;const Args:array of const;const FSettings:TFormatSettings):str
1034: Func FormatInt(i: Int):Str;
1035: Func FormatInt64(i: int64):Str;
1036: Func FormatMaskText( const EditMask :Str; const Value :Str) :Str
1037: Func FormatValue( AValue :Card) :Str
1038: Func FormatVersionString( const HiV, LoV : Word) :Str;
1039: Func FormatVersionString1( const Major, Minor, Build, Revision : Word) :Str;
1040: Func Frac(X: Extended): Extended;
1041: Func FreeResource( ResData : HGLOBAL) : LongBool
1042: Func FromCommon( const AValue : Double) : Double
1043: Func FromCommon(const AValue: Double): Double;
1044: Func FTPGMTDateTimeToMLS( const ATimestamp:TDateTime; const AIncludeMSecs:Boolean):str
1045: Func FTPLocalDateTimeToMLS( const ATimestamp:TDateTime;const AIncludeMSecs:Boolean):str
1046: Func FTPMLSToGMTDateTime( const ATimestamp :Str) : TDateTime
1047: Func FTPMLSToLocalDateTime( const ATimestamp :Str) : TDateTime
1048: Func FuncIn(AValue: Variant; ASet: Variant):Bool;
1049: //Func FuncList Size is: 6444 of mX3.9.8.9
1050: Func FutureValue(const Rate:Extended;NPeriods:Int;const Payment,
PresentValue:Extended;PaymentTime:TPaymentTime):Extended
1051: Func FullTimeToStr(SUMTime: TDateTime):Str;;
1052: Func Gauss( const x, Spread : Double) : Double
1053: Func Gauss(const x,Spread: Double): Double;
1054: Func GCD(x, y : LongInt) : LongInt;
1055: Func GCDJ( X, Y :Card) :Card
1056: Func GDAL: LongWord
1057: Func GdiFlush : BOOL
1058: Func GdiSetBatchLimit( Limit : DWORD) : DWORD
1059: Func GdiGetBatchLimit : DWORD
1060: Func GenerateHeader : TIdHeaderList
1061: Func GeometricMean( const X : TDynFloatArray) : Float
1062: Func Get( AURL :Str) :Str;
1063: Func Get2( AURL :Str) :Str;
1064: Func Get8087CW : Word
1065: Func GetActiveOleObject(const ClassName:Str): IDispatch;
1066: Func GetAliasDriverName( const AliasName :Str) :Str
1067: Func GetAPMBatteryFlag : TAPMBatteryFlag
1068: Func GetAPMBatteryFullLifeTime : DWORD
1069: Func GetAPMBatteryLifePercent : Int
1070: Func GetAPMBatteryLifeTime : DWORD
1071: Func GetAPMLineStatus : TAPMLineStatus
1072: Func GetAppdataFolder :Str
1073: Func GetAppDispatcher : TComponent
1074: Func GetArrayLength: Int;
1075: Func GetASCII:Str;
1076: Func GetASCIILine:Str;
1077: Func GetAsHandle( Format : Word) : THandle
1078: Func GetAssociatedProgram(const Extension:str;var Filename,Description:str):bool;
1079: Func GetBackupFileName( const FileName :Str) :Str
1080: Func GetBaseAddress(PID:DWORD):DWORD; //Process API
1081: Func GetBBitmap( Value : TBitmap) : TBitmap
1082: Func GetBIOSCopyright :Str
1083: Func GetBIOSDate : TDateTime
1084: Func GetBIOSExtendedInfo :Str
1085: Func GetBIOSName :Str
1086: Func getBitmap(apat:Str): TBitmap;
1087: Func GetBitmap( Index : Int; Image : TBitmap) :Bool //object
1088: Func GetBitMapObject(const bitmappath:Str): TBitmap;
1089: Func GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1090: Func GetCapsLockKeyState :Bool
1091: Func GetCaptureControl: TControl;
1092: Func GetCDAudioTrackList(var TrackList:TJclCdTrackInfoArray;Drive:Char):TJclCdTrackInfo;
1093: Func GetCDAudioTrackList1(TrackList:TStrings;IncludeTrackType:Boolean;Drive:Char):str;
1094: Func GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) :Str
1095: Func GetChangedText( const Text:str; SelStart, SelLength : Int; Key : Char):str
1096: Func GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1097: Func GetClockValue : Int64
1098: Func getCmdLine: PChar;
1099: Func getCmdShow: Int;
1100: Func GetCPUSpeed: Double;
1101: Func GetColField( DataCol : Int) : TField
1102: Func GetColorBlue( const Color : TColor) : Byte
1103: Func GetColorFlag( const Color : TColor) : Byte
1104: Func GetColorGreen( const Color : TColor) : Byte
1105: Func GetColorRed( const Color : TColor) : Byte
1106: Func GetComCtlVersion : Int
1107: Func GetComPorts: TStringlist;
1108: Func GetCommonAppdataFolder :Str
1109: Func GetCommonDesktopdirectoryFolder:Str
1110: Func GetCommonFavoritesFolder :Str
1111: Func GetCommonFilesFolder :Str
1112: Func GetCommonProgramsFolder :Str
1113: Func GetCommonStartmenuFolder :Str
1114: Func GetCommonStartupFolder :Str
1115: Func GetComponent( Owner, Parent : TComponent) : TComponent
1116: Func GetConnectionRegistryFile( DesignMode :Bool):Str
1117: Func GetCookiesFolder :Str
1118: Func GetCPUSpeed( var CpuSpeed : TFreqInfo) :Bool

```

```

1119: Func GetCurrent:TFavoriteLinkItem
1120: Func GetCurrent:TListItem
1121: Func GetCurrent:TTaskDialogBaseButtonItem
1122: Func GetCurrent:TToolButton
1123: Func GetCurrent:TTreeNode
1124: Func GetCurrent:WideString
1125: Func GetCurrentDir :Str
1126: Func GetCurrentDir:Str)
1127: Func GetCurrentFolder :Str
1128: Func GETCURRENTRECORD( BUFFER : PCHAR) :Bool
1129: Func GetCurrentProcessId : TidPID
1130: Func GetCurrentThreadHandle : THandle
1131: Func GetCurrentThreadID: LongWord; stdcall;
1132: Func GetCustomHeader( const Name :Str) :Str
1133: Func GetDataItem( Value : Pointer) : Longint
1134: Func GetDataLinkFiles(FileNames:TWideStrings; Directory string): Int;
1135: Func GetDataLinkFiles1( FileNames : TStrings; Directory:Str):Int;
1136: Func GETDATASIZE : Int
1137: Func GetDC(hwnd: HWND): HDC;
1138: Func GetDefaultFileExt( const MIMEType :Str) :Str
1139: Func GetDefaults :Bool
1140: Func GetDefaultSchemaName : WideString
1141: Func GetDefaultStreamLoader : IStreamLoader
1142: Func GetDesktopDirectoryFolder :Str
1143: Func GetDesktopFolder :Str
1144: Func GetDFASState( oStates : TList) : TniRegularExpressionState
1145: Func GetDirectorySize( const Path :Str) : Int64
1146: Func GetDisplayWidth : Int
1147: Func GetDLLVersion(const DLLName:str;var pdwMajor, pdwMinor:Int):Boolean
1148: Func GetDomainName :Str
1149: Func GetDriverRegistryFile( DesignMode :Bool) :Str
1150: Func GetDriveType(rootpath: pchar):Card;
1151: Func GetDriveTypeStr( const Drive : Char) :Str
1152: Func GetEnumerator : TFavoriteLinkItemsEnumerator
1153: Func GetEnumerator : TListItemsEnumerator
1154: Func GetEnumerator : TTaskDialogButtonsEnumerator
1155: Func GetEnumerator : TToolBarEnumerator
1156: Func GetEnumerator : TTreeNodesEnumerator
1157: Func GetEnumerator : TWideStringsEnumerator
1158: Func GetEnvVar( const VarName :Str) :Str
1159: Func GetEnvironmentVar( const AVariableName :Str) :Str
1160: Func GetEnvironmentVariable( const VarName :Str) :Str
1161: Func GetEnvironmentVar(const Name string; var Value:str;Expand:Boolean):Bool
1162: Func GetEnvironmentVars( const Vars : TStrings; Expand :Bool):Boolean
1163: Func getEnvironmentString:Str;
1164: Func GetExceptionHandler : TObject
1165: Func GetFavoritesFolder :Str
1166: Func GetFieldByName( const Name :Str) :Str
1167: Func GetFieldInfo(const Origin: WideString;var FieldInfo TFieldInfo):Boolean
1168: Func GetFieldValue( ACol : Int) :Str
1169: Func GetFileAgeCoherence( const FileName :Str) :Bool
1170: Func GetFileCreation( const FileName :Str) : TFileTime
1171: Func GetFileCreationTime( const Filename :Str) : TDateTime
1172: Func GetFileInformation( const FileName :Str) : TSearchRec
1173: Func GetFileLastAccess( const FileName :Str) : TFileTime
1174: Func GetFileLastWrite( const FileName :Str) : TFileTime
1175: Func GetFileList(FileList: TStringlist; apath:Str): TStringlist;
1176: Func GetFileList1(apath:Str): TStringlist;
1177: Func GetFileMIMEType( const AFileName :Str) :Str
1178: Func GetFileSize( const FileName :Str) : Int64
1179: Func GetFileVersion( AFileName :Str) :Card
1180: Func GetFileVersion( const AFilename :Str) :Card
1181: Func GetFileSize2(Handle: Int; x: Int): Int; stdcall;
1182: Func GetFileDate(aFile:str; aWithTime:Boolean):str;
1183: Func GetFileCount(adirmask:Str): Int; //files count in directory!
1184: Func GetFilterData( Root : PExprNode) : TExprData
1185: Func getFirstChild : LongInt
1186: Func getFirstChild : TTreeNode
1187: Func GetFirstDelimitedToken( const cDelim:char;const cStr:Str):str
1188: Func GetFirstNode : TTreeNode
1189: Func GetFontsFolder :Str
1190: Func GetFormulaValue( const Formula :Str) : Extended
1191: Func GetFreePageFileMemory : Int
1192: Func GetFreePhysicalMemory : Int
1193: Func GetFreeSystemResources( const ResourceType : TFreeSysResKind): Int;
1194: Func GetFreeSystemResources1 : TFreeSystemResources;
1195: Func GetFreeVirtualMemory : Int
1196: Func GetFromClipboard :Bool
1197: Func GetFullURI( const AOptionalFields : TidURIOptionalFieldsSet):Str
1198: Func GetGBitmap( Value : TBitmap) : TBitmap
1199: Func GetGMTDateByName( const AFileName : TidFileName) : TDateTime
1200: Func GetGroupState( Level : Int) : TGroupPosInds
1201: Func GetHandle : HWND
1202: Func GETHELPCONTEXT( VALUE : Int; BYCOMMAND :Bool) : THELPCONTEXT
1203: Func GetHexArray(ahexdig: THexArray): THexArray;
1204: Func GetHighLightColor( const Color : TColor; Luminance : Int):TColor
1205: Func GetHINSTANCE: longword;
1206: Func GetHistoryFolder :Str
1207: Func GetHitTestInfoAt( X, Y : Int) : THitTests

```



```

1208: Func getHMODULE: longword;
1209: Func GetHostByName(const AComputerName:Str):Str;
1210: Func GetHostName:Str
1211: Func getHostIP:Str;
1212: Func GetHotSpot : TPoint
1213: Func GetHueBitmap( Value : TBitmap) : TBitmap
1214: Func GetImageBitmap : HBITMAP
1215: Func GETIMAGELIST : TCUSTOMIMAGELIST
1216: Func GetIncome( const aNetto : Currency): Currency
1217: Func GetIncome( const aNetto : Extended): Extended
1218: Func GetIncome( const aNetto : Extended): Extended
1219: Func GetIncome(const aNetto : Extended) : Extended
1220: Func GetIncome(const aNetto: Currency): Currency
1221: Func GetIncome2( const aNetto : Currency) : Currency
1222: Func GetIncome2( const aNetto : Currency): Currency
1223: Func getIndex_Attrs( tag :Str; var idx : Int; var Attrs :Str) :Str
1224: Func GETINDEXFORFIELDS( const FIELDS :Str; CASEINSENSITIVE :Bool) : TINDEXDEF
1225: Func GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet): TIndexDef
1226: Func GetInstRes(Instance:THandle;ResType:TResType;const
Name:Str;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1227: Func
GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1228: Func GetIntelCacheDescription( const D : Byte):Str
1229: Func GetInteractiveUserName :Str
1230: Func GetInternetCacheFolder :Str
1231: Func GetInternetFormattedFileTimeStamp( const AFilename :Str) :Str
1232: Func GetIPAddress( const HostName :Str) :Str
1233: Func GetIP( const HostName :Str) :Str
1234: Func GetIPHostByName(const AComputerName:Str):Str;
1235: Func GetIsAdmin:Bool;
1236: Func GetItem( X, Y : Int) : LongInt
1237: Func GetItemAt( X, Y : Int) : TListItem
1238: Func GetItemHeight(Font: TFont): Int;
1239: Func GetItemPath( Index : Int) :Str
1240: Func GetKeyFieldNames( List : TStrings) : Int;
1241: Func GetKeyFieldNames1( List : TWideStrings) : Int;
1242: Func GetKeyState( const VirtualKey :Card) :Bool
1243: Func GetLastChild : LongInt
1244: Func GetLastChild : TTreeNode
1245: Func GetLastDelimitedToken(const cDelim:char;const cStr:str):Str
1246: Func GetLastError: Int
1247: Func GetLAT_CONV_FACTOR: double; //for WGS84 power(1 - 1 / 298.257223563, 2);
1248: Func GetLinesCount(sFileName :Str): Int;
1249: Func GetLoader( Ext :Str) : TBitmapLoader
1250: Func GetLoadFilter :Str
1251: Func GetLocalComputerName :Str
1252: Func GetLocaleChar( Locale, LocaleType : Int; Default : Char) : Char
1253: Func GetLocaleStr( Locale, LocaleType : Int; Default :Str) :Str
1254: Func GetLocalUserName :Str
1255: Func GetLoginUsername : WideString
1256: Func getLongDayNames:Str)
1257: Func GetLongHint(const hint:Str):Str
1258: Func getLongMonthNames:Str)
1259: Func GetMacAddresses( const Machine :Str; const Addresses : TStrings) : Int
1260: Func GetMainAppWndFromPid( PID : DWORD) : HWND
1261: Func GetMapX(C_form,apath:Str; const Data:Str):Bool; //c_form: [html/json/xml]
1262: //if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne then
1263: Proc GetGEOMap(C_form,apath:Str; const Data:Str);
1264: Func GetMapXGeoReverse(C_form:Str; const lat,long:Str):Str;
1265: //if GetMapXGeoReverse('XML',topPath,'47.0397826','7.62914761277888 then
1266: Func GetGeoCode(C_form,apath:Str; const data:Str; sfile:bool):Str;
1267: Func GetMaskBitmap : HBITMAP
1268: Func GetMaxAppAddress : Int
1269: Func GetMciErrorMessage( const MciErrNo : MCIERROR) :Str
1270: Func GetMemoryLoad : Byte
1271: Func GetMIMEDefaultFileExt( const MIMEType :Str) : TidFileName
1272: Func GetMIMETypeFromFile( const AFile :Str) :Str
1273: Func GetMIMETypeFromFile( const AFile : TidFileName) :Str
1274: Func GetMinAppAddress : Int
1275: Func GetModule : TComponent
1276: Func GetModuleHandle( ModuleName : PChar) : HMODULE
1277: Func GetModuleName( Module : HMODULE) :Str
1278: Func GetModulePath( const Module : HMODULE) :Str
1279: Func GetModuleFileName(Module: Int; Filename: PChar;Size: Int): Int; stdcall;
1280: Func GetMorseID(InChar : Char): Word;;
1281: Func GetMorseString2(InChar : Char):Str;;
1282: Func GetMorseLine(dots:Bool):Str;; //whole table! {1 or dots}
1283: Func GetMorseTable(dots:Bool):Str;; //whole table!
1284: Func GetMorseSign(InChar : Char):Str;;
1285: Func GetCommandLine: PChar; stdcall;
1286: Func GetMonochromeBitmap( Value : TBitmap) : TBitmap
1287: Func GetMultiN(aval: Int):Str;
1288: Func GetName :Str
1289: Func GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1290: Func GetNethoodFolder :Str
1291: Func GetNext : TTreeNode
1292: Func GetNextChild( Value : LongInt) : LongInt
1293: Func GetNextChild( Value : TTreeNode) : TTreeNode
1294: Func GetNextDelimitedToken( const cDelim : char; var cStr :Str) :Str

```

```

1295: Func GetNextItem(StartItem:TListItem;Direction:TSearchDirection;States:TItemStates):TListItem
1296: Func GetNextPacket : Int
1297: Func getNextSibling : TTreeNode
1298: Func GetNextVisible : TTreeNode
1299: Func GetNode( ItemId : HTreeItem) : TTreeNode
1300: Func GetNodeAt( X, Y : Int) : TTreeNode
1301: Func GetNodeDisplayWidth( Node : TOutlineNode) : Int
1302: Func GetNumberOfProcessors: longint;
1303: Func GetNumLockKeyState :Bool
1304: Func GetObjectProperty( Instance : TPersistent; const PropName :Str) : TObject
1305: Func GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1306: Func GetOptionalParam( const ParamName :Str) : OleVariant
1307: Func GetOSName:Str;
1308: Func GetOSVersion:Str;
1309: Func GetOSNumber:Str;
1310: Func GetOsVersionInfo: TOSVersionInfo; //thx to wischniewski
1311: Func GetPackageModuleHandle( PackageName : PChar) : HMODULE
1312: Func GetPageSize:Card;
1313: Func GetParameterFileName :Str
1314: Func GetParams( var OwnerData : OleVariant) : OleVariant
1315: Func GETPARENTCOMPONENT : TCOMPONENT
1316: Func GetParentForm(control: TControl): TForm
1317: Func GETPARENTMENU : TMENU
1318: Func GetPassword:Bool
1319: Func GetPassword:Str
1320: Func GetPersonalFolder :Str
1321: Func GetPidFromProcessName( const ProcessName :Str) : DWORD
1322: Func getPI: extended; //of const PI math
1323: Func GetPixel(dc:HDC; PixelCoords.X: TPoint, PixelCoords.Y: TPoint);
1324: Func GetPosition : TPoint
1325: Func GetPrev : TTreeNode
1326: Func GetPrevChild( Value : LongInt) : LongInt
1327: Func GetPrevChild( Value : TTreeNode) : TTreeNode
1328: Func getPrevSibling : TTreeNode
1329: Func GetPrevVisible : TTreeNode
1330: Func GetPrinthoodFolder :Str
1331: Func GetPrivilegeDisplayName( const PrivilegeName:Str):Str
1332: Func getProcessList: TStrings;
1333: Func GetProcessId : TidPID
1334: Func GetProcessNameFromPid( PID : DWORD) :Str
1335: Func GetProcessNameFromWnd( Wnd : HWND) :Str
1336: Func GetProcessMemoryInfo(Process:THandle;ppsmemCounters:TProcessMemoryCounters;cb:DWORD):BOOL
1337: Func getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1338: Func getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1339: Func GetProgramFilesFolder :Str
1340: Func GetProgramsFolder :Str
1341: Func GetProxy :Str
1342: Func GetQuoteChar : WideString
1343: Func GetQrCode4(Width,Height:Word; Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
1344: Func GetQrCodetoFile(Width,Height:Word;Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
1345: Func GetRate : Double
1346: Func getPerfTime:Str;
1347: Func getRuntime:Str;
1348: Func GetRBitmap( Value : TBitmap) : TBitmap
1349: Func GetReadableName( const AName :Str) :Str
1350: Func GetRecentDocs : TStringList
1351: Func GetRecentFolder :Str
1352: Func GetRecords( Count : Int; out RecsOut : Int; Options : Int) : OleVariant;
1353: Func GetRecords1(Count:Int;out RecsOut:Int;Options:Int;const CommandText:WideString;var Params,
OwnerData:OleVariant):OleVariant;
1354: Func GetRecordset(const CommandText: WideString;ConnectionString:WideString):_Recordset
1355: Func GetRegisteredCompany :Str
1356: Func GetRegisteredOwner :Str
1357: Func GetResource(ResType:TResType;const Name:str;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor:Bool
1358: Func GetResourceName( ObjStream : TStream; var AName :Str) :Bool
1359: Func GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1360: Func GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1361: Func GetRValue(rgb: DWORD) : Byte
1362: Func GetGValue(rgb: DWORD) : Byte
1363: Func GetBValue(rgb: DWORD) : Byte
1364: Func GetCValue(cmyk: COLORREF) : Byte
1365: Func GetMValue(cmyk: COLORREF) : Byte
1366: Func GetYValue(cmyk: COLORREF) : Byte
1367: Func GetKValue(cmyk: COLORREF) : Byte
1368: Func CMYK( c, m, y, k : Byte) : COLORREF
1369: Proc GetScreenShot(var ABitmap : TBitmap);
1370: Func GetOSName:Str;
1371: Func GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR):FARPROC
1372: Func GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1373: Func GetSafeCallExceptionMsg :Str
1374: Func GetSaturationBitmap( Value : TBitmap) : TBitmap
1375: Func GetSaveFilter :Str
1376: Func GetSaver( Ext :Str) : TBitmapLoader
1377: Func GetScrollLockKeyState :Bool
1378: Func GetSearchString :Str
1379: Func GetSelections( AList : TList) : TTreeNode
1380: Func GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:Int):Int
1381: Func GetSendToFolder :Str
1382: Func GetServer : TAppServer

```

```

1383: Func GetServerList : OleVariant
1384: Func GetShadowColor( const Color : TColor; Luminance:Int): TColor
1385: Func GetShellProcessHandle : THandle
1386: Func GetShellProcessName :Str
1387: Func GetShellVersion :Card
1388: Func getShortDayNames:Str
1389: Func GetShortHint(const hint:Str):Str
1390: Func getShortMonthNames:Str
1391: Func GetSizeOfFile( const FileName :Str) : Int64;
1392: Func GetSizeOfFile( Handle : THandle) : Int64;
1393: Func GetStdHandle(nStdHandle: Int): Int; stdcall;
1394: Func GetStartmenuFolder :Str
1395: Func GetStartupFolder :Str
1396: Func GetStringProperty( Instance: TPersistent; const PropName:str):WideString
1397: Func GetSuccessor( cChar : char): TniRegularExpressionState
1398: Func GetSwapFileSize : Int
1399: Func GetSwapFileUsage : Int
1400: Func GetSystemLocale : TIdCharSet
1401: Func GetSystemMetrics( nIndex : Int) : Int
1402: Func GetSystemPathSH(Folder: Int): TFilename ;
1403: Func GetTableNameFromQuery( const SQL : WideString) : WideString
1404: Func GetTableNameFromSQL( const SQL : WideString) : WideString
1405: Func GetTableNameFromSQLEx(const SQL:WideString; IdOption:IDENTIFIEROption):WideString
1406: Func GetTasksList( const List : TStrings) :Bool
1407: Func getTeamViewerID:Str;
1408: Func GetTemplatesFolder :Str
1409: Func GetText : PwideChar
1410: Func GetText:PChar
1411: Func GetTextBuf( Buffer : PChar; BufSize : Int) : Int
1412: Func GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:Int):Int
1413: Func GetTextItem( const Value :Str) : Longint
1414: Func GETTEXTLEN:Int
1415: Func GetThreadLocale: Longint; stdcall
1416: Func GetCurrentThreadID: LongWord; stdcall;
1417: Func GetTickCount :Card
1418: Func GetTickDiff( const AOldTickCount, ANewTickCount :Card) :Card
1419: Func GetTicketNr : longint
1420: Func GetTime :Card
1421: Func GetTime : TDateTime
1422: Func GetTimeout : Int
1423: Func GetTimeStr:Str
1424: Func GetTimeString:Str
1425: Func GetTimeZoneInformation(var lpTimeZoneInformation: TTimeZoneInformation):DWORD
1426: Ex: if GetTimeZoneInformation(tzi) >0 then writeln(itoa(tzi.bias));
1427: Func GetTodayFiles(startdir, amask:Str): TStringlist;
1428: Func getTokenCounts : Int
1429: Func GetTotalPageFileMemory : Int
1430: Func GetTotalPhysicalMemory : Int
1431: Func GetTotalVirtualMemory : Int
1432: Func GetUniqueFileName( const APath, APrefix, AExt :Str) :Str
1433: Func GetUseNowForDate :Bool
1434: Func GetUserDomainName( const CurUser :Str) :Str
1435: Func GetUserName :Str
1436: Func GetUserName:Str;
1437: Func GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD):BOOL;
1438: Func GetUserObjectName( hUserObject : THandle) :Str
1439: Func GetValueBitmap( Value : TBitmap) : TBitmap
1440: Func GetValueMSec :Card
1441: Func GetValueStr :Str
1442: Func GetVersion: int;
1443: Func GetVersionString(FileName:Str):Str;
1444: Func getVideoDrivers:Str;
1445: Func GetVisibleNode( Index : LongInt) : TOutlineNode
1446: Func GetVolumeFileSystem( const Drive :Str) :Str
1447: Func GetVolumeName( const Drive :Str) :Str
1448: Func GetVolumeSerialNumber( const Drive :Str) :Str
1449: Func GetWebAppServices : IWebAppServices
1450: Func GetWebRequestHandler : IWebRequestHandler
1451: Func GetWindowCaption( Wnd : HWND) :Str
1452: Func GetWindowDC(hdwnd: HWND): HDC;
1453: Func GetWindowIcon( Wnd : HWND; LargeIcon :Bool) : HICON
1454: Func GetWindowRect(hwnd: HWND; arect: TRect):Bool
1455: Func GetWindowsComputerID :Str
1456: Func GetWindowsDirectory(lpBuffer: PChar; uSize: longword):longword;
1457: Func GetWindowsFolder :Str
1458: Func GetWindowsServicePackVersion : Int
1459: Func GetWindowsServicePackVersionString :Str
1460: Func GetWindowsSystemFolder :Str
1461: Func GetWindowsTempFolder :Str
1462: Func GetWindowsUserID :Str
1463: Func GetWindowsVersion : TWindowsVersion
1464: Func GetWindowsVersionString :Str
1465: Func GmtOffsetStrToDateTime( S :Str) : TDateTime
1466: Func GMTToLocalDateTime( S :Str) : TDateTime
1467: Func GotoKey :Bool
1468: Func GradToCycle( const Grads : Extended) : Extended
1469: Func GradToDeg( const Grads : Extended) : Extended
1470: Func GradToDeg( const Value : Extended) : Extended;
1471: Func GradToDegl( const Value : Double) : Double;

```

```

1472: Func GradToDeg2( const Value : Single) : Single;
1473: Func GradToRad( const Grads : Extended) : Extended
1474: Func GradToRad( const Value : Extended) : Extended;
1475: Func GradToRad1( const Value : Double) : Double;
1476: Func GradToRad2( const Value : Single) : Single;
1477: Func Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1478: Func GreenComponent( const Color32 : TColor32) : Int
1479: Func GUIDToString(const GUID: TGUID):Str
1480: Func HandleAllocated :Bool
1481: Func HandleAllocated:Bool;
1482: Func HandleRequest :Bool
1483: Func HandleRequest( Request : TWebRequest; Response : TWebResponse:Boolean
1484: Func HarmonicMean( const X : TDynFloatArray) : Float
1485: Func HasAsParent( Value : TTreeNode) :Bool
1486: Func HASCHILDDDFS :Bool
1487: Func HasCurValues :Bool
1488: Func HasExtendCharacter( const s : UTF8String) :Bool
1489: Func HasFormat( Format : Word) :Bool
1490: Func HashValue( AStream : TStream) : T5x4LongWordRecord;
1491: Func HashValue(AStream : TStream) : T4x4LongWordRecord
1492: Func HashValue(AStream: TStream): LongWord
1493: Func HashValue(AStream: TStream): Word
1494: Func HashValue1(AStream TStream; const ABeginPos, AEndPos: Int64): T5x4LongWordRecord;
1495: Func HashValue1(AStream : TStream) : T4x4LongWordRecord
1496: Func HashValue128(const ASrc: Str): T4x4LongWordRecord;
1497: Func HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1498: Func HashValue16( const ASrc : Str) : Word;
1499: Func HashValue16Stream( AStream : TStream) : Word;
1500: Func HashValue32( const ASrc : Str) : LongWord;
1501: Func HashValue32Stream( AStream : TStream) : LongWord;
1502: Func HasMergeConflicts :Bool
1503: Func hasMoreTokens :Bool
1504: Func HASPARENT :Bool
1505: Func HasParent:Bool
1506: Func HasTransaction( Transaction : TDBXTransaction) :Bool
1507: Func HasUTF8BOM( S : TStream) :Bool;
1508: Func HasUTF8BOM1( S : Ansistr) :Bool;
1509: Func Haversine( X : Float) : Float
1510: Func Head( s : Str; const subs : Str; var tail : Str): str
1511: Func HELPCOMMAND(COMMAND: Int; DATA: LONGINT): BOOLEAN
1512: Func HELPCONTEXT(CONTEXT: THELPCONTEXT): BOOLEAN
1513: Func HELPJUMP(JUMPID: str): BOOLEAN
1514: Func HeronianMean( const a, b : Float) : Float
1515: Func HexStrToStr(Value: Str): Str;
1516: Func HexToBin(Text, Buffer: PChar; BufSize: Int): Int;
1517: Func HexToBin2(HexNum: Str): Str;
1518: Func HexToDouble( const Hex : Str) : Double
1519: Func HexToInt(hexnum: Str): LongInt;
1520: Func HexToStr(Value: Str): Str;
1521: Func HexifyBlock( var Buffer, BufferSize : Int) : Str
1522: Func Hi(vdat: word): byte;
1523: Func HiByte(W: Word): Byte)
1524: Func High: Int64;
1525: Func HighlightCell(DataCol, DataRow: Int; const Value: str; AState: TGridDrawState) : Bool
1526: Func HINSTANCE: longword;
1527: Func HiWord(l: DWORD): Word)
1528: Func HMODULE: longword;
1529: Func HourOf( const AValue : TDateTime) : Word
1530: Func HourOfTheDay( const AValue : TDateTime) : Word
1531: Func HourOfTheMonth( const AValue : TDateTime) : Word
1532: Func HourOfTheWeek( const AValue : TDateTime) : Word
1533: Func HourOfTheYear( const AValue : TDateTime) : Word
1534: Func HoursBetween( const ANow, AThen : TDateTime) : Int64
1535: Func HourSpan( const ANow, AThen : TDateTime) : Double
1536: Func HSLToRGB1( const H, S, L : Single) : TColor32;
1537: Func HTMLDecode( const AStr : Str) : Str
1538: Func HTMLEncode( const AStr : Str) : Str
1539: Func HTMLEscape( const Str : Str) : Str
1540: Func HtmlTable(DataSet: TDataSet; DataSetHandler: TDSTableProducer; MaxRows: Int): str
1541: Func HTTPDecode( const AStr : Str) : Str
1542: Func HTTPEncode( const AStr : Str) : Str
1543: Func Hypot( const X, Y : Extended) : Extended
1544: Func IBSMax( n1, n2 : Int) : Int
1545: Func IBSMin( n1, n2 : Int) : Int
1546: Func IBRandomString( iLength : Int) : Str
1547: Func IBRandomInt( iLow, iHigh : Int) : Int
1548: Func IBStripString( st : Str; CharsToStrip : Str) : Str
1549: Func IBFormatIdentifier( Dialect : Int; Value : Str) : Str
1550: Func IBFormatIdentifierValue( Dialect : Int; Value : Str) : Str
1551: Func IBExtractIdentifier( Dialect : Int; Value : Str) : Str
1552: Func IBQuoteIdentifier( Dialect : Int; Value : Str) : Str
1553: Func IBAddIBParamsSQLForDetail(Params: TParams; SQL: str; Native: Boolean; Dialect: Int): str
1554: Proc IBDecomposeDatabaseName(DatabaseName: Str; var ServerName, Protocol, DatabasePath: str)
1555: Func RandomString( iLength : Int) : Str;
1556: Func RandomInt( iLow, iHigh : Int) : Int;
1557: Func StripString( st : Str; CharsToStrip : Str) : Str;
1558: Func Strip(const SubString: Str; MainString: Str): Str;
1559: Func StripTags(const S: Str): Str; //<'> of HTML
1560: Func SizeToString(size : Int64; const unitStr : Str) : Str;

```

```

1561: Func NumberToString(No: Word):Str;
1562: Func FormatIdentifier( Dialect : Int; Value :Str) :Str;
1563: Func FormatIdentifierValue( Dialect : Int; Value :Str) :Str;
1564: Func ExtractIdentifier( Dialect : Int; Value :Str) :Str;
1565: Func QuoteIdentifier( Dialect : Int; Value :Str) :Str;
1566: Func AddIBParamSQLForDetail(Params:TParams;SQL:str;Native:Boolean;Dialect:Int):str;
1567: Proc DecomposeDatabaseName(DatabaseName:str;var ServerName,Protocol,DatabasePath:str;
1568: Func NextSQLToken(var p: PChar; var Token:Str; CurSection: TSQLToken): TSQLToken;
1569: Func IconToBitmap( Ico : HICON ) : TBitmap
1570: Func IconToBitmap2( Ico : HICON; Size : Int; TransparentColor : TColor ) : TBitmap
1571: Func IconToBitmap3( Ico : HICON; Size : Int; TransparentColor : TColor ) : TBitmap
1572: Func IdentToCharset(const Ident:Str; var CharSet: Longint):Bool)
1573: Func IdentToColor(const Ident:Str; var Color: Longint):Bool)
1574: Func IdentToCursor(const Ident:Str; var cursor: Longint):Bool;
1575: Func IdGetDefaultCharSet : TidCharSet
1576: Func IDispatchInvoke(Self:IDispatch;ProperSet:Bool;const Name:Str;Par:array of variant):variant
1577: Func IdPorts2 : TStringList
1578: Func IdToMib( const Id :Str) :Str
1579: Func IdSHA1Hash(aphath:Str):Str;
1580: Func IdHashSHA1(aphath:Str):Str;
1581: Func IfStr(const bCondition:boolean;const sTrue:str;const sFalse:str):Str
1582: Func IfThen( AValue :Bool; const ATrue :Str; AFalse :Str) :Str;
1583: Func IfThenInt( AValue :Bool; const ATrue : Int; AFalse : Int): Int;;
1584: Func IfThenDouble( AValue :Bool; const ATrue : double; AFalse : double): double;;
1585: Func IfThenBool( AValue :Bool; const ATrue :Bool; AFalse :Bool):Bool;;
1586: Func iif1( ATest :Bool; const ATrue : Int; const AFalse : Int) : Int;
1587: Func iif2( ATest :Bool; const ATrue :Str; const AFalse :Str) :Str;
1588: Func iif3( ATest:Boolean; const ATrue:Bool;const AFalse:Bool):Bool;
1589: Func ImportTest(S1:Str;s2:longint; s3:Byte; s4:word; var s5:str):Str;
1590: Func IncDay( const AValue : TDateTime; const ANumberOfDays : Int) : TDateTime
1591: Func IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1592: Func IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1593: Func IncLimit1(var B:Shortint;const Limit:Shortint;const Incr:Shortint):Shortint;
1594: Func IncLimit2(var B:Smallint;const Limit:Smallint;const Incr:Smallint):Smallint;
1595: Func IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1596: Func IncLimit4( var B : Int; const Limit : Int; const Incr : Int) : Int;
1597: Func IncLimit5(var B:Card;const Limit:Card;const Incr:Card):Card;
1598: Func IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1599: Func IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1600: Func IncLimitClamp1(var B: Shortint;const Limit:Shortint;const Incr: Shortint): Shortint;
1601: Func IncLimitClamp2(var B:Smallint;const Limit:Smallint;const Incr: Smallint): Smallint;
1602: Func IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1603: Func IncLimitClamp4( var B : Int; const Limit : Int; const Incr : Int) : Int;
1604: Func IncLimitClamp5(var B:Card;const Limit:Card; const Incr:Card):Card;
1605: Func IncLimitClamp6(var B: Int64;const Limit:Int64; const Incr:Int64):Int64;
1606: Func IncludeTrailingBackslash( S :Str) :Str
1607: Func IncludeTrailingBackslash(const S:Str):Str)
1608: Func IncludeTrailingPathDelimiter( const APath :Str) :Str
1609: Func IncludeTrailingPathDelimiter( S :Str) :Str
1610: Func IncludeTrailingPathDelimiter(const S:Str):Str)
1611: Func IncludeTrailingSlash( const APath :Str) :Str
1612: Func IncMilliSecond(const AValue:TDateTime;const ANumberOfMilliSeconds:Int64):TDateTime
1613: Func IncMinute(const AValue:TDateTime;const ANumberOfMinutes:Int64):TDateTime
1614: Func IncMonth( DateTime : TDateTime; NumberOfMonths : Int) : TDateTime
1615: Func IncMonth(const DateTime: TDateTime; NumberOfMonths: Int): TDateTime)
1616: Func IncSecond(const AValue:TDateTime;const ANumberOfSeconds: nt64) : TDateTime
1617: Func IncWeek(const AValue : TDateTime; const ANumberOfWeeks : Int) : TDateTime
1618: Func IncYear(const AValue : TDateTime; const ANumberOfYears : Int) : TDateTime
1619: Func IndexOf(AClass : TClass) : Int
1620: Func IndexOf(AComponent : TComponent) : Int
1621: Func IndexOf(AObject : TObject) : Int
1622: Func INDEXOF(const ANAME :Str) : Int
1623: Func IndexOf(const DisplayName :Str) : Int
1624: Func IndexOf(const Item : TBookmarkStr) : Int
1625: Func IndexOf(const S : WideString) : Int
1626: Func IndexOf(const View : TJclFileMappingView) : Int
1627: Func INDEXOF( FIELD : TFIELD) : Int
1628: Func IndexOf( ID : LCID) : Int
1629: Func INDEXOF( ITEM : TMENUITEM) : Int
1630: Func IndexOf(Value : TListItem) : Int
1631: Func IndexOf(Value : TTreeNode) : Int
1632: Func IndexOf(const S:Str) : Int;
1633: Func IndexOfName( const Name : WideString) : Int
1634: Func IndexOfName(Name:Str): Int;
1635: Func IndexOfObject( AObject : TObject) : Int
1636: Func IndexofObject(AObject:tObject):Int
1637: Func IndexOfTabAt( X, Y : Int) : Int
1638: Func IndexStr( const AText :Str; const AValues : array of string) : Int
1639: Func IndexText( const AText :Str; const AValues : array of string): Int
1640: Func IndexOfInt( AList : TStringList; Value : Variant) : Int
1641: Func IndexOfFloat( AList : TStringList; Value : Variant) : Int
1642: Func IndexOfDate( AList : TStringList; Value : Variant) : Int
1643: Func IndexOfString( AList : TStringList; Value : Variant) : Int
1644: Func IndyCompareStr( const A1 :Str; const A2 :Str) : Int
1645: Func IndyGetHostName :Str
1646: Func IndyInterlockedDecrement( var I : Int) : Int
1647: Func IndyInterlockedExchange( var A : Int; B : Int) : Int
1648: Func IndyInterlockedExchangeAdd( var A : Int; B : Int) : Int
1649: Func IndyInterlockedIncrement( var I : Int) : Int

```



```

1650: Func IndyLowerCase( const A1 :Str) :Str
1651: Func IndyStrToBool( const AString :Str) :Bool
1652: Func IndyUpperCase( const A1 :Str) :Str
1653: Func InitCommonControl( CC : Int) :Bool
1654: Func InitTempPath :Str
1655: Func InMainThread :Bool
1656: Func InOpArray(W : WideChar; sets:array of WideChar):Bool
1657: Func Input: Text
1658: Func InputBox( const ACaption, APrompt, ADefault :Str) :Str
1659: Func InputBox(const ACaption:str;const APrompt:str;const ADefault:str):str
1660: Func InputLn(const AMask:str;AEcho:Boolean;ATabWidth:Int;AMaxLineLength:Int):str
1661: Func InputQuery( const ACaption, APrompt :Str; var Value :Str) :Bool
1662: Func InputQuery(const ACaption:str; const APrompt:str; var Value:Str):Bool
1663: Func InquireSignal( RtlSigNum : Int) : TSignalState
1664: Func InRangeR( const A, Min, Max : Double) :Bool
1665: Func Insert(Index:Int):TCollectionItem
1666: Func Insert(Index:Int):TComboExItem
1667: Func Insert(Index:Int):THeaderSection
1668: Func Insert(Index:Int):TListItem
1669: Func Insert(Index:Int):TStatusPanel
1670: Func Insert(Index:Int):TWorkArea
1671: Func Insert(Index:LongInt; const Text :Str) : LongInt
1672: Func Insert(Sibling : TTreeNode; const S :Str) : TTreeNode
1673: Func INSERTNEWLINEAFTER( AITEM : TMENUITEM) : Int
1674: Func INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : Int
1675: Func InsertNode( Node, Sibling : TTreeNode; const S :Str; Ptr : Pointer) : TTreeNode
1676: Func InsertObject( Index : LongInt; const Text :Str; const Data : Pointer) : LongInt
1677: Func InsertObject( Sibling : TTreeNode; const S :Str; Ptr : Pointer) : TTreeNode
1678: Func Instance : Longint
1679: Func InstanceSize: Longint
1680: Func Int(e : Extended) : Extended;
1681: Func Int64ToStr(i: Int64):Str;
1682: Func IntToBcd( const AValue : Int) : TBcd
1683: Func Intensity( const Color32 : TColor32) : Int;
1684: Func Intensity( const R, G, B : Single) : Single;
1685: Func InterestPayment(const Rate:Ext;Period,NPeriods:Int;const PresentValue,FutureValue:Ext;PaymentTime
:TPaymentTime):Ext
1686: Func InterestRate(NPeriods:Int;const Payment,PresVal,FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1687: Func InternalDecodeDate( DateTime:TDateTime;var Year,Month,Day,DOW:Word):Boolean
1688: Func InternalRateOfReturn(const Guess:Extended;const CashFlows:array of Double):Extended
1689: Func InternalUpdateRecord( Tree : TUpdateTree) :Bool
1690: Func IntersectRect( out Rect : TRect; const R1, R2 : TRect) :Bool
1691: Func IntersectRect(out Rect: TRect; const R1, R2: TRect):Bool
1692: Func IntMibToStr( const Value :Str) :Str
1693: Func IntPower( const Base : Extended; const Exponent : Int) : Extended
1694: Func IntToBin( Value :Card) :Str
1695: Func IntToHex( Value : Int; Digits : Int) :Str;
1696: Func IntToHex(a: Int; b: Int):Str;
1697: Func IntToHex64( Value : Int64; Digits : Int) :Str;
1698: Func IntToHex64(Value: Int64; Digits: Int):Str
1699: Func IntTo3Str( Value : Longint; separator:Str) :Str
1700: Func inttobool( aInt : LongInt) :Bool
1701: Func IntToStr(i: Int64):Str;
1702: Func IntToStr64(Value: Int64):Str
1703: Func IOResult: Int
1704: Func IPv6AddressToStr(const AValue: TIdIPv6Address):Str
1705: Func IPAddrToHostName(const IP:Str):Str;
1706: Func IsAccel(VK: Word; const Str:Str):Bool
1707: Func IsAddressInNetwork( Address :Str) : Bool
1708: Func IsAdministrator :Bool
1709: Func IsAlias( const Name :Str) :Bool
1710: Func IsApplicationRunning( const AClassName, ApplName:str):Bool
1711: Func IsASCII( const AByte : Byte) :Bool;
1712: Func IsASCIILDH( const AByte : Byte) :Bool;
1713: Func IsAssembly(const FileName:Str):Bool;
1714: Func IsBcdNegative( const Bcd : TBcd) :Bool
1715: Func IsBinary(const AChar : Char) :Bool
1716: Func IsConsole:Bool
1717: Func IsDelimiter( Delimiters, S :Str; Index : Int) :Bool
1718: Func IsDelimiter(const Delimiters:Str;const S:Str;Index: Int):Boolean
1719: Func IsDelphiDesignMode :Bool
1720: Func IsDelphiRunning :Bool
1721: Func IsDFAState :Bool
1722: Func IsDirectory( const FileName :Str) :Bool
1723: Func IsDomain( const S :Str) :Bool
1724: Func IsDragObject(Sender: TObject):Bool;
1725: Func IsEditing :Bool
1726: Func ISEMPTY :Bool
1727: Func IsEqual( Value : TParameters) :Bool
1728: Func ISEQUAL( VALUE : TPARAMS) :Bool
1729: Func IsEqualGUID(const guid1, guid2: TGUID):Bool
1730: Func IsFirstNode :Bool
1731: Func IsFloatZero( const X : Float) :Bool
1732: Func IsFormatRegistered( Extension, AppID :Str) :Bool
1733: Func IsFormOpen(const FormName:Str):Bool;
1734: Func IsFQDN( const S :Str) :Bool
1735: Func IsGrayScale :Bool
1736: Func IsHex( AChar : Char) :Bool;
1737: Func IsHexString(const AString:Str):Bool;

```

```

1738: Func IsHostname( const S :Str) :Bool
1739: Func IsInfinite( const AValue : Double) :Bool
1740: Func IsInLeapYear( const AValue : TDateTime) :Bool
1741: Func IsInternet:Bool;
1742: Func IsLeadChar( ACh : Char) :Bool
1743: Func IsLeapYear( Year : Word) :Bool
1744: Func IsLeapYear(Year: Word):Bool)
1745: Func IsLibrary:Bool)
1746: Func ISLINE :Bool
1747: Func IsLinkedTo( DataSet : TDataSet) :Bool
1748: Func ISLINKEDTO( DATASOURCE : TDATASOURCE) :Bool
1749: Func IsLiteralChar( const EditMask :Str; Offset : Int) :Bool
1750: Func IsMatch( const Pattern, Text :Str) :Bool //Grep like RegEx
1751: Func IsMainAppWindow( Wnd : HWND) :Bool
1752: Func IsMediaPresentInDrive( Drive : Char) :Bool
1753: Func IsMemoryManagerSet:Bool)
1754: Func IsMultiTableQuery( const SQL : WideString) :Bool
1755: Func IsMultiThread:Bool)
1756: Func IsNumeric( AChar : Char) :Bool;
1757: Func IsNumeric2( const AString :Str) :Bool;
1758: Func IsNTFS:Bool;
1759: Func IsOctal( AChar : Char) :Bool;
1760: Func IsOctalString(const AString:Str) :Bool;
1761: Func IsPathDelimiter( S :Str; Index : Int) :Bool
1762: Func IsPathDelimiter(const S:Str; Index: Int):Bool)
1763: Func IsPM( const AValue : TDateTime) :Bool
1764: Func IsPositiveFloatArray( const X : TDynFloatArray) :Bool
1765: Func IsPortAvailable( ComNum :Card) :Bool;
1766: Func IsCOMPortReal( ComNum :Card) :Bool;
1767: Func IsCOM( ComNum :Card) :Bool;
1768: Func IsCOMPort:Bool;
1769: Func IsPrimeFactor( const F, N :Card) :Bool
1770: Func IsPrimeRM( N :Card) :Bool //rabin miller
1771: Func IsPrimeTD( N :Card) :Bool //trial division
1772: Func IsPrivilegeEnabled( const Privilege :Str) :Bool
1773: Func ISqrt( const I : Smallint) : Smallint
1774: Func IsReadOnly(const Filename:Str):Bool;
1775: Func IsRectEmpty( const Rect : TRect) :Bool
1776: Func IsRectEmpty(const Rect: TRect):Bool)
1777: Func IsRelativePrime( const X, Y :Card) :Bool
1778: Func ISRIGHTTOLEFT :Bool
1779: Func IsRightToLeft:Bool
1780: Func IsSameDay( const AValue, ABasis : TDateTime) :Bool
1781: Func ISSEQUENCED :Bool
1782: Func IsSystemModule( const Module : HMODULE) :Bool
1783: Func IsSystemResourcesMeterPresent :Bool
1784: Func IsTCPPortOpen(dwPort : Word; ipAddressStr:Str):Bool;
1785: Func IsToday( const AValue : TDateTime) :Bool
1786: Func IsToday(const AValue: TDateTime):Bool;
1787: Func IsTopDomain( const AStr :Str) :Bool
1788: Func IsUTF8LeadByte( Lead : Char) :Bool
1789: Func IsUTF8String( const s : UTF8String) :Bool
1790: Func IsUTF8TrailByte( Lead : Char) :Bool
1791: Func ISVALIDCHAR( INPUTCHAR : CHAR) :Bool
1792: Func IsValidDate( const AYear, AMonth, ADay : Word) :Bool
1793: Func IsValidDateDay( const AYear, ADayOfYear : Word) :Bool
1794: Func IsValidDateMonthWeek(const AYear,AMonth,AWeekOfMonth,ADayOfWeek:Word) : Bool
1795: Func IsValidDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond:Word) : Bool
1796: Func IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) :Bool
1797: Func IsValidIdent( Ident :Str) :Bool
1798: Func IsValidIdent1(const Ident:Str; AllowDots:Bool):Bool)
1799: Func IsValidIP( const S :Str) :Bool
1800: Func IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) :Bool
1801: Func IsValidPNG(stream: TStream):Bool;
1802: Func IsValidJPEG(stream: TStream):Bool;
1803: Func IsValidISBN( const ISBN : Ansistr) :Bool
1804: Func IsVariantManagerSet:Bool; //deprecated;
1805: Func IsVirtualPcGuest :Bool;
1806: Func IsVmWareGuest :Bool;
1807: Func IsVCLControl(Handle: HWND):Bool;
1808: Func IsWhiteString( const AStr :Str) :Bool
1809: Func IsWindowResponding( Wnd : HWND; Timeout : Int) :Bool
1810: Func IsWow64:Bool;
1811: Func IsWin64:Bool;
1812: Func IsWow64String(var s:Str):Bool;
1813: Func IsWin64String(var s:Str):Bool;
1814: Func IsWindowsVista:Bool;
1815: Func isPowerof2(num: int64):Bool;
1816: Func powerOf2(exponent: Int): int64;
1817: Func IsZero(const A: Extended; Epsilon: Extended):Bool //overload;
1818: Func IsZero1(const A: Double; Epsilon: Double):Bool //overload;
1819: Func IsZero2(const A: Single; Epsilon: Single):Bool //overload;
1820: Func ItemAtPos( Pos : TPoint; IgnoreTabHeight :Bool) : Int
1821: Func ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):Int
1822: Func ItemRect( Index : Int) : TRect
1823: Func ITEMRECT(INDEX:Int):TRECT
1824: Func ItemWidth( Index : Int) : Int
1825: Func JavahashCode(val:Str): Int;
1826: Func JosephusG(n,k: Int; var graphout:Str): Int;

```

```

1827: Func JulianDateToDateTime( const AValue : Double) : TDateTime
1828: Func JustName(PathName :Str) :Str; //in path and ext
1829: Func JvMessageBox( const Text, Caption :Str; Flags : DWORD) : Int;
1830: Func JvMessageBox1( const Text :Str; Flags : DWORD) : Int;
1831: Func KeepAlive :Bool
1832: Func KeysToShiftState(Keys: Word) : TShiftState;
1833: Func KeyDataToShiftState(KeyData: Longint): TShiftState;
1834: Func KeyboardStateToShiftState2(const KeyboardState: TKeyboardState):TShiftState;
1835: Func KeyboardStateToShiftState: TShiftState; overload;
1836: Func Languages : TLanguages
1837: Func Last: TClass
1838: Func Last: TComponent
1839: Func Last: TObject
1840: Func LastDelimiter( Delimiters, S :Str) : Int
1841: Func LastDelimiter(const Delimiters:Str; const S:Str): Int)
1842: Func LastPos( const ASubstr :Str; const AStr :Str) : Int
1843: Func Latitude2WGS84(lat: double): double;
1844: Func LCM(m,n:longint):longint;
1845: Func LCMJ( const X, Y :Card) :Card
1846: Func Ldexp( const X : Extended; const P : Int) : Extended
1847: Func LeftStr( const AText : Ansistr; const ACount : Int) : Ansistr;
1848: Func LeftStr( const AText : WideString; const ACount : Int) : WideString;
1849: Func Length: Int;
1850: Proc LetFileList(FileList: TStringlist; apath:Str);
1851: Func lengthmp3(mp3path:Str):Int;
1852: Func LineInRect( const P1, P2 : TPoint; const Rect : TRect) :Bool;
1853: Func LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) :Bool;
1854: Func LinesCount(sfilename:str):extended;
1855: Func TextFileLineCount(const FileName:Str): Int;
1856: Func LineSegmentIntersection(const L1P1:TFloatPoint;L1P2:TFloatPoint;const L2P1:TFloatPoint;
L2P2:TFloatPoint;var P:TFloatPoint):Boolean
1857: Func LineStart(Buffer, BufPos: PChar): PChar
1858: Func LineStart(Buffer, BufPos: PChar): PChar)
1859: Func ListSeparator: char;
1860: Func Ln(x: Extended): Extended;
1861: Func LnXP1( const X : Extended) : Extended
1862: Func Lo(vdat: word): byte;
1863: Func LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1864: Func LoadedModulesList(const List:TStrings;ProcessID:DWORD;HandlesOnly:Boolean):Bool
1865: Func LoadFileAsString( const FileName :Str) :Str
1866: Func LoadFromFile( const FileName :Str) : TBitmapLoader
1867: Func LoadFile(const FileName: TFileName):Str;
1868: Func LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1869: Func LoadPackage(const Name:Str): HMODULE
1870: Func LoadResource( ModuleHandle : HMODULE; ResHandle: TResourceHandle):HGLOBAL
1871: Func LoadStr( Ident : Int) :Str
1872: Func LoadString(Instance: Longint; IDent: Int; Buffer: PChar; Size: Int):Int;stdcall;
1873: Func LoadWideStr( Ident : Int) : WideString
1874: Func LOCATE(const KEYFIELDS:Str;const KEYVALUES:VARIANT;OPTIONS:TLOCATEOPTIONS):BOOLEAN
1875: Func LockRegion(libOffset:Longint;cb:Largeint;dwLockType:Longint): HRESULT
1876: Func LockServer( fLock : LongBool) : HRESULT
1877: Func LockVolume( const Volume :Str; var Handle : THandle) :Bool
1878: Func Log( const X : Extended) : Extended
1879: Func Log10( const X : Extended) : Extended
1880: Func Log2( const X : Extended) : Extended
1881: Func LogBase10(X: Float): Float;
1882: Func LogBase2(X: Float): Float;
1883: Func LogBaseN(Base, X: Float): Float;
1884: Func LogN( const Base, X : Extended) : Extended
1885: Func LogOffOS :Bool
1886: Func LoginDialog( const ADatabaseName :Str; var AUserName, APassword :Str):Bool
1887: Func LoginDialogEx(const ADbName:str;var AUserName,APassword:str;NameReadOnly:Bool):Bool;
1888: Func LongDateFormat:Str;
1889: Func LongTimeFormat:Str;
1890: Func LongWordToFourChar( ACardinal : LongWord) :Str
1891: Func LOOKUP(const KEYFIELDS:Str;const KEYVALUES:VARIANT;const RESULTFIELDS:str):VARIANT
1892: Func LookupName( const name :Str) : TInAddr
1893: Func LookupService( const service :Str) : Int
1894: Func Low: Int64;
1895: Func LowerCase( S :Str) :Str
1896: Func LowerCase(s : AnyString) : AnyString;
1897: Func LRot( const Value : Byte; const Count : TBitRange) : Byte;
1898: Func LRot1( const Value : Word; const Count : TBitRange) : Word;
1899: Func LRot2( const Value : Int; const Count : TBitRange) : Int;
1900: Func MainInstance: longword
1901: Func MainThreadID: longword
1902: Func Map(x, in_min, in_max, out_min, out_max: Int): Int; //arduino
1903: Func mapMax(ax, in_min, in_max, out_min, out_max: Int): Int;
1904: Func MakeCanonicalIPv4Address( const AAddr :Str) :Str
1905: Func MakeCanonicalIPv6Address( const AAddr :Str) :Str
1906: Func MakeDIB( out Bitmap : PBitmapInfo) : Int
1907: Func MakeDWordIntoIPv4Address( const ADWord :Card) :Str
1908: Func MakeFile(const FileName:Str): Int);
1909: Func MakeLong(A, B: Word): Longint)
1910: Func MakeTempFilename( const APath :Str) :Str
1911: Func MakeValidFileName( const Str :Str) :Str
1912: Func MakeValueMap( Enumeration :Str; ToCds :Bool) :Str
1913: Func MakeWord(A, B: Byte): Word)
1914: Func MakeYear4Digit( Year, Pivot : Int) : Int

```

```

1915: Func MapDateTime( const DateFormatType: str; DateFormat: str; Value: str; ToCds: Bool ): str
1916: Func MapValues( Mapping : Str; Value : Str ) : Str
1917: Func MaskDoFormatText( const EditMask: str; const Value: str; Blank: Char ): str
1918: Func MaskGetCharType( const EditMask : Str; MaskOffset : Int ) : TMaskCharType
1919: Func MaskGetCurrentDirectives( const EditMask : Str; MaskOffset: Int ): TMaskDirectives
1920: Func MaskGetFldSeparator( const EditMask : Str ) : Int
1921: Func MaskGetMaskBlank( const EditMask : Str ) : Char
1922: Func MaskGetMaskSave( const EditMask : Str ) : Bool
1923: Func MaskIntLiteralToChar( IChar : Char ) : Char
1924: Func MaskOffsetToOffset( const EditMask : Str; MaskOffset : Int ) : Int
1925: Func MaskOffsetToWideOffset( const EditMask : Str; MaskOffset : Int ) : Int
1926: Func MaskString( Mask, Value : Str ) : Str
1927: Func Match( const sString : Str ) : TniRegularExpressionMatchResul
1928: Func Match1( const sString: Str; iStart: Int ): TniRegularExpressionMatchResult
1929: Func Matches( const Filename : Str ) : Bool
1930: Func MatchesMask( const Filename, Mask : Str ) : Bool
1931: Func MatchStr( const AText : Str; const AValues : array of string ): Bool
1932: Func MatchText( const AText : Str; const AValues : array of string ): Bool
1933: Func Max( AValueOne, AValueTwo : Int ) : Int
1934: Func Max( const x, y: Int ): Int;
1935: Func Max1( const B1, B2 : Shortint ) : Shortint;
1936: Func Max2( const B1, B2 : Smallint ) : Smallint;
1937: Func Max3( const B1, B2 : Word ) : Word;
1938: Func Max3( const x, y, z: Int ): Int;
1939: Func Max4( const B1, B2 : Int ) : Int;
1940: Func Max5( const B1, B2 : Card ) : Card;
1941: Func Max6( const B1, B2 : Int64 ) : Int64;
1942: Func Max64( const AValueOne, AValueTwo : Int64 ) : Int64
1943: Func MaxFloat( const X, Y : Float ) : Float
1944: Func MaxFloatArray( const B : TDynFloatArray ) : Float
1945: Func MaxFloatArrayIndex( const B : TDynFloatArray ) : Int
1946: Func MaxIntValue( const Data: array of Int ): Int
1947: Func MaxJ( const B1, B2 : Byte ) : Byte;
1948: Func MaxPath: Str;
1949: Func MaxValue( const Data: array of Double ): Double
1950: Func MaxCalc( const Formula : Str ) : Extended //math expression parser
1951: Proc MaxCalcF( const Formula : Str ); //out to console memo2
1952: Func MaxCalcS( const Formula : Str ): Str;
1953: Func MD5( const fileName: Str ): Str;
1954: Func Mean( const Data : array of Double ) : Extended
1955: Func Median( const X : TDynFloatArray ) : Float
1956: Func Memory : Pointer
1957: Func MemoryPos( const ASubStr : Str; MemBuff : PChar; MemorySize: Int ): Int
1958: Func MessageBox( hndl: Card; text, caption: Str; utype: Card ): Int;
1959: Func MESSAGEBOX( TEXT, CAPTION: PCHAR; FLAGS: WORD ): Int
1960: Func MessageDlg( const Msg: str; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint ): Int;
1961: Func MessageDlg1( const
Msg: str; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint; DefaultButton: TMsgDlgBtn ): Int;
1962: Func MessageDlgPos( const Msg: str; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint; X, Y: Int ): Int;
1963: Func MessageDlgPos1( const Msg: str; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint; X, Y:
Int; DefaultButton: TMsgDlgBtn ): Int;
1964: Func MessageDlgPosHelp( const Msg: str; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint; X, Y:
Int; const HelpFileName: str ): Int;
1965: Func MessageDlgPosHelp1( const Msg: str; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint; X, Y :
Int; const HelpFileName: str; DefaultButton: TMsgDlgBtn ): Int;
1966: Func MibToId( Mib : Str ) : Str
1967: Func MidStr( const AText : Ansistr; const AStart, ACount : Int ) : Ansistr;
1968: Func MidStr( const AText : WideString; const AStart, ACount : Int ) : WideString;
1969: Func microsecondsToCentimeters( mseconds: longint ): longint; //340m/s speed of sound
1970: Func Micros( const Timer: THPTimer; const TimerRunning: Boolean ): Int64 //TypeS('THPTimer', 'Int64
1971: Func MIDIOut( DeviceID : Card ) : IJclMIDIOut
1972: Proc GetMidiOutputs( const List : TStrings)
1973: // GetGEOmapX('html', ExePath+'cologne2mapX.html', 'cathedral cologne
1974: Proc GetGEOmap( C_form, apath: str; const Data: Str ); //c_form: [html/json/xml]
1975: Func MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single ) : TSingleNoteTuningData
1976: Func MIDINoteToStr( Note : TMIDINote ) : Str
1977: Func WinMidiOut( DeviceID : Card ) : IJclWinMidiOut
1978: Proc GetMidiOutputs( const List : TStrings)
1979: Proc MidiOutCheck( Code : MMRResult)
1980: Proc MidiInCheck( Code : MMRResult)
1981: Func MilliSecondOf( const AValue : TDateTime ) : Word
1982: Func MilliSecondOfDay( const AValue : TDateTime ): LongWord
1983: Func MilliSecondOfTheHour( const AValue : TDateTime ): LongWord
1984: Func MilliSecondOfTheMinute( const AValue : TDateTime ): LongWord
1985: Func MilliSecondOfTheMonth( const AValue : TDateTime ): LongWord
1986: Func MilliSecondOfTheSecond( const AValue : TDateTime ): Word
1987: Func MilliSecondOfTheWeek( const AValue : TDateTime ): LongWord
1988: Func MilliSecondOfTheYear( const AValue : TDateTime ): Int64
1989: Func MilliSecondsBetween( const ANow, AThen : TDateTime ): Int64
1990: Func MilliSecondSpan( const ANow, AThen : TDateTime ): Double
1991: Func milliToDateTime( MilliSecond : LongInt ) : TDateTime;
1992: Func Micros( const Timer : THPTimer; const TimerRunning: Boolean ): Int64
1993: Func millis: int64;
1994: Func Min( AValueOne, AValueTwo : Int ) : Int
1995: Func Min1( const B1, B2: Shortint ) : Shortint;
1996: Func Min2( const B1, B2: Smallint ) : Smallint;
1997: Func Min3( const B1, B2: Word ) : Word;
1998: Func Min4( const B1, B2: Int ) : Int;
1999: Func Min5( const B1, B2: Card ) : Card;

```

```

2000: Func Min6(const B1,B2: Int64) : Int64;
2001: Func Min64( const AValueOne, AValueTwo : Int64) : Int64
2002: Func MinClientRect : TRect;
2003: Func MinClientRect1( IncludeScroller :Bool) : TRect;
2004: Func MinClientRect2( TabCount : Int; IncludeScroller :Bool) : TRect;
2005: Func MinFloat( const X, Y : Float) : Float
2006: Func MinFloatArray( const B : TDynFloatArray) : Float
2007: Func MinFloatArrayIndex( const B : TDynFloatArray) : Int
2008: Func MinimizeName(const Filename:str;Canvas : TCanvas; MaxLen:Int):Str
2009: Func MinimizeName(const Filename:TFileName;Canvas:TCanvas; MaxLen:Int):FileName
2010: Func MinimizeName(const Filename:Str; Canvas: TCanvas;MaxLen: Int):TFileName
2011: Func MinIntValue( const Data : array of Int) : Int
2012: Func MinIntValue(const Data: array of Int):Int)
2013: Func MinJ( const B1, B2 : Byte) : Byte;
2014: Func MinuteOf( const AValue : TDateTime) : Word
2015: Func MinuteOfTheDay( const AValue : TDateTime) : Word
2016: Func MinuteOfTheHour( const AValue : TDateTime) : Word
2017: Func MinuteOfTheMonth( const AValue : TDateTime) : Word
2018: Func MinuteOfTheWeek( const AValue : TDateTime) : Word
2019: Func MinuteOfTheYear( const AValue : TDateTime) : LongWord
2020: Func MinutesBetween( const ANow, AThen : TDateTime) : Int64
2021: Func MinuteSpan( const ANow, AThen : TDateTime) : Double
2022: Func MinValue( const Data : array of Double) : Double
2023: Func MinValue(const Data: array of Double): Double)
2024: Func MixerLeftRightToArray( Left, Right :Card) : TDynCardinalArray
2025: Func MMCheck( const MciError : MCIERROR; const Msg :Str) : MCIERROR
2026: Func ModFloat( const X, Y : Float) : Float
2027: Func ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
2028: Func Modify( const Key :Str; Value : Int) : Bool
2029: Func ModuleCacheID :Card
2030: Func ModuleFromAddr( const Addr : Pointer) : HMODULE
2031: Func MonitorFromPoint(const Point:TPoint;MonitorDefault:TMonitorDefaultTo): TMonitor
2032: Func MonitorFromRect(const Rect:TRect;MonitorDefault:TMonitorDefaultTo):TMonitor
2033: Func MonitorFromWindow(const Handle:THandle; MonitorDefault:TMonitorDefaultTo): TMonitor
2034: Func MonthOf( const AValue : TDateTime) : Word
2035: Func MonthOfTheYear( const AValue : TDateTime) : Word
2036: Func MonthsBetween( const ANow, AThen : TDateTime) : Int
2037: Func MonthSpan( const ANow, AThen : TDateTime) : Double
2038: Func MonthStr( DateTime : TDateTime) :Str
2039: Func MouseCoord( X, Y : Int) : TGridCoord
2040: Func MOVEBY( DISTANCE : Int) : Int
2041: Func MoveFile( Source, Dest :Str; Flags : FILEOP_FLAGS) :Bool
2042: Func MoveNext :Bool
2043: Func MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
2044: Func MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
2045: Func Name :Str
2046: Func NetPresentValue(const Rate:Extended;const CashFlows:array of Double;PaymentTime:TPaymentTime):Extended
2047: Func NetworkVolume(DriveChar: Char):Str
2048: Func NEWBOTOMLINE : Int
2049: Func NewCompareNode(Field: TField;Operator:TCANOperator; const Value:Variant) : PExprNode
2050: Func NEWITEM(const ACAPTION:str;AShortcut:TShortcut;AChecked,AEnabled:BOOL;AOnClick:
TNotifyEvent;HCTX:WORD;const ANAME:str):TMenuItem
2051: Func NEWLINE : TMenuItem
2052: Func NEWMENU( OWNER : TComponent;const ANAME:str;ITEMS: array of TMenuItem): TMAINMENU
2053: Func NewNode(Kind: TExprNodeKind;Operator:TCANOperator;const Data:Variant;Left,Right:PExprNode):PExprNode
2054: Func NEWPOPUPMENU(OWNER:TComponent;const ANAME:str; ALIGNMENT:TPOPOPALIGNMENT;AUTOPOPUP:BOOLEAN; const
ITEMS : array of TCMENUITEM) : TPOPOPUPMENU
2055: Func NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
2056: Func NEWSUBMENU(const ACAPT:str;HCTX:WORD;const ANAME:str;ITEMS:array of TMenuItem;AEnabled:BOOL):
TMenuItem
2057: Func NEWTOPLINE : Int
2058: Func Next : TIdAuthWhatsNext
2059: Func NextCharIndex( S :Str; Index : Int) : Int
2060: Func NextRecordSet : TCustomSQLDataSet
2061: Func NextRecordset( var RecordsAffected : Int) : _Recordset
2062: Func NextSQLToken1(var p: WideChar;out Token:WideString;CurSection:TSQLToken):TSQLToken;
2063: Func NextToken : Char
2064: Func nextToken : WideString
2065: Func NextToken:Char
2066: Func Norm( const Data : array of Double) : Extended
2067: Func NormalizeAngle( const Angle : Extended) : Extended
2068: Func NormalizeBcd(const InBcd TBcd;var OutBcd:TBcd;const Prec,Scale Word):Bool
2069: Func NormalizeRect( const Rect : TRect) : TRect
2070: Func NormalizeRect(const Rect: TRect): TRect;
2071: Func Now :TDateTime
2072: Func Now2:tDateTime
2073: Func NumProcessThreads:Int
2074: Func NumThreadCount : Int
2075: Func NthDayOfWeek( const AValue : TDateTime) : Word
2076: Func NtProductType : TntProductType
2077: Func NtProductTypeString :Str
2078: Func Null: Variant;
2079: Func NullPoint : TPoint
2080: Func NullRect : TRect
2081: Func Null2Blank(aString:str):str;
2082: Func NumberOfPeriods(const Rate:Extended;Payment:Extended;const PresentValue,
FutureValue:Extended;PaymentTime:TPaymentTime):Extended
2083: Func NumIP:Int
2084: Func Odd(x: Longint):Bool;

```



```

2085: function OleVariantToString(const Value: OleVariant): string;
2086: function StringToOleVariant(const Value: string): OleVariant;
2087: function OleVariantToMemoryStream(const OV: OleVariant): TMemoryStream;
2088: Func OffsetFromUTC : TDateTime
2089: Func OffsetPoint( const P, Offset : TPoint ) : TPoint
2090: Func OffsetRect( var Rect : TRect; DX: Int; DY : Int) :Bool
2091: Func OffsetRect(var Rect: TRect; DX: Int; DY: Int): Bool
2092: Func OffsetToMaskOffset( const EditMask :Str; Offset : Int) : Int
2093: Func OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment): Bool
2094: Func OldBCDToCurr( const BCD : TBcd; var Curr : Currency) :Bool
2095: Func OldCurrToBCD(const Curr: Currency; var BCD: TBcd; Precision: Int; Decimals: Int): Bool
2096: Func OpenBit: Int
2097: Func OpenDatabase : TDatabase
2098: Func OpenDatabase( const DatabaseName: str): TDatabase
2099: Proc OpenDir(adir: Str);
2100: Func OpenGLColorToWinColor( const Red, Green, Blue: Float): TColor
2101: Func OpenMap(const Data: Str): Bool;
2102: Func OpenMapX(const Data: Str): Bool;
2103: Func OpenObject( Value : PChar) : Bool;
2104: Func OpenObject1( Value : Str) : Bool;
2105: Proc Openweb('http://snippets.delphidabbler.com/#')
2106: Proc OpenURL( const AText : TKkString);
2107: Proc OpenWeb( const AText : TKkString);
2108: Proc OpenBrowser( const AText : TKkString);
2109: Func OpenSession( const SessionName : Str) : TSession
2110: Func OpenVolume( const Drive : Char) : THandle
2111: Func OrdFourByteToCardinal( AByte1, AByte2, AByte3, AByte4 : Byte): Card
2112: Func OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte): LongWord
2113: Func OrdToBinary( const Value: Byte) : Str;
2114: Func OrdToBinary1(const Value: Shortint) : Str;
2115: Func OrdToBinary2(const Value: Smallint) : Str;
2116: Func OrdToBinary3(const Value: Word) : Str;
2117: Func OrdToBinary4(const Value: Int) : Str;
2118: Func OrdToBinary5(const Value: Card) : Str;
2119: Func OrdToBinary6(const Value: Int64) : Str;
2120: Func OSCheck( RetVal : Bool) : Bool
2121: Func OSFileTypeToString( const OSFileType: DWORD; const OSFileSubType: DWORD): str
2122: Func OSIdentToString( const OSIdent : DWORD) : Str
2123: Func Output: Text
2124: Func Overlay( ImageIndex : Int; Overlay : TOverlay): Boolean
2125: Func Owner : TCustomListView
2126: Func Owner : TPersistent
2127: Func PadInputLiterals(const EditMask: str; const Value: str; Blank: Char): str
2128: Func PadL( pStr : Str; pLth : Int) : Str
2129: Func PadL(s : AnyString; I : longInt) : AnyString;
2130: Func PadLCh( pStr : Str; pLth : Int; pChr : char) : Str
2131: Func PadR( pStr : Str; pLth : Int) : Str
2132: Func PadR(s : AnyString; I : longInt) : AnyString;
2133: Func PadRCh( pStr : Str; pLth : Int; pChr : char) : Str
2134: Func PadString(const AString: Str; const ALen: Int; const AChar: Char): str
2135: Func Padz(s : AnyString; I : longInt) : AnyString;
2136: Func PaethPredictor( a, b, c : Byte) : Byte
2137: Func PARAMBYNAME( const VALUE: Str) : TPARAM
2138: Func ParamByName( const Value : WideString): TParameter
2139: Func ParamCount: Int
2140: Func ParamsEncode( const ASrc : Str): Str
2141: Func ParamStr(Index: Int): Str
2142: Func ParseDate( const DateStr : Str): TDateTime
2143: Func PARSESQL( SQL : Str; DOCREATE : Bool) : Str
2144: Func ParseSQL( SQL : WideString; DoCreate : Bool): WideString
2145: Func PathAddExtension( const Path, Extension : Str): str
2146: Func PathAddSeparator( const Path : Str) : Str
2147: Func PathAppend( const Path, Append : Str) : Str
2148: Func PathBuildRoot( const Drive : Byte) : Str
2149: Func PathCanonicalize( const Path : Str) : Str
2150: Func PathCommonPrefix( const Path1, Path2 : Str) : Int
2151: Func PathCompactPath(const DC: HDC; const Path: str; const Width: Int; CmpFmt: TCompactPath): str;
2152: Func PathCompactPath1(const Canv: TCanvas; const Path: str; const Width: Int; CmpFmt: TCompactPath): str;
2153: Func PathEncode( const ASrc : Str) : Str
2154: Func PathExtractFileDirFixed( const S : Ansistr): Ansistr
2155: Func PathExtractFileNameNoExt( const Path : Str): Str
2156: Func PathExtractPathDepth( const Path : Str; Depth: Int): str
2157: Func PathGetDepth( const Path : Str): Int
2158: Func PathGetLongName( const Path : Str): Str
2159: Func PathGetLongName2( Path : Str): Str
2160: Func PathGetShortName( const Path : Str): Str
2161: Func PathIsAbsolute( const Path : Str): Bool
2162: Func PathIsChild( const Path, Base : Ansistr): Bool
2163: Func PathIsDiskDevice( const Path : Str): Bool
2164: Func PathIsUNC( const Path : Str) : Bool
2165: Func PathRemoveExtension( const Path: Str): Str
2166: Func PathRemoveSeparator( const Path: Str): Str
2167: Func Payment(Rate: Extended; NPeriods: Int; const PresentVal,
FutureVal: Extended; PaymentTime: TPaymentTime): Extended
2168: Func Peek: Pointer
2169: Func Peek: TObject
2170: Func PERFORM(MSG: Card; WPARAM, LPARAM: LONGINT): LONGINT
2171: Func PeriodPayment(const Rate: Extended; Period, NPeriods: Int; const PresentValue, FutureValue:
Extended; PaymentTime: TPaymentTime): Extended

```

```

2172: Func Permutation(npr, k: Int): extended;
2173: Func PermutationInt(npr, k: Int): Int64;
2174: Func PermutationJ( N, R :Card) : Float
2175: Func Pi : Extended;
2176: Func PiE : Extended;
2177: Func PixelsToDialogsX( const Pixels : Word) : Word
2178: Func PixelsToDialogsY( const Pixels : Word) : Word
2179: Func PlaySound(s: pchar; flag,syncflag: Int):Bool;
2180: Func Point( X, Y : Int) : TPoint
2181: Func Point(X, Y: Int): TPoint)
2182: Func PointAssign( const X, Y : Int) : TPoint
2183: Func PointDist( const P1, P2 : TPoint) : Double;
2184: Func PointDist(const P1,P2: TFloatPoint): Double;
2185: Func PointDist1( const P1, P2 : TFloatPoint) : Double;
2186: Func PointDist2(const P1,P2: TPoint): Double;
2187: Func PointEqual( const P1, P2 : TPoint) :Bool
2188: Func PointIsNull( const P : TPoint) :Bool
2189: Func PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint): Double
2190: Func Poly( const X : Extended; const Coefficients : array of Double): Extended
2191: Func PortTCPIsOpen(dwPort : Word; ipAddressStr:Str):bool;
2192: Func IsTCPPortOpen(dwPort : Word; ipAddressStr:Str):bool;
2193: Func Pop : Pointer
2194: Func Pop : TObject
2195: Func PopnStdDev( const Data : array of Double) : Extended
2196: Func PopnVariance( const Data : array of Double) : Extended
2197: Func PopulationVariance( const X : TDynFloatArray) : Float
2198: Func Pos(SubStr, S: AnyString): Longint;
2199: Func PosEqual( const Rect : TRect) :Bool
2200: Func PosEx( const SubStr, S :Str; Offset : Int) : Int
2201: Func PosInSmallIntArray(const ASearchInt:SmallInt;AArray:array of SmallInt):Int
2202: Func PosInStrArray(const SearchStr:str;Contents:array of str;const CaseSensitive:Bool):Int
2203: Func Post1( AURL:str; const ASource : TStrings) :Str;
2204: Func Post2( AURL:str; const ASource : TStream) :Str;
2205: Func Post3( AURL:str; const ASource : TIdMultiPartFormDataStream) :Str;
2206: Func PostData( const UserData : WideString; const CheckSum : DWORD) :Bool
2207: Func PostData( const UserData : WideString; const CheckSum : Int):Bool
2208: Func PostMessage(hWnd:HWND;Msg:longword;wParam: ongint;lParam:longint):Bool;
2209: Func Power( const Base, Exponent : Extended) : Extended
2210: Func PowerBig(aval, n:Int):Str;
2211: Func PowerIntJ( const X : Float; N : Int) : Float;
2212: Func PowerJ( const Base, Exponent : Float) : Float;
2213: Func PowerOffOS :Bool
2214: Func PreformatDateString( Ps :Str) :Str
2215: Func PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2216: Func PrimeFactors( N :Card) : TDynCardinalArray
2217: Func Printer : TPrinter
2218: Func ProcessPath2(const ABasePath:str;const APath:str;const APathDelim:str):str
2219: Func ProcessResponse : TIdHTTPWhatsNext
2220: Func ProduceContent :Str
2221: Func ProduceContentFromStream( Stream : TStream) :Str
2222: Func ProduceContentFromString( const S :Str) :Str
2223: Func ProgIDToClassID(const ProgID:Str): TGUID;
2224: Func PromptDataLinkFile(ParentHandle:THandle;InitialFile:WideString):WideString
2225: Func PromptDataSource(ParentHandle:THandle;InitialString:WideString):WideString
2226: Func PromptForFileName(var AFileName:str;const AFilter:str;const ADefaultExt:str;const ATitle:str;const
AInitialDir:str;SaveDialog:Bool):Bool
2227: Func PromptForFileName(var AFileName:str;const AFilter:str;const ADefaultExt:str;const ATitle:str;const
AInitialDir:Str;SaveDialog:Bool):Bool)
2228: Func PSScriptNeedFile(Sender:TObject;const OrginFileName:Str;var FileName,Output:Str):Bool
2229: Func PtInRect( const Rect : TRect; const P : TPoint) :Bool
2230: Func PtInRect(const Rect: TRect; const P: TPoint):Bool)
2231: Func Push( AItem : Pointer) : Pointer
2232: Func Push( AObject : TObject) : TObject
2233: Func Put1( AURL :Str; const ASource : TStream) :Str;
2234: Func Pythagoras( const X, Y : Extended) : Extended
2235: Func queryDLLInterface( var queryList : TStringList) : TStringList
2236: Func queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2237: Func QueryInterface(const IID: TGUID; out Obj): HRESULT, CdStdCall
2238: Func queryPerformanceCounter2(mse: int64): int64;
2239: //Func QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2240: //Func QueryPerformanceFrequency(mse: int64):Bool;
2241: Func QueryPerformanceCounter(var lcount: Int64):Bool; stdcall;
2242: Func QueryPerformanceFrequency(var lfreq: int64):Bool; stdcall;
2243: Proc QueryPerformanceCounter1(var aC: Int64);
2244: Func QueryPerformanceFrequency1(var freq: int64):Bool;
2245: Func Quote( const ACommand :Str) : SmallInt
2246: Func QuotedStr( S :Str) :Str
2247: Func RadToCycle( const Radians : Extended) : Extended
2248: Func RadToDeg( const Radians : Extended) : Extended
2249: Func RadToDeg( const Value : Extended) : Extended;
2250: Func RadToDeg1(const Value : Double) : Double;
2251: Func RadToDeg2(const Value : Single) : Single;
2252: Func RadToGrad(const Radians : Extended) : Extended
2253: Func RadToGrad(const Value : Extended) : Extended;
2254: Func RadToGrad1( const Value : Double) : Double;
2255: Func RadToGrad2( const Value : Single) : Single;
2256: Func RandG( Mean, StdDev : Extended) : Extended
2257: Func Random(const ARange: Int): Int;

```

```

2258: Func random2(a: Int): double
2259: Func RandomE: Extended;
2260: Func RandomF: Extended;
2261: Func RandomFrom( const AValues : array of string):Str;
2262: Func RandomRange( const AFrom, ATo : Int) : Int
2263: Func randSeed: longint
2264: Proc setRandseed; //no params!
2265: Func RawToDataColumn( ACol : Int) : Int
2266: Func Read : Char
2267: Func Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HRESULT
2268: Func Read(Buffer:Str;Count:LongInt):LongInt
2269: Func ReadBinaryStream( const Section, Name :Str; Value : TStream) : Int
2270: Func ReadBool( const Section, Ident :Str; Default :Bool) :Bool
2271: Func ReadCardinal( const AConvert :Bool) :Card
2272: Func ReadChar : Char
2273: Func ReadClient( var Buffer, Count : Int) : Int
2274: Func ReadDate( const Section, Name :Str; Default : TDateTime) : TDateTime
2275: Func ReadDateTime( const Section, Name :Str; Default : TDateTime) : TDateTime
2276: Func ReadFloat( const Section, Name :Str; Default : Double) : Double
2277: Func ReadFromStack(const ARaiseExceptfDiscon:Bool;ATimeout:Int;const ARaiseExceptTimeout:Bool):Int
2278: Func ReadInt( const AConvert :Bool) : Int
2279: Func ReadInt( const Section, Ident :Str; Default : Longint) : Longint
2280: Func ReadLn :Str
2281: Func ReadLn(ATerminator:Str;const ATimeout:Int;AMaxLineLength:Int):str
2282: Func Readln(question:Str):Str;
2283: Func readm:Str; //read last line in memo2 - console!
2284: Func ReadLnWait( AFailCount : Int) :Str
2285: Func ReadReg(Base: HKEY; KeyName, ValueName:Str):Str;
2286: Func ReadRegistry(Base: HKEY; KeyName, ValueName:Str):Str;
2287: Func ReadSmallInt( const AConvert :Bool) : SmallInt
2288: Func ReadString( const ABytes : Int) :Str
2289: Func ReadString( const Section, Ident, Default :Str) :Str
2290: Func ReadString( Count : Int) :Str
2291: Func ReadTime( const Section, Name :Str; Default : TDateTime) : TDateTime
2292: Func ReadTimeStampCounter : Int64
2293: Func RebootOS :Bool
2294: Func Receive( ATimeOut : Int) : TReplyStatus
2295: Func ReceiveBuf( var Buf, Count : Int) : Int
2296: Func ReceiveLength : Int
2297: Func ReceiveText :Str
2298: Func ReceiveSerialData(var Data: TByteArray; DataSize:Card):Card
2299: Func ReceiveSerialText:Str
2300: Func RecodeDate(const AValue:TDateTime;const AYear,AMonth,ADay:Word):TDateTime
2301: Func RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,Amin,ASec,AMilliSec:Word):TDateTime
2302: Func RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2303: Func RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2304: Func RecodeMilliSecond(const AValue:TDateTime;const AMilliSecond:Word):TDateTime
2305: Func RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2306: Func RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2307: Func RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2308: Func RecodeTime(const AValue:TDateTime;const AHour,AMinute,ASec,AMilliSec:Word):TDateTime
2309: Func RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2310: Func Reconcile( const Results : OleVariant) :Bool
2311: Func Rect( Left, Top, Right, Bottom : Int) : TRect
2312: Func Rect(ALeft: Int; ATop: Int; ARight: Int; ABottom: Int): TRect)
2313: Func Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2314: Func RectAssign( const Left, Top, Right, Bottom : Int) : TRect
2315: Func RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2316: Func RectBounds( const Left, Top, Width, Height : Int) : TRect
2317: Func RectCenter( const R : TRect) : TPoint
2318: Func RectEqual( const R1, R2 : TRect) :Bool
2319: Func RectHeight( const R : TRect) : Int
2320: Func RectIncludesPoint( const R : TRect; const Pt : TPoint) :Bool
2321: Func RectIncludesRect( const R1, R2 : TRect) :Bool
2322: Func RectIntersection( const R1, R2 : TRect) : TRect
2323: Func RectIntersectRect( const R1, R2 : TRect) :Bool
2324: Func RectIsEmpty( const R : TRect) :Bool
2325: Func RectIsNull( const R : TRect) :Bool
2326: Func RectIsSquare( const R : TRect) :Bool
2327: Func RectIsValid( const R : TRect) :Bool
2328: Func RectsAreValid( R : array of TRect) :Bool
2329: Func RectUnion( const R1, R2 : TRect) : TRect
2330: Func RectWidth( const R : TRect) : Int
2331: Func RedComponent( const Color32 : TColor32) : Int
2332: Func Refresh :Bool
2333: Func RefStringListCopy(aRefArray:TStringlist):TStringList;
2334: Func RegisterConversionFamily( const ADescription :Str) : TConvFamily
2335: Func RegisterConversionType(AConvTypeInfo: TConvTypeInfo; out AType : TConvType) :Bool;
2336: Func RegisterConversionType(const AFam:TConvFam;const ADescr:str;const AFact:Double):TConvType
2337: Func RegistryRead(keyHandle: Longint; keyPath, myField:Str):Str;
2338: Func ReleaseDC(hdwnd: HWND; hdc: HDC) : Int;
2339: Func ReleaseHandle : HBITMAP
2340: Func ReleaseHandle : HENHMETAFILE
2341: Func ReleaseHandle : HICON
2342: Func ReleasePalette : HPALETTE
2343: Func RemainderFloat( const X, Y : Float) : Float
2344: Func Remove(AClass : TClass) : Int
2345: Func Remove(AComponent : TComponent) : Int
2346: Func Remove(AItem : Int) : Int

```

```

2347: Func Remove(AItem : Pointer) : Pointer
2348: Func Remove(AItem : TObject) : TObject
2349: Func Remove(AObject : TObject) : Int
2350: Func RemoveBackslash( const PathName :Str) :Str
2351: Func RemovedF( aString:Str) :Str //removes thousand separator
2352: Func RemovedDir( Dir :Str) :Bool
2353: Func RemovedDir(const Dir:Str):Bool
2354: Func RemoveDirectory(PathName: PChar): WordBool; stdcall;
2355: Func RemoveFileExt( const FileName :Str) :Str
2356: Func RemoveHeaderEntry( AHeader, AEntry :Str) :Str
2357: Func RenameFile( OldName, NewName :Str) :Bool
2358: Func RenameFile(const OldName:Str; const NewName:Str):Bool
2359: Func ReplaceStr( const AText, AFromText, AToText :Str) :Str
2360: Func ReplaceText( const AText, AFromText, AToText :Str):Str
2361: Func Replicate(c : char;I : longInt) :Str;
2362: Func Request : TWebRequest
2363: Func ResemblesText( const AText, AOther :Str):Bool
2364: Func Reset :Bool
2365: Func Reset2(mypath:Str): TStringlist //string;
2366: Func ResInstLoad(Instance:THandle;ResType:TResType;const Name:str;MaskColor:TColor):Bool
2367: Func ResourceLoad(ResType:TResType;const Name string;MaskColor:TColor):oolean
2368: Func Response : TWebResponse
2369: Func ResumeSupported :Bool
2370: Func RETHINKHOTKEYS :Bool
2371: Func RETHINKLINES :Bool
2372: Func Retrieve( const MsgNum : Int; AMsg : TIdMessage) :Bool
2373: Func RetrieveCurrentDir :Str
2374: Func RetrieveDeltas( const cdsArray : array of TClientDataset): Variant
2375: Func RetrieveHeader( const MsgNum : Int; AMsg : TIdMessage):Bool
2376: Func RetrieveMailBoxSize : Int
2377: Func RetrieveMsgSize( const MsgNum : Int) : Int
2378: Func RetrieveProviders( const cdsArray : array of TClientDataset): Variant
2379: Func RetrieveRaw( const MsgNum : Int; const Dest: TString):bool
2380: Func ReturnMIMEType( var MediaType, EncType :Str):Bool
2381: Func ReverseBits( Value:Byte) : Byte;
2382: Func ReverseBits1(Value:Shortint) : Shortint;
2383: Func ReverseBits2(Value:Smallint) : Smallint;
2384: Func ReverseBits3(Value:Word) : Word;
2385: Func ReverseBits4(Value:Card) :Card;
2386: Func ReverseBits4(Value:Int) : Int;
2387: Func ReverseBits5(Value:Int64) : Int64;
2388: Func ReverseBytes(Value:Word) : Word;
2389: Func ReverseBytes1(Value:Smallint):Smallint;
2390: Func ReverseBytes2(Value:Int): Int;
2391: Func ReverseBytes3(Value:Card):Card;
2392: Func ReverseBytes4(Value:Int64): Int64;
2393: Func ReverseString(const AText :Str) :Str
2394: Func ReverseDNSLookup(const IPAddr:Str;DNSServer:Str;Timeout,Retries:Int;var HName:Str):Bool;
2395: Func Revert : HRESULT
2396: Func RGB(R,G,B: Byte): TColor;
2397: Func RGB2BGR( const Color : TColor) : TColor
2398: Func RGB2TColor( R, G, B : Byte) : TColor
2399: Func RGBToWebColorName( RGB : Int) :Str
2400: Func RGBToWebColorStr( RGB : Int) :Str
2401: Func RgbToHtml( Value : TColor) :Str
2402: Func HtmlToRgb(const Value:Str): TColor;
2403: Func RightStr(const AStr :Str; Len : Int) :Str
2404: Func RightStr(const AText : Ansistr; const ACount : Int): Ansistr;
2405: Func RightStr(const AText : WideString; const ACount : Int): WideString;
2406: Func ROL( AVal: LongWord; AShift : Byte): LongWord
2407: Func ROR( AVal: LongWord; AShift : Byte): LongWord
2408: Func RotatePoint(Point:TFloatPoint;const Center:TFloatPoint;const Angle:Float):TFloatPoint
2409: Func RotatePoint(Point:TFloatPoint;const Center:TFloatPoint;const Angle:Double):TFloatPoint;
2410: Func Round(e : Extended) : Longint;
2411: Func Round64(e: extended): Int64;
2412: Func RoundAt( const Value :Str; Position : SmallInt) :Str
2413: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2414: Func RoundTo(const AValue:Extended;const ADigit:TRoundToEXRangeExtended):Extended;;
2415: Func SimpleRoundTo(const AValue:Extended;const ADigit: TRoundToRange): Extended;;
2416: Func RoundFrequency( const Frequency : Int) : Int
2417: Func RoundInt( Value : Int; StepSize : Int) : Int
2418: Func RoundPoint( const X, Y : Double) : TPoint
2419: Func RoundRect( const ALeft, ATop, ARight,ABottom :Double):TRect
2420: Func RowCount : Int
2421: Func RowRequest(const Row:OleVariant;RequestType:Int;var OwnerData:OleVariant):OleVariant
2422: Func RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2423: Func RPos( const ASub, AIn :Str; AStart : Int) : Int
2424: Func RRot( const Value : Byte; const Count : TBitRange) : Byte;
2425: Func RRot1( const Value : Word; const Count : TBitRange) : Word;
2426: Func RRot2( const Value : Int; const Count : TBitRange) : Int;
2427: Func RunDLL32(const ModuleNa,FuncName,CmdLine:str;WaitForCompletion:Bool;CmdShow:Int):Bool
2428: Func RunningProcessesList( const List : TString; FullPath :Bool) :Bool
2429: Func RunByteCode(Bytecode: Ansistr; out RuntimeErrors: Ansistr):Bool;;
2430: Func RunCompiledScript2(Bytecode: Ansistr;out RuntimeErrors:Ansistr): Bool;
2431: Func S_AddBackSlash( const ADirName :Str) :Str
2432: Func S_AllTrim( const cStr :Str) :Str
2433: Func S_AtRepl( const cAT, cStr, cRepl :Str) :Str
2434: Func S_Cut( const cStr :Str; const iLen : Int) :Str
2435: Func S_DecryptCRC32( const crc :Str; StartKey, MultKey, AddKey : Int): Int

```

```

2436: Func S_DirExists( const ADir :Str) :Bool
2437: Func S_Empty( const cStr :Str) :Bool
2438: Func S_EncryptCRC32(const crc:LongWORD;StartKey,MultKey,AddKey:Int):str
2439: Func S_LargeFontsActive :Bool
2440: Func S_LimitDigits( AValue : Extended; ANumDigits : Int) : Extended
2441: Func S_LTrim( const cStr :Str) :Str
2442: Func S_ReadNextTextLineFromStream( stream : TStream) :Str
2443: Func S_RepeatChar( const iLen : Int; const AChar : Char) :Str
2444: Func S_ReplFirst( const cAT, cStr, cRepl :Str) :Str
2445: Func S_RoundDecimal( AValue : Extended; APlaces : Int) : Extended
2446: Func S_RTrim( const cStr :Str) :Str
2447: Func S_RTrimCopy( const cStr :Str; iPos, iLen : Int) :Str
2448: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2449: Func S_ShellExecute(aFilename:str;aParameters:str;aCommand:TS_ShellExecuteCmd):str
2450: Func S_Space( const iLen : Int) :Str
2451: Func S_StrBlanks( const cStr :Str; const iLen : Int) :Str
2452: Func S_StrBlanksCuttooLong( const cStr :Str; const iLen : Int) :Str
2453: Func S_StrCRC32( const Text :Str) : LongWORD
2454: Func S_StrDecrypt96(const InString:Str;StartKey,MultKey,AddKey:Int):str
2455: Func S_StrEncrypt96(const InString:Str;StartKey,MultKey,AddKey:int):Str
2456: Func S_StringtoUTF_8( const AString :Str) :Str
2457: Func S_StrLBlanks( const cStr :Str; const iLen : Int) :Str
2458: Func S_StrToReal(const cStr:Str; var R: Double):Bool
2459: Func S-TokenEnd( cBuffer : PChar; lEmptyToken :Bool) :Bool
2460: Func S-TokenNext( cBuffer : PChar; lEmptyToken :Bool):Str
2461: Func S_UTF_8ToString( const AString :Str) :Str
2462: Func S_WBox( const AText :Str) : Int
2463: Func SameDate( const A, B : TDateTime) :Bool
2464: Func SameDate(const A, B: TDateTime):Bool;
2465: Func SameDateTime( const A, B : TDateTime) :Bool
2466: Func SameDateTime(const A, B: TDateTime):Bool;
2467: Func SameFileName( S1, S2 :Str) :Bool
2468: Func SameText( S1, S2 :Str) :Bool
2469: Func SameText(const S1:Str; const S2:Str):Bool)
2470: Func SameTime( const A, B : TDateTime) :Bool
2471: Func SameTime(const A, B: TDateTime):Bool;
2472: Func SameValue(const A, B: Extended; Epsilon: Extended):Bool //overload;
2473: Func SameValue1(const A, B: Double; Epsilon: Double):Bool //overload;
2474: Func SameValue2(const A, B: Single; Epsilon: Single):Bool //overload;
2475: Func SampleVariance( const X : TDynFloatArray) : Float
2476: Func Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2477: Func Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2478: Func Sar2( const Value : Int; const Count : TBitRange) : Int;
2479: Func SaveToFile( const AFileName : TFileName) :Bool
2480: Func SaveAsExcelFile(AGrid:TStringGrid;ASheetName,AFileName:str;open:bool):Bool;
2481: Func SaveAsExcel(aGrid:TStringGrid;aSheetName,aFileName:str;openexcel:boolean):Bool;
2482: Func ScanF(const aformat:Str; const args: array of const):Str;
2483: Func SCREENTOCLIENT(POINT:TPOINT):TPOINT
2484: Func SearchBuf(Buf:PChar;BufLen:Int;SelStart,
SelLength:Int;SearchString:str;Options:TStringSearchOptions):PChar
2485: Func SearchBuf2(Buf:str;SelStart,SelLength:Int;SearchString:str;Options:TStringSearchOptions):Int;
2486: Func SearchRecattr: Int;
2487: Func SearchRecExcludeAttr: Int;
2488: Func SearchRecFileSize64( const SearchRec: TSearchRec):Int64
2489: Func SearchRecname:Str;
2490: Func SearchRecsize: Int;
2491: Func SearchRecTime: Int;
2492: Func Sec( const X : Extended) : Extended
2493: Func Secant( const X : Extended) : Extended
2494: Func SecH( const X : Extended) : Extended
2495: Func SecondOf( const AValue : TDateTime) : Word
2496: Func SecondOfTheDay( const AValue : TDateTime) : LongWord
2497: Func SecondOfTheHour( const AValue : TDateTime) : Word
2498: Func SecondOfTheMinute( const AValue : TDateTime) : Word
2499: Func SecondOfTheMonth( const AValue : TDateTime) : LongWord
2500: Func SecondOfTheWeek( const AValue : TDateTime) : LongWord
2501: Func SecondOfTheYear( const AValue : TDateTime) : LongWord
2502: Func SecondsBetween( const ANow, AThen : TDateTime) : Int64
2503: Func SecondSpan( const ANow, AThen : TDateTime) : Double
2504: Func SectionExists( const Section :Str) :Bool
2505: Func Seek( const KeyValues : Variant; SeekOption : TSeekOption) :Bool
2506: Func Seek(dlibMove: Longint; dwOrigin: Longint; out libNewPosition : Largeint):HResult
2507: Func Seek(Offset:LongInt;Origin:Word):LongInt
2508: Func SelectDirectory(var Directory:str; Options:TSelectDirOpts;HelpCtx:Longint):Bool;
2509: Func SelectDirectory1(const Caption:str;const Root:WideString;var Directory:str;
Options:TSelectDirExtOpts;Parent:TWinControl):Bool;
2510: Func SelectImage(var AFileName:str;const Extensions,Filter:str):Boolean
2511: Func SendAppMessage(Msg:Card; WParam, LParam: Longint) : Longint
2512: Func SendBuf( var Buf, Count : Int) : Int
2513: Func SendCmd( const AOut :Str; const AResponse : SmallInt) : SmallInt;
2514: Func SendCmd1( const AOut :Str; const AResponse : array of SmallInt) : SmallInt;
2515: Func SendKey( AppName :Str; Key : Char) :Bool
2516: Func SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint):Bool;
2517: Func SendStream( AStream : TStream) :Bool
2518: Func SendStreamThenDrop( AStream : TStream) :Bool
2519: Func SendText( const S :Str) : Int
2520: Func SendSerialData(Data: TByteArray; DataSize:Card):Card
2521: Func SendSerialText(Data:Str):Card
2522: Func Sent :Bool

```



```

2523: Func ServicesFilePath:Str
2524: Func SetAlpha( const Color32 : TColor32; NewAlpha : Int) : TColor32
2525: Func SetBit( const Value:Byte; const Bit : TBitRange) : Byte;
2526: Func SetBit1(const Value:Shortint; const Bit : TBitRange) : Shortint;
2527: Func SetBit2(const Value:Smallint; const Bit : TBitRange) : Smallint;
2528: Func SetBit3(const Value:Word; const Bit : TBitRange) : Word;
2529: Func SetBit4(const Value:Card; const Bit : TBitRange) : Cardinal;
2530: Func SetBit4(const Value:Int; const Bit : TBitRange) : Int;
2531: Func SetBit5(const Value:Int64; const Bit : TBitRange) : Int64;
2532: Func SetClipboard( NewClipboard : TClipboard) : TClipboard
2533: Func SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2534: Func SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2535: Func SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2536: Func SetColorRed( const Color : TColor; const Red : Byte) : TColor
2537: Func SetCurrentDir( Dir :Str) :Bool
2538: Func SetCurrentDir(const Dir:Str):Bool
2539: Func SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2540: Func SetDirCreation( const DirName :Str; const DateTime : TDateTime):Bool
2541: Func SetDirLastAccess(const DirName:str;const DateTime: TDateTime) : Bool
2542: Func SetDirLastWrite(const DirName:str; const DateTime : TDateTime): Bool
2543: Func SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2544: Func SetEndOfFile(Handle: Int): LongBool; stdcall;
2545: Func SetEnvironmentVar( const Name, Value :Str) :Bool
2546: Func SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2547: Func SetFileCreation(const FileName:str; const DateTime : TDateTime): Bool
2548: Func SetFileLastAccess( const FileName:str; const DateTime:TDateTime): Bool
2549: Func SetFileLastWrite(const FileName:str;const DateTime:TDateTime):BoolFunc SetFileTimeStamp(const
  FileName:str;TimeStamp:Int):Bool
2550: Func SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2551: Func SetLocalTime( Value : TDateTime) :Bool
2552: Func SetPrecisionTolerance( NewTolerance : Float) : Float
2553: Func SetPrinter( NewPrinter : TPrinter) : TPrinter
2554: Func SetPrivilege(privilegeName:Str; enable:Bool):Bool;
2555: Func SetRGBValue( const Red, Green, Blue : Byte) : TColor
2556: Func SetSequence( S, Localizar, Substituir : shortstring): shortstring
2557: Func SetSize( libNewSize : Longint) : HResult
2558: Func SetUserObjectFullAccess( hUserObject : THandle) :Bool
2559: Func Sgn( const X : Extended) : Int
2560: Func SHA1(const fileName:Str):Str;
2561: Func SHA256(astr:Str; amode: char):Str)
2562: Func SHA512(astr:Str; amode: char):Str)
2563: Func ShareMemoryManager :Bool
2564: Func ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:str;ShowCmd:Int):Int;stdcall;
2565: Func ShellExecute2(hwnd: HWND; const FileName:Str):Int; stdcall;
2566: Func ShellExecute3(aFilename:str;aParameters:str;aCommand:TS_ShellExecuteCmd):str;
2567: Func SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE): TSHORTCUT
2568: Func SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT):Str
2569: Func ShortDateFormat:Str;
2570: Func ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
  RTL:Bool;EllipsisWidth:Int):WideString
2571: Func ShortTimeFormat:Str;
2572: Func SHOWMODAL:Int
2573: Func
  ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:TColor;BW:Int;Title:str;
2574: Func ShowModalPanel(aPnl:TCustomPanel;Titl:str;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2575: Func ShowWindow(C1: HWND; C2: Int):Bool;
2576: Proc ShowMemory //in Dialog
2577: Func ShowMemory2:Str;
2578: Func ShutDownOS :Bool
2579: Func Signe( const X, Y : Extended) : Extended
2580: Func Sign( const X : Extended) : Int
2581: Func Sin(e : Extended) : Extended;
2582: //Assuming trigonometric arguments in degrees //| Use instead
2583: Printf('in deg: %.18f',[sin(degtorad(60.0))]); //0.866025403784438647
2584: Func sinc( const x : Double) : Double
2585: Func SinJ( X : Float) : Float
2586: Func Size( const AFileName :Str) : Int
2587: Func SizeOf: Longint;
2588: Func SizeofResource( ModuleHandle : HMODULE; ResHandle: TResourceHandle): Int
2589: Func SlashSep(const Path, S:Str):Str
2590: Func SLNDepreciation( const Cost, Salvage : Extended; Life : Int) : Extended
2591: Func SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2592: Func SmallPoint(X, Y: Int): TSmallPoint)
2593: Func Soundex( const AText :Str; ALength : TSoundexLength) :Str
2594: Func SoundexCompare( const AText, AOther :Str; ALength : TSoundexLength) : Int
2595: Func SoundexInt( const AText :Str; ALength : TSoundexIntLength) : Int
2596: Func SoundexProc( const AText, AOther :Str) :Bool
2597: Func SoundexSimilar(const AText,AOther:str;ALength:TSoundexLength):Boolean
2598: Func SoundexWord( const AText :Str) : Word
2599: Func SourcePos : Longint
2600: Func SourcePos:LongInt
2601: Func Split0( Str :Str; const substr :Str) : TStringList
2602: Proc SplitNameValue( const Line :Str; var Name, Value :Str)
2603: Func SQLRequiresParams( const SQL : WideString) :Bool
2604: Func Sqr(e : Extended) : Extended;
2605: Func Sqrt(e : Extended) : Extended;
2606: Func StartIP:Str
2607: Func StartPan( WndHandle : THandle; AControl : TControl) :Bool
2608: Func StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;

```

```

2609: Func StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2610: Func StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2611: Func StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek:Word) :TDateTime
2612: Func StartOfAYear( const AYear : Word) : TDateTime
2613: Func StartOfTheDay( const AValue : TDateTime) : TDateTime
2614: Func StartOfTheMonth( const AValue : TDateTime) : TDateTime
2615: Func StartOfTheWeek( const AValue : TDateTime) : TDateTime
2616: Func StartOfTheYear( const AValue : TDateTime) : TDateTime
2617: Func StartsStr( const ASubText, AText :Str) :Bool
2618: Func StartsText( const ASubText, AText :Str) :Bool
2619: Func StartsWith( const ANSIStr, APattern :Str) :Bool
2620: Func StartsWith( const str :Str; const sub :Str) :Bool
2621: Func StartsWithACE( const ABytes : TIdBytes) :Bool
2622: Func StatusString( StatusCode : Int) :Str
2623: Func StdDev( const Data : array of Double) : Extended
2624: Func Stop : Float
2625: Func StopCount( var Counter : TJclCounter) : Float
2626: Func StoreColumns :Bool
2627: Func StrAfter( const sString :Str; const sDelimiters :Str):Str;
2628: Func StrAfter1(const sString:str; const sDelimiters:str;out cDelimiter:char):str;
2629: Func StrAlloc( Size :Card) : PChar
2630: Func StrAlloc(Size:Card): PChar)
2631: Func StrBefore( const sString :Str; const sDelimiters :Str) :Str;
2632: Func StrBefore1(const sString:str;const sDelimiters:str;out cDelimiter:char):str;
2633: Func StrBufSize( Str : PChar) :Card
2634: Func StrBufSize(const Str: PChar):Card)
2635: Func StrByteType( Str : PChar; Index :Card) : TMbcsByteType
2636: Func StrByteType(Str: PChar; Index:Card): TMbcsByteType)
2637: Func StrCat( Dest : PChar; Source : PChar) : PChar
2638: Func StrCat(Dest: PChar; const Source: PChar): PChar)
2639: Func StrCharLength( Str : PChar) : Int
2640: Func StrComp( Str1, Str2 : PChar) : Int
2641: Func StrComp(const Str1: PChar; const Str2: PChar): Int)
2642: Func StrCopy( Dest : PChar; Source : PChar) : PChar
2643: Func StrCopy(Dest: PChar; const Source: PChar): PChar)
2644: Func Stream_to_Ansistr( Source : TStream) : Ansistr
2645: Func Stream_to_Base64( Source : TStream) : Ansistr
2646: Func Stream_to_decimalbytes( Source : TStream) :Str
2647: Func Stream2WideString( oStream : TStream) : WideString
2648: Func StreamtoAnsistr( Source : TStream) : Ansistr
2649: Func StreamToByte( Source : TStream) :Str
2650: Func StreamToDecimalbytes( Source : TStream) :Str
2651: Func StreamtoOrd( Source : TStream) :Str
2652: Func StreamToString( Source: TStream) :Str
2653: Func StreamToString2(Source: TStream) :Str
2654: Func StreamToString3(Source: TStream) :Str
2655: Func StreamToString4(Source: TStream) :Str
2656: Func StrECopy( Dest : PChar; Source : PChar) : PChar
2657: Func StrEmpty( const sString :Str) :Bool
2658: Func StrEnd( Str : PChar) : PChar
2659: Func StrEnd(const Str: PChar): PChar)
2660: Func StrFilter( const sString :Str; xValidChars : TCharSet) :Str
2661: Func StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2662: Func StrGet(var S :Str; I : Int) : Char;
2663: Func StrGet2(S :Str; I : Int) : Char;
2664: Func StrHasPrefix( const sString :Str; const sPrefix :Str) :Bool
2665: Func StrHasSuffix( const sString :Str; const sSuffix :Str) :Bool
2666: Func StrHtmlDecode( const AStr :Str) :Str
2667: Func StrHtmlEncode( const AStr :Str) :Str
2668: Func StrToBytes(const Value:Str): TBytes;
2669: Func StrIComp( Str1, Str2 : PChar) : Int
2670: Func StringOfChar(c : char;I : longInt) :Str;
2671: Func StringOfChar2( ch : WideChar; Count : Int) : WideString;
2672: Func StringPad(InputStr,FillChar:Str; StrLen:Int; StrJustify:Boolean):Str;
2673: Func StringRefCount(const s:Str): Int;
2674: Func StringReplace(S,OldPattern,NewPattern:str;Flags:TReplaceFlags):str
2675: Func JStringReplace(const S,OldPattern, NewPattern:Str; Flags : TReplaceFlags):str
2676: Func StringReplace(const SourceString,OldPattern,NewPattern:str;Flags:TReplaceFlags):str;
2677: Func StringRemove( const S, Pattern :Str; Flags : TReplaceFlags) :Str
2678: Func StringToBoolean( const Ps :Str) :Bool
2679: Func StringToColor(const S:Str): TColor)
2680: Func StringToCursor(const S:Str): TCursor;
2681: Func StringToGUID(const S:Str): TGUID)
2682: Func StringTokenizer( const str :Str; const delim :Str) : IStringTokenizer
2683: Func StringToStringArray( const str :Str; const delim:Str): TStringDynArray
2684: Func StringWidth( S :Str) : Int
2685: Func StrInternetToDateTime( Value :Str) : TDateTime
2686: Func StrIsDateTime( const Ps :Str) :Bool
2687: Func StrIsFloatMoney( const Ps :Str) :Bool
2688: Func StrIsInt( const S :Str) :Bool
2689: Func StrLCat( Dest : PChar; Source : PChar; MaxLen :Card) : PChar
2690: Func StrLComp( Str1, Str2 : PChar; MaxLen :Card) : Int
2691: Func StrLCopy( Dest : PChar; Source : PChar; MaxLen :Card) : PChar
2692: Func StrLen( Str : PChar) :Card
2693: Func StrLen(const Str: PChar):Card)
2694: Func StrLessPrefix( const sString :Str; const sPrefix:Str):Str
2695: Func StrLessSuffix( const sString :Str; const sSuffix:Str):Str
2696: Func StrLIComp( Str1, Str2 : PChar; MaxLen :Card) : Int
2697: Func StrLower( Str : PChar) : PChar

```

```

2698: Func StrMove( Dest : PChar; Source : PChar; Count :Card) : PChar
2699: Func StrMove(Dest: PChar; const Source: PChar; Count:Card): PChar)
2700: Func StrNew( Str : PChar) : PChar
2701: Func StrNew(const Str: PChar): PChar)
2702: Func StrNextChar( Str : PChar) : PChar
2703: Func StrPad( const sString:str;const sPad:str;const iLength:Int):Str
2704: Func StrParse( var sString :Str; const sDelimiters :Str) :Str;
2705: Func StrParsel(var sString:str;const sDelimiters:str;out cDelimiter:char) string;
2706: Func StrPas( Str : PChar) :Str
2707: Func StrPas(const Str: PChar):Str)
2708: Func StrPCopy( Dest : PChar; Source :Str) : PChar
2709: Func StrPCopy(Dest: PChar; const Source:Str): PChar)
2710: Func StrPLCopy( Dest : PChar; Source :Str; MaxLen :Card) : PChar
2711: Func StrPos( Str1, Str2 : PChar) : PChar
2712: Func StrScan(const Str: PChar; Chr: Char): PChar)
2713: Func StrRScan(const Str: PChar; Chr: Char): PChar)
2714: Func StrToBcd( const AValue :Str) : TBcd
2715: Func StrToBool( S :Str) :Bool
2716: Func StrToBoolDef( S :Str; Default :Bool) :Bool
2717: Func StrToCard( const AStr :Str) :Card
2718: Func StrToConv( AText :Str; out AType : TConvType) : Double
2719: Func StrToCurr( S :Str) : Currency;
2720: Func StrToCurr(const S:Str): Currency)
2721: Func StrToCurrDef( S :Str; Default : Currency) : Currency;
2722: Func StrToDate( S :Str) : TDateTime;
2723: Func StrToDate(const s:Str): TDateTime;
2724: Func StrToDateDef( S :Str; Default : TDateTime) : TDateTime;
2725: Func StrToDateTime( S :Str) : TDateTime;
2726: Func StrToDateTime(const S:Str): TDateTime)
2727: Func StrToDateTimeDef( S :Str; Default : TDateTime) : TDateTime;
2728: Func StrToDay( const ADay :Str) : Byte
2729: Func StrToFloat( S :Str) : Extended;
2730: Func StrToFloat(s:Str): Extended;
2731: Func StrToFloatDef( S :Str; Default : Extended) : Extended;
2732: Func StrToFloatDef(const S:Str; const Default: Extended): Extended)
2733: Func StrToFloat( S :Str) : Extended;
2734: Func StrToFloat2( S :Str; FormatSettings : TFormatSettings) : Extended;
2735: Func StrToFloatDef( S :Str; Default : Extended) : Extended;
2736: Func StrToFloatDef2(S:str;Default:Extended;FormatSettings:TFormatSettings):Extended;
2737: Func StrToCurr( S :Str) : Currency;
2738: Func StrToCurr2( S :Str; FormatSettings : TFormatSettings) : Currency;
2739: Func StrToCurrDef( S :Str; Default : Currency) : Currency;
2740: Func StrToCurrDef2(S:str; Default:Currency; FormatSettings:TFormatSettings):Currency;
2741: Func StrToTime2( S :Str; FormatSettings : TFormatSettings) : TDateTime;
2742: Func StrToTimeDef( S :Str; Default : TDateTime) : TDateTime;
2743: Func StrToTimeDef2(S:str; Default:TDateTime;FormatSettings:TFormatSettings):TDateTime;
2744: Func TryStrToTime( S :Str; Value : TDateTime) :Bool;
2745: Func StrToDateTime( S :Str) : TDateTime;
2746: Func StrToDateTime2( S :Str; FormatSettings : TFormatSettings) : TDateTime;
2747: Func StrToDateTimeDef( S :Str; Default : TDateTime) : TDateTime;
2748: Func StrToFloatRegionalIndependent(aValue:str;aDecimalSymbol:Char;aDigitGroupSymbol:Char):Extended
2749: Func StrToInt( S :Str) : Int
2750: Func StrToInt(s:Str): Longint;
2751: Func StrToInt64( S :Str) : Int64
2752: Func StrToInt64(s:Str): int64;
2753: Func StrToInt64Def( S :Str; Default : Int64) : Int64
2754: Func StrToInt64Def(const S:Str; const Default: Int64):Int64)
2755: Func StrToIntDef( S :Str; Default : Int) : Int
2756: Func StrToIntDef(const S:Str; Default: Int): Int)
2757: Func StrToIntDef(s:Str; def: Longint): Longint;
2758: Func StrToMonth( const AMonth :Str) : Byte
2759: Func StrToTime( S :Str) : TDateTime;
2760: Func StrToTime(const S:Str): TDateTime)
2761: Func StrToTimeDef( S :Str; Default : TDateTime) : TDateTime;
2762: Func StrToWord( const Value :Str) : Word
2763: Func StrToXmlDate( const DateStr :Str; const Format :Str) :Str
2764: Func StrToXmlDateTime( const DateStr :Str; const Format :Str) :Str
2765: Func StrToXmlTime( const TimeStr :Str; const Format :Str) :Str
2766: Func StrUpper( Str : PChar) : PChar
2767: Func StuffString(const AText:str;AStart, ALength:Card;const ASubText:str):str
2768: Func Sum( const Data : array of Double) : Extended
2769: Func SumFloatArray( const B : TDynFloatArray) : Float
2770: Func SumInt( const Data : array of Int) : Int
2771: Func SumOfSquares( const Data : array of Double) : Extended
2772: Func SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2773: Func SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2774: Func SumSquareFloatArray( const B : TDynFloatArray) : Float
2775: Func Supports( CursorOptions : TCursorOptions) :Bool
2776: Func SupportsClipboardFormat( AFormat : Word) :Bool
2777: Func SwapWord(w : word): word)
2778: Func SwapInt(i : Int): Int)
2779: Func SwapLong(L : longint): longint)
2780: Func Swap(i : Int): Int)
2781: Func SYDDepreciation(const Cost,Salvage:Extended;Life,Period:Int): Extended
2782: Func SyncTime :Bool
2783: Func SysErrorMessage( ErrorCode : Int) :Str
2784: Func SysErrorMessage(ErrorCode: Int):Str)
2785: Func SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2786: Func SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;

```

```

2787: Func SysStringLen(const S: WideString): Int; stdcall;
2788: Func TabRect( Index : Int) : TRect
2789: Func Tan( const X : Extended) : Extended
2790: Func TaskMessageDlg(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint):Int;
2791: Func TaskMessageDlg1(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;
  HelpCtx:Longint;DefaultButton:TMsgDlgBtn):Int;
2792: Func TaskMessageDlgPos(const Title,Msg:str;DlgType:TMsgDlgType;Bts:TMsgDlgButtons;HelpCtx:Longint;X,
  Y:Int):Int;
2793: Func TaskMessageDlgPos1(const Title,Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
  Y:Int;DefaultButton:TMsgDlgBtn):Int;
2794: Func TaskMessageDlgPosHelp(const Title,
  Msg:str;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,Y:Int;const HelpFileName:str): Int;
2795: Func TaskMessageDlgPosHelp1(const Title, Msg:str;DlgType: TMsgDlgType; Buttons :
  TMsgDlgButtons;HelpCtx:Longint;X,Y:Int;const HelpFileName:str;DefaultButton:TMsgDlgBtn):Int;
2796: Func TenToY( const Y : Float) : Float
2797: Func TerminateApp( ProcessID : DWORD; Timeout : Int) : TJclTerminateAppResult
2798: Func TerminateTask( Wnd : HWND; Timeout : Int) : TJclTerminateAppResult
2799: Func TestBit( const Value:Byte; const Bit : TBitRange) : Bool;
2800: Func TestBit2(const Value:Shortint; const Bit : TBitRange): Bool;
2801: Func TestBit3(const Value:Smallint; const Bit : TBitRange): Bool;
2802: Func TestBit4(const Value:Word; const Bit : TBitRange):Bool;
2803: Func TestBit5(const Value:Card; const Bit : TBitRange): Bool;
2804: Func TestBit6(const Value:Int; const Bit : TBitRange): Bool;
2805: Func TestBit7(const Value:Int64; const Bit : TBitRange): Bool;
2806: Func TestBits(const Value, Mask:Byte) :Bool;
2807: Func TestBits1(const Value,Mask:Shortint) :Bool;
2808: Func TestBits2(const Value,Mask:Smallint) :Bool;
2809: Func TestBits3(const Value,Mask:Word) :Bool;
2810: Func TestBits4(const Value,Mask:Card) :Bool;
2811: Func TestBits5(const Value,Mask:Int):Bool;
2812: Func TestBits6(const Value,Mask:Int64) :Bool;
2813: Func TestFDIVInstruction :Bool
2814: Func TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2815: Func TextExtent( const Text :Str) : TSize
2816: Func TextHeight(Text:Str): Int;
2817: Func TextIsSame( const A1 :Str; const A2 :Str) :Bool
2818: Func TextStartsWith( const S, SubS :Str) :Bool
2819: Func TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue):Boolean)
2820: Func ConvInt(i : Int):str;
2821: Func IntToText(i : Int):str;
2822: Func TEXTTOSHORTCUT( TEXT :Str) : TSHORTCUT
2823: Func TextWidth(Text:Str): Int;
2824: Func ThreadCount : Int
2825: Func ThousandSeparator: char;
2826: Func Ticks :Card
2827: Func Time : TDateTime
2828: Func Time: TDateTime;
2829: Func TimeGetTime: int64;
2830: Func TimeOf( const AValue:TDateTime):TDateTime
2831: Func TimeSeparator: char;
2832: Func TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2833: Func TimeStampToMsecs( TimeStamp : TTimeStamp) : Comp
2834: Func TimeStampToMsecs(const TimeStamp: TTimeStamp): Comp)
2835: Func TimeToStr( DateTime : TDateTime):Str;
2836: Func TimeToStr(const DateTime: TDateTime):Str;
2837: Func TimeZoneBias : TDateTime
2838: Func ToCommon( const AValue : Double) : Double
2839: Func ToCommon(const AValue: Double): Double;
2840: Func Today : TDateTime
2841: Func ToggleBit( const Value:Byte; const Bit : TBitRange) : Byte;
2842: Func ToggleBit1(const Value:Shortint; const Bit : TBitRange): Shortint;
2843: Func ToggleBit2(const Value:Smallint; const Bit : TBitRange): Smallint;
2844: Func ToggleBit3(const Value:Word; const Bit : TBitRange) : Word;
2845: Func ToggleBit4(const Value:Card; const Bit : TBitRange):Card;
2846: Func ToggleBit5(const Value:Int; const Bit : TBitRange) : Int;
2847: Func ToggleBit6(const Value:Int64; const Bit : TBitRange) : Int64;
2848: Func TokenComponentIdent:str
2849: Func TokenFloat : Extended
2850: Func TokenFloat:Extended
2851: Func TokenInt : Longint
2852: Func TokenInt:LongInt
2853: Func TokenString :Str
2854: Func TokenString:str
2855: Func TokenSymbolIs( const S :Str) :Bool
2856: Func TokenSymbolIs(S:str):Boolean
2857: Func Tomorrow : TDateTime
2858: Func ToRightOf( const pc : TControl; piSpace : Int) : Int
2859: Func ToString :Str
2860: Func TotalVariance( const Data : array of Double) : Extended
2861: Func Trace2( AURL :Str) :Str;
2862: Func TrackMenu( Button : TToolButton) :Bool
2863: Func TRANSLATE( SRC, DEST : PCHAR; TOOEM :Bool) : Int
2864: Func TranslateURI( const URI :Str) :Str
2865: Func TranslationMatchesLanguages( Exact :Bool) :Bool
2866: Func TransparentStretchBlt(DstDC:HDC;DstX,DstY,DstW,DstH:Int;SrcDC:HDC;SrcX,SrcY,SrcW,
  SrcH:Int;MaskDC:HDC;MaskX, MaskY:Int):Bool
2867: Func Trim( S :Str) :Str;
2868: Func Blank( S :Str) :Str; //alias to Trim
2869: Func Trim( S : WideString) : WideString;

```

```

2870: Func Trim(s : AnyString) : AnyString;
2871: Func TrimAllOf( ATrim, AText : Str):Str;
2872: Func TrimLeft( S :Str) :Str;
2873: Func TrimLeft( S : WideString) : WideString;
2874: Func TrimLeft(const S:Str):Str;
2875: Func TrimRight( S:Str) :Str;
2876: Func TrimRight( S: WideString) : WideString;
2877: Func TrimRight(const S:Str):Str;
2878: Func TrueBoolStrs: array of string;
2879: Func Trunc(e : Extended) : Longint;
2880: Func Trunc64(e: extended): Int64;
2881: Func TruncPower( const Base, Exponent : Float) : Float;
2882: Func TryConvTypeToFamily(const AFrom, ATo: TConvType; out AFamily: TConvFamily): Bool;
2883: Func TryConvTypeToFamily1(const AType: TConvType; out AFamily: TConvFamily): Bool;
2884: Func TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Bool;
2885: Func TryEncodeDateDay(const AYear, ADayOfYear: Word; out AValue: TDateTime): Bool;
2886: Func TryEncodeDateMonthWeek(const AY, AMonth, AWeekOfMonth, ADayOfWeek: Word; var AValue: TDateTime): Bool;
2887: Func TryEncodeDateMonthWeek(const AYear, AMonth, ADay, AHour, AMin, ASec, AMilliSecond: Word; out AValue: TDateTime): Bool;
2888: Func TryEncodeDateWeek(const AY, AWeekOfYear: Word; out AValue: TDateTime; const ADayOfWeek: Word): Bool;
2889: Func TryEncodeDayOfWeekInMonth(const AYear, AMonth, ANthDayOfWeek, ADayOfWeek: Word; out AVal: TDateTime): Bool;
2890: Func TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Bool;
2891: Func TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Bool;
2892: Func TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Bool;
2893: Func TryLock : Bool;
2894: Func TryModifiedJulianDateToDateTime(const AValue: Double; out ADateTime: TDateTime): Boolean;
2895: Func TryRecodeDate( const AValue: TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
AMilliSecond: Word; out AResult: TDateTime): Boolean;
2896: Func TryStrToBcd( const AValue : Str; var Bcd : TBcd) : Bool;
2897: Func TryStrToConv( AText : Str; out AValue : Double; out AType : TConvType) : Bool;
2898: Func TryStrToDate( S : Str; Value : TDateTime) : Bool;
2899: Func TryStrToDateTime( S : Str; Value : TDateTime) : Bool;
2900: Func TryStrToTime( S : Str; Value : TDateTime) : Bool;
2901: Func TryStrToInt(const S: Ansistr; var I: Int): Bool;
2902: Func TryStrToInt64(const S: Ansistr; var I: Int64): Bool;
2903: Func TryStrToBool(const S: Str; out Value: Bool): Bool;
2904: Func TwoByteToWord( AByte1, AByte2 : Byte) : Word;
2905: Func TwoCharToWord( AChar1, AChar2 : Char) : Word;
2906: Func TwoToY( const Y : Float) : Float;
2907: Func UCS4StringToWideString( const S : UCS4String) : WideString;
2908: Func UIDL( const ADest : TStrings; const AMsgNum : Int) : Bool;
2909: Func Unassigned: Variant;
2910: Func UndoLastChange( FollowChange : Bool) : Bool;
2911: Func UniCodeToStr(Value: Str): Str;
2912: Func UnionRect( out Rect : TRect; const R1, R2 : TRect) : Bool;
2913: Func UnionRect(out Rect: TRect; const R1, R2: TRect): Bool;
2914: Func UnixDateTimeToDelphiDateTime( UnixDateTime : Card) : TDateTime;
2915: Func UnixPathToDosPath( const Path : Str) : Str;
2916: Func UnixToDateTime( const AValue : Int64) : TDateTime;
2917: Func UnixToDateTime(U: Int64): TDateTime;
2918: Func UnlockRegion(libOffset Longint; cb: Largeint; dwLockType: Longint): HRESULT;
2919: Func UnlockResource( ResData : HGLOBAL) : LongBool;
2920: Func UnlockVolume( var Handle : THandle) : Bool;
2921: Func UnMaskString( Mask, Value : Str) : Str;
2922: Func UpCase(ch : Char) : Char;
2923: Func UpCaseFirst( const AStr : Str) : Str;
2924: Func UpCaseFirstWord( const AStr : Str) : Str;
2925: Func UpdateAction( Action : TBasicAction) : Bool;
2926: Func UpdateKind : TUpdateKind;
2927: Func UPDATESTATUS : TUPDATESTATUS;
2928: Func UpperCase( S : Str) : Str;
2929: Func Uppercase(s : AnyString) : AnyString;
2930: Func URLDecode( ASrc : Str) : Str;
2931: Func URLEncode( const ASrc : Str) : Str;
2932: Func UseRightToLeftAlignment : Bool;
2933: Func UseRightToLeftAlignmentForField(const AField: TField; Alignment: TAlignment): Bool;
2934: Func UseRightToLeftReading : Bool;
2935: Func UTF8CharLength( Lead : Char) : Int;
2936: Func UTF8CharSize( Lead : Char) : Int;
2937: Func UTF8Decode( const S : UTF8String) : WideString;
2938: Func UTF8Encode( const WS : WideString) : UTF8String;
2939: Func UTF8LowerCase( const S : UTF8String) : UTF8String;
2940: Func Utf8ToAnsi( const S : UTF8String) : Str;
2941: Func Utf8ToAnsiEx( const S : UTF8String; const cp : Int) : Str;
2942: Func UTF8UpperCase( const S : UTF8String) : UTF8String;
2943: Func ValidFieldIndex( FieldIndex : Int) : Bool;
2944: Func ValidParentForm(control: TControl): TForm;
2945: Func Value : Variant;
2946: Func ValueExists( const Section, Ident : Str) : Bool;
2947: Func ValueOf( const Key : Str) : Int;
2948: Func ValueInSet(AValue: Variant; ASet: Variant): Bool;
2949: Func VALUEOFKEY( const AKEY : VARIANT) : VARIANT;
2950: Func VarArrayFromStrings( Strings : TStrings) : Variant;
2951: Func VarArrayFromWideStrings( Strings : TWideStrings) : Variant;
2952: Func VarArrayGet(var S : Variant; I : Int) : Variant;
2953: Func VarFMTBcd : TVarType;
2954: Func VarFMTBcdCreate1 : Variant;
2955: Func VarFMTBcdCreate2(const AValue : Str; Precision, Scale : Word) : Variant;
2956: Func VarFMTBcdCreate3(const AValue: Double; Precision: Word; Scale: Word): Variant;
2957: Func VarFMTBcdCreate4(const ABcd : TBcd) : Variant;

```



```

2958: Func Variance(const Data : array of Double) : Extended
2959: Func VariantAdd2( const V1 : Variant; const V2 : Variant):Variant
2960: Func VariantAnd2( const V1 : Variant; const V2 : Variant):Variant
2961: Func VariantDiv2( const V1 : Variant; const V2 : Variant):Variant
2962: Func VariantGetElement( const V : Variant; i1 : Int) : Variant;
2963: Func VariantGetElement1( const V : Variant; i1, i2 : Int) : Variant;
2964: Func VariantGetElement2( const V : Variant; i1, i2, i3 : Int) : Variant;
2965: Func VariantGetElement3( const V : Variant; i1, i2, i3, i4 : Int):Variant;
2966: Func VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5: Int):Variant;
2967: Func VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2968: Func VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2969: Func VariantNeg(const V1: Variant) : Variant
2970: Func VariantNot(const V1: Variant) : Variant
2971: Func VariantOr2(const V1: Variant; const V2 : Variant) :Variant
2972: Func VariantShl2(const V1: Variant; const V2 : Variant):Variant
2973: Func VariantShr2(const V1: Variant; const V2 : Variant):Variant
2974: Func VariantSub2(const V1: Variant; const V2 : Variant):Variant
2975: Func VariantXor2(const V1: Variant; const V2 : Variant):Variant
2976: Func VarIsEmpty(const V: Variant):Bool;
2977: Func VarIsFMTBcd( const AValue : Variant) :Bool;
2978: Func VarIsNull(const V: Variant):Bool;
2979: Func VarToBcd( const AValue : Variant) : TBcd
2980: Func VarType(const V: Variant): TVarType;
2981: Func VarType( const V : Variant) : TVarType
2982: Func VarAsType( const V : Variant; AVarType : TVarType): Variant
2983: Func VarIsType( const V : Variant; AVarType : TVarType):Bool;
2984: Func VarIsType1(const V: Variant;const AVarTypes: array of TVarType):Bool;
2985: Func VarIsByRef( const V : Variant) : Bool
2986: Func VarIsEmpty( const V : Variant) : Bool
2987: Proc VarCheckEmpty( const V : Variant)
2988: Func VarIsNull( const V : Variant) : Bool
2989: Func VarIsClear( const V : Variant) : Bool
2990: Func VarIsCustom( const V : Variant) : Bool
2991: Func VarIsOrdinal( const V : Variant) : Bool
2992: Func VarIsFloat( const V : Variant) : Bool
2993: Func VarIsNumeric( const V : Variant) : Bool
2994: Func VarIsStr( const V : Variant) :Bool
2995: Func VarToStr( const V : Variant) :Str
2996: Func VarToStrDef( const V : Variant; const ADefault :Str) :Str
2997: Func VarToWideStr( const V : Variant) : WideString
2998: Func VarToWideStrDef( const V : Variant; const ADefault : WideString):WideString
2999: Func VarToDateTime( const V : Variant) : TDateTime
3000: Func VarFromDateTime( const DateTime : TDateTime) : Variant
3001: Func VarInRange( const AValue, AMin, AMax : Variant) :Bool
3002: Func VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
3003: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual)
3004: Func VarSameValue( const A, B : Variant) :Bool
3005: Func VarCompareValue( const A, B : Variant) : TVariantRelationship
3006: Func VarIsEmptyParam( const V : Variant) :Bool
3007: Func VarIsError( const V : Variant; out AResult : HRESULT):Boolean;
3008: Func VarIsError1( const V : Variant) :Bool;
3009: Func VarAsError( AResult : HRESULT) : Variant
3010: Proc VarCopyNoInd( var Dest : Variant; const Source : Variant)
3011: Func VarIsArray( const A : Variant) :Bool;
3012: Func VarIsArray1( const A : Variant; AResolveByRef :Bool) :Bool;
3013: Func VarArrayCreate( const Bounds : array of Int; AVarType : TVarType):Variant
3014: Func VarArrayOf( const Values : array of Variant) : Variant
3015: Func VarArrayRef( const A : Variant) : Variant
3016: Func VarTypeIsValidArrayType( const AVarType : TVarType):Bool
3017: Func VarTypeIsValidElementType( const AVarType : TVarType):Boolean
3018: Func VarArrayDimCount( const A : Variant) : Int
3019: Func VarArrayLowBound( const A : Variant; Dim : Int) : Int
3020: Func VarArrayHighBound( const A : Variant; Dim : Int) : Int
3021: Func VarArrayLock( const A : Variant) : __Pointer
3022: Proc VarArrayUnlock( const A : Variant)
3023: Func VarArrayGet( const A : Variant; const Indices : array of Int) : Variant
3024: Proc VarArrayPut(var A:Variant;const Value:Variant;const Indices: array of Int)
3025: function VarArrayToStr2(const vArray: variant): string;
3026: function VarStrNull2(const V: OleVariant): string;
3027: function CopyObject(Src, Dest: TObject; Related: Boolean = FALSE): Boolean;
3028: function VariantToString(V : OleVariant) : String;
3029: Proc DynArrayToVariant(var V: Variant;const DynArray: __Pointer;TypeInfo: __Pointer)
3030: Proc DynArrayFromVariant(var DynArray: __Pointer;const V: Variant;TypeInfo: __Pointer)
3031: Func Unassigned : Variant
3032: Func Null : Variant
3033: Func VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
3034: Func VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
3035: Func VectorDot( const V1, V2 : TFloatPoint) : Double
3036: Func VectorDot(const V1,V2: TFloatPoint): Double;
3037: Func VectorLengthSqr( const V : TFloatPoint) : Double
3038: Func VectorLengthSqr(const V: TFloatPoint): Double;
3039: Func VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
3040: Func VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
3041: Func VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
3042: Func VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
3043: Func Verify( AUserName :Str) :Str
3044: Func Versine( X : Float) : Float
3045: Func VersionCheck:Bool;
3046: Func VersionCheckAct:Str;

```

```

3047: Func VersionLanguageId( const LangIdRec : TLangIdRec ) : Str
3048: Func VersionLanguageName( const LangId : Word ) : Str
3049: Func VersionResourceAvailable( const FileName : Str ) : Boolean
3050: Func Visible : Bool
3051: Func VolumeID( DriveChar : Char ) : Str
3052: Func WaitFor( const AString : Str ) : Str
3053: Func WaitFor( const Timeout : Card ) : TJCWaitResult
3054: Func WaitFor1 : TWaitResult;
3055: Func WaitForData( Timeout : Longint ) : Bool
3056: Func WebColorNameToColor( WebColorName : Str ) : TColor
3057: Func WebColorStrToColor( WebColor : Str ) : TColor
3058: Func WebColorToRGB( WebColor : Int ) : Int
3059: Func wGet(aURL, afile:Str):Bool;'
3060: Func wGet2(aURL, afile:Str):Bool;' //without file open
3061: Func wGetX(aURL, afile:Str):Bool;'
3062: Func wGetX2(aURL, afile:Str):Bool;' //without file open
3063: Func WebGet(aURL, afile:Str):Bool;'
3064: Func WebExists:Bool; //alias to isinternet
3065: Func WeekOf( const AValue : TDateTime ) : Word
3066: Func WeekOfTheMonth( const AValue : TDateTime ) : Word;
3067: Func WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word ) : Word;
3068: Func WeekOfTheYear( const AValue : TDateTime ) : Word;
3069: Func WeekOfTheYear1( const AValue : TDateTime; var AYear : Word ) : Word;
3070: Func WeeksBetween( const ANow, ATen : TDateTime ) : Int
3071: Func WeeksInAYear( const AYear : Word ) : Word
3072: Func WeeksInYear( const AValue : TDateTime ) : Word
3073: Func WeekSpan( const ANow, ATen : TDateTime ) : Double
3074: Func WideAdjustLineBreaks( const S : WideString; Style:TTextLineBreakStyle):WideString
3075: Func WideCat( const x, y : WideString ) : WideString
3076: Func WideCompareStr( S1, S2 : WideString ) : Int
3077: Func WideCompareStr(const S1: WideString; const S2: WideString): Int)
3078: Func WideCompareText( S1, S2 : WideString ) : Int
3079: Func WideCompareText(const S1: WideString; const S2: WideString): Int)
3080: Func WideCopy( const src : WideString; index, count : Int ) : WideString
3081: Func WideDequotedStr( const S : WideString; AQuote : WideChar ) : WideString
3082: Func WideEqual( const x, y : WideString ) : Bool
3083: Func WideFormat(const Format: WideString; const Args: array of const):WideString)
3084: Func WideGreater( const x, y : WideString ) : Bool
3085: Func WideLength( const src : WideString ) : Int
3086: Func WideLess( const x, y : WideString ) : Bool
3087: Func WideLowerCase( S : WideString ) : WideString
3088: Func WideLowerCase(const S: WideString): WideString)
3089: Func WidePos( const src, sub : WideString ) : Int
3090: Func WideQuotedStr( const S : WideString; Quote : WideChar ) : WideString
3091: Func WideReplaceStr( const AText, AFromText, AToText : WideString ) : WideString
3092: Func WideReplaceText( const AText, AFromText, AToText : WideString ) : WideString
3093: Func WideSameStr( S1, S2 : WideString ) : Bool
3094: Func WideSameStr(const S1: WideString; const S2: WideString):Bool)
3095: Func WideSameText( S1, S2 : WideString ) : Bool
3096: Func WideSameText(const S1: WideString; const S2: WideString):Bool)
3097: Func WideStringReplace(const S,OldPattern,NewPattern:Widestr;Flags:TReplaceFlags):Widestr)
3098: Func WideStringToUCS4String( const S : WideString ) : UCS4String
3099: Func WideUpperCase( S : WideString ) : WideString
3100: Func Win32BackupFile( const FileName : Str; Move : Bool ) : Bool
3101: Func Win32Check( RetVal : Bool ) : Bool
3102: Func Win32DeleteFile(const FileName:Str; MoveToRecycleBin:Boolean):Bool
3103: Func Win32RestoreFile( const FileName : Str ) : Bool
3104: Func Win32Type : TIdWin32Type
3105: Func WinColor( const Color32 : TColor32 ) : TColor
3106: Func winexec( FileName : pchar; showCmd : Int ) : Int;
3107: Func WinExec32( const Cmd : Str; const CmdShow : Int ) : Bool
3108: Func WinExec32AndWait( const Cmd : Str; const CmdShow : Int ) : Card
3109: Func WithinPastDays( const ANow, ATen : TDateTime; const ADays : Int ) : Bool
3110: Func WithinPastHours(const ANow, ATen:TDateTime; const AHours : Int64): Bool
3111: Func WithinPastMilliseconds(const ANow, ATen:TDateTime;const AMilliSeconds:Int64):Bool
3112: Func WithinPastMinutes(const ANow, ATen:TDateTime;const AMinutes:Int64):Bool
3113: Func WithinPastMonths(const ANow, ATen:TDateTime; const AMonths:Int): Bool
3114: Func WithinPastSeconds(const ANow, ATen: DateTime;const ASeconds:Int64): Bool
3115: Func WithinPastWeeks( const ANow, ATen : TDateTime; const AWeeks : Int ) : Bool
3116: Func WithinPastYears( const ANow, ATen : TDateTime; const AYears : Int ) : Bool
3117: Func WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar ) : DWORD
3118: Func WordToStr( const Value : Word ) : Str
3119: Func WordGridFormatIdentToInt(const Ident:Str; var Value:Longint): Bool
3120: Func IntToWordGridFormatIdent( Value : Longint; var Ident : Str ) : Bool
3121: Proc GetWordGridFormatValues( Proc : TGetStrProc )
3122: Func WorkArea : Int
3123: Func WrapText( Line : Str; MaxCol : Int ) : Str;
3124: Func WrapText2( Line, BreakStr:Str;BreakChars:TSysCharSet;MaxCol:Int):Str;
3125: Func Write( pv : Pointer; cb : Longint; pcbWritten : PLongint ) : HResult
3126: Func Write(Buffer:Str;Count:LongInt):LongInt
3127: Func WriteClient( var Buffer, Count : Int ) : Int
3128: Func WriteFile(const AFile:Str; const AEnableTransferFile:Boolean):Card
3129: Func WriteHeaders(StatusCode: Int; const ReasonString, Headers:Str):Boolean
3130: Func WriteString( const AString : Str ) : Bool
3131: Func WStrGet(var S : AnyString; I : Int): WideChar;
3132: Func wvsprintf( Output : PChar; Format : PChar; arglist : va_list ) : Int
3133: Func wsprintf( Output : PChar; Format : PChar ) : Int
3134: Func XmlDateTimeToStr( const XmlDateTime : Str; const Format:Str):Str
3135: Func XmlTimeToStr( const XmlTime : Str; const Format : Str ) : Str

```

```

3136: Func XorDecode( const Key, Source :Str) :Str
3137: Func XorEncode( const Key, Source :Str) :Str
3138: Func XorString( const Key, Src : ShortString) : ShortString
3139: Func Yield : Bool
3140: Func YearOf( const AValue : TDateTime) : Word
3141: Func YearsBetween( const ANow, AThen : TDateTime) : Int
3142: Func YearSpan( const ANow, AThen : TDateTime) : Double
3143: Func Yesterday : TDateTime
3144: Func YesNoDialog(const ACaption, AMsg:Str):Bool;
3145: Function(const Name :Str; Proc : TUserFunction)
3146: using Special_Scholz from V 3.8.5.0
3147: Func TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3148: Func FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3149: Func FloatToTime2Dec(value:Extended):Extended;
3150: Func MinToStd(value:Extended):Extended;
3151: Func MinToStdAsString(value:Extended):str;
3152: Func RoundFloatToStr(zahl:Extended; decimals:Int):str;
3153: Func RoundFloat(zahl:Extended; decimals:Int):Extended;
3154: Func Round2Dec (zahl:Extended):Extended;
3155: Func GetAngle(x,y:Extended):Double;
3156: Func AddAngle(a1,a2:Double):Double;
3157:
3158: *****
3159: unit uPSI_StText;
3160: *****
3161: Func TextSeek( var F : TextFile; Target : LongInt) :Bool
3162: Func TextFileSize( var F : TextFile) : LongInt
3163: Func TextPos( var F : TextFile) : LongInt
3164: Func TextFlush( var F : TextFile) :Bool
3165:
3166: *****
3167: from JvVCLUtils;
3168: *****
3169: { Windows resources (bitmaps and icons) VCL-oriented routines }
3170: Proc DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Int;Bitmap:TBitmap;TransparentColor:TColor);
3171: Proc DrawBitmapRectTransparent(Dest: TCanvas;DstX,
    DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3172: Proc StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Int; SrcRect: TRect; Bitmap:
    TBitmap; TransparentColor:TColor);
3173: Func MakeBitmap(ResID: PChar): TBitmap;
3174: Func MakeBitmapID(ResID: Word): TBitmap;
3175: Func MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3176: Func CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3177: Func CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3178: Func CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
    HighlightColor, ShadowColor: TColor; DrawHighlight:Bool): TBitmap;
3180: Func CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3181: Func ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3182: Proc AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Int);
3183: {$IFDEF WIN32}
3184: Proc ImageListDrawDisabled/Images: TImageList; Canvas: TCanvas;
3185: X, Y, Index: Int; HighlightColor, GrayColor: TColor; DrawHighlight:Bool);
3186: {$ENDIF}
3187: Func MakeIcon(ResID: PChar): TIcon;
3188: Func MakeIconID(ResID: Word): TIcon;
3189: Func MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3190: Func CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3191: {$IFDEF WIN32}
3192: Func CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3193: {$ENDIF}
3194: { Service routines }
3195: Proc NotImplemented;
3196: Proc ResourceNotFound(ResID: PChar);
3197: Func PointInRect(const P: TPoint; const R: TRect):Bool;
3198: Func PointInPolyRgn(const P: TPoint; const Points: array of TPoint):Bool;
3199: Func PaletteColor(Color: TColor): Longint;
3200: Func WidthOf(R: TRect): Int;
3201: Func HeightOf(R: TRect): Int;
3202: Proc PaintInverseRect(const RectOrg, RectEnd: TPoint);
3203: Proc DrawInvertFrame(ScreenRect: TRect; Width: Int);
3204: Proc CopyParentImage(Control: TControl; Dest: TCanvas);
3205: Proc Delay(MSecs: Longint);
3206: Proc DeleteLine(StrList: TStringList; SearchPattern:Str);
3207: Proc CenterControl(Control: TControl);
3208: Func PaletteEntries( Palette : HPALETTE) : Int
3209: Func WindowClassName( Wnd : HWND) :Str
3210: Func ScreenWorkArea : TRect
3211: Proc MoveWindowOrg( DC : HDC; DX, DY : Int)
3212: Proc SwitchToWindow( Wnd : HWND; Restore :Bool)
3213: Proc ActivateWindow( Wnd : HWND)
3214: Proc ShowWinNoAnimate( Handle : HWND; CmdShow : Int)
3215: Proc CenterWindow( Wnd : HWND)
3216: Proc ShadeRect( DC : HDC; const Rect : TRect)
3217: Proc KillMessage( Wnd : HWND; Msg :Card)
3218: Func DialogsToPixelsX(Dlgs:Word):Word
3219: Func DialogsToPixelsY(Dlgs:Word):Word
3220: Func PixelsToDialogsX(Pixs:Word):Word
3221: Func PixelsToDialogsY(Pixs:Word):Word
3222: {$IFDEF WIN32}

```

```

3223: Proc ShowMDIClientEdge(ClientHandle: THandle; ShowEdge:Bool);
3224: Func MakeVariant(const Values: array of Variant): Variant;
3225: { $SENDIF }
3226: Func CreateRotatedFont(Font: TFont; Angle: Int): HFONT;
3227: Func MsgBox(const Caption, Text:Str; Flags: Int): Int;
3228: Func MsgDlg(const Msg:Str; AType: TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3229: { $IFDEF CBUILDER }
3230: Func FindPrevInstance(const MainFormClass: ShortString; const ATitle:Str): HWND;
3231: Func ActivatePrevInstance(const MainFormClass: ShortString; const ATitle:Str): Boolean;
3232: { $ELSE }
3233: Func FindPrevInstance(const MainFormClass, ATitle:Str): HWND;
3234: Func ActivatePrevInstance(const MainFormClass, ATitle:Str): Bool;
3235: { $ENDIF CBUILDER }
3236: Func IsForegroundTask: Bool;
3237: Proc MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Bool);
3238: Func GetAveCharSize(Canvas: TCanvas): TPoint;
3239: Func MinimizeText(const Text:Str; Canvas: TCanvas; MaxWidth: Int): Str;
3240: Proc FreeUnusedOle;
3241: Proc Beep;
3242: Func GetWindowsVersionJ: Str;
3243: Func LoadDLL(const LibName: Str): THandle;
3244: Func RegisterServer(const ModuleName: Str): Bool;
3245: { $IFDEF WIN32 }
3246: Func IsLibrary: Bool;
3247: { $ENDIF }
3248: { Gradient filling routine }
3249: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3250: Proc GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor,
    EndColor: TColor; Direction: TFillDirection; Colors: Byte);
3251: { String routines }
3252: Func GetEnvVar(const VarName: Str): Str;
3253: Func AnsiUpperFirstChar(const S: Str): Str;
3254: Func StringToPChar(var S: Str): PChar;
3255: Func StrPAlloc(const S: Str): PChar;
3256: Proc SplitCommandLine(const CmdLine: Str; var ExeName, Params: Str);
3257: Func DropT(const S: Str): Str;
3258: { Memory routines }
3259: Func AllocMemo(Size: Longint): Pointer;
3260: Func ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3261: Proc FreeMemo(var fpBlock: Pointer);
3262: Func GetMemoSize(fpBlock: Pointer): Longint;
3263: Func CompareMem(fpBlock1, fpBlock2: Pointer; Size: Card): Bool;
3264: { $IFDEF COMPILER5_UP }
3265: Proc FreeAndNil(var Obj);
3266: { $ENDIF }
3267: // from PNGLoader
3268: Func OptimizeForPNG(Image: TLinearBitmap; QuantizationSteps: Int; TransparentColor: TColor): Int
3269: Proc TransformRGB2LOCO( Image : TLinearBitmap)
3270: Proc TransformLOCO2RGB( Image : TLinearBitmap)
3271: Proc SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3272: Func DrawButtonFace(Canvas: TCanvas; const Client: TRect; BevelWidth: Int; Style: TButtonStyle; IsRounded, IsDown,
    IsFocused: Boolean): TRect // TButtons
3273: Func IsAnAllResult( const AModalResult : TModalResult ) : Bool
3274: Func InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam: Longint): Longint
3275: AddConstantN('CTL3D_ALL', 'LongWord').SetUInt( $FFFF);
3276: //Proc ChangeBiDiModeAlignment( var Alignment : TAlignment)
3277: //Func SendAppMessage( Msg :Card; WParam, LParam : Longint ) : Longint
3278: //Proc MoveWindowOrg( DC : HDC; DX, DY : Int)
3279: Proc SetImeMode( hWnd : HWND; Mode : TImeMode)
3280: Proc SetImeName( Name : TImeName)
3281: Func Win32NLSEnableIME( hWnd : HWND; Enable : Bool ) : Bool
3282: Func Imm32GetContext( hWnd : HWND ) : HIMC
3283: Func Imm32ReleaseContext( hWnd: HWND; hImc : HIMC ) : Bool
3284: Func Imm32GetConversionStatus(hImc : HIMC; var Conversion, Sentence : longword): Bool
3285: Func Imm32SetConversionStatus(hImc : HIMC; Conversion, Sentence : longword): Bool
3286: Func Imm32SetOpenStatus( hImc : HIMC; fOpen : Bool ) : Bool
3287: // Func Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM): Bool
3288: //Func Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA ) : Bool
3289: Func Imm32GetCompositionString(hImc: HIMC; dwWord1: longword; lpBuf: Str; dwBufLen: longint): Longint
3290: Func Imm32IsIME( hKl : longword ) : Bool
3291: Func Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue: longword): Boolean
3292: Proc DragDone( Drop : Bool)
3293:
3294:
3295: //*****add from jvjvclutils
3296: Func CanvasMaxTextHeight(Canvas: TCanvas): Int;
3297: Func ReplaceComponentReference(This, NewReference: TComponent; var VarReference: TComponent): Bool;
3298: Proc DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Int);
3299: Func IsPositiveResult(Value: TModalResult): Bool;
3300: Func IsNegativeResult(Value: TModalResult): Bool;
3301: Func IsAbortResult(const Value: TModalResult): Bool;
3302: Func StripAllFromResult(const Value: TModalResult): TModalResult;
3303: // returns either BrightColor or DarkColor depending on the luminance of AColor
3304: // This Func gives the same result (AFAIK) as the Func used in Windows to
3305: // calculate the desktop icon text color based on the desktop background color
3306: Func SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3307: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3308: Proc HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
    const State: TOwnerDrawState; const Text: Str; var Width: Int;

```

```

3310: CalcType: TJvHTMLCalcType; MouseX, mouseY: Int; var MouseOnLink:Bool;
3311: var LinkName:Str; Scale: Int = 100); overload;
3312: Proc HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3313: const State: TOwnerDrawState; const Text:Str; var Width, Height: Int;
3314: CalcType: TJvHTMLCalcType; MouseX, mouseY: Int; var MouseOnLink:Bool;
3315: var LinkName:Str; Scale: Int = 100); overload;
3316: Func HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3317: const State: TOwnerDrawState; const Text:Str; Scale: Int = 100):Str;
3318: Func HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3319: const State: TOwnerDrawState; const Text:Str; MouseX,mouseY: Int; Scale: Int = 100):Str;
3320: Func HTMLPlainText(const Text:Str):Str;
3321: Func HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3322: const State: TOwnerDrawState; const Text:Str; Scale: Int = 100): TSize;
3323: Func HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3324: const State: TOwnerDrawState; const Text:Str; Scale: Int = 100): Int;
3325: Func HTMLTextHeight(Canvas: TCanvas; const Text:Str;Scale:Int = 100):Int;
3326: Func HTMLPrepareText(const Text:Str):Str;
3327:
3328: ***** uPSI_JvAppUtils;
3329: Func GetDefaultSection( Component : TComponent) :Str
3330: Proc GetDefaultIniData(Control:TControl;var IniFileName,Section:str;UseRegistry:Bool)
3331: Proc GetDefaultIniData( Control : TControl; var IniFileName, Section :Str)
3332: Func GetDefaultIniName :Str
3333: //OnGetDefaultIniName','TOnGetDefaultIniName);
3334: Func GetDefaultIniRegKey :Str
3335: Func FindForm( FormClass : TFormClass) : TForm
3336: Func FindShowForm( FormClass : TFormClass; const Caption :Str) : TForm
3337: Func ShowDialog( FormClass : TFormClass) :Bool
3338: //Func InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3339: Proc SaveFormPlacement( Form : TForm; const IniFileName :Str; UseRegistry :Bool)
3340: Proc RestoreFormPlacement( Form : TForm; const IniFileName :Str; UseRegistry :Bool)
3341: Proc SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3342: Proc SaveFormPlacement( Form : TForm; const IniFileName :Str)
3343: Proc RestoreFormPlacement( Form : TForm; const IniFileName :Str)
3344: Func GetUniqueFileNameInDir( const Path, FileNameMask :Str) :Str
3345: Func StrToIniStr( const Str :Str) :Str
3346: Func IniStrToStr( const Str :Str) :Str
3347: Func IniReadString( IniFile : TObject; const Section, Ident, Default:Str):Str
3348: Proc IniWriteString( IniFile : TObject; const Section, Ident, Value:Str)
3349: Func IniReadInt(IniFile:TObject;const Section, Ident :Str; Default:Longint): Longint
3350: Proc IniWriteInt( IniFile : TObject; const Section, Ident :Str; Value : Longint)
3351: Func IniReadBool(IniFile:TObject; const Section,Ident:str; Default:Boolean):Bool
3352: Proc IniWriteBool( IniFile : TObject; const Section, Ident :Str; Value :Bool)
3353: Proc IniReadSections( IniFile : TObject; Strings : TStrings)
3354: Proc IniEraseSection( IniFile : TObject; const Section :Str)
3355: Proc IniDeleteKey( IniFile : TObject; const Section, Ident :Str)
3356: Proc AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3357: Proc AppBroadcast( Msg, wParam : Word; lParam : Longint)
3358: Proc AppTaskbarIcons( AppOnly :Bool)
3359: Proc InternalSaveGridLayout(Grid : TCustomGrid; IniFile : TObject; const Section:str)
3360: Proc InternalRestoreGridLayout(Grid:TCustomGrid;IniFile : TObject; const Section:str)
3361: Proc InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3362: Proc InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3363: ***** uPSI_JvDBUtils;
3364: Func CreateLocate( DataSet : TDataSet) : TJvLocateObject
3365: Func IsDataSetEmpty( DataSet : TDataSet) :Bool
3366: Proc RefreshQuery( Query : TDataSet)
3367: Func DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:str;CaseInsensitive:Bool):Bool
3368: Func DataSetSectionName( DataSet : TDataSet) :Str
3369: Proc InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section :Str)
3370: Proc InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:str;RestoreVisible:Bool)
3371: Func DataSetLocateThrough(DataSet:TDataSet;const KeyFields:str;const
KeyValues:Variant;Options:TLocateOptions):Bool
3372: Proc SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3373: Proc RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible :Bool)
3374: Proc AssignRecord( Source, Dest : TDataSet; ByName :Bool)
3375: Func ConfirmDelete :Bool
3376: Proc ConfirmDataSetCancel( DataSet : TDataSet)
3377: Proc CheckRequiredField( Field : TField)
3378: Proc CheckRequiredFields( const Fields : array of TField)
3379: Func DateToSQL( Value : TDateTime) :Str
3380: Func FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName :Str) :Str
3381: Func FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName :Str) :Str
3382: Func FormatSQLNumericRange(const FieldName:str;LowVal,HighVal,LowEmpty,HighEmpty:Double;Inclusive:Bool):str
3383: Func StrMaskSQL( const Value :Str) :Str
3384: Func FormatSQLCondition(const FieldName,Operator,Val:str;FieldType:TFieldType;Exact:Bool):str
3385: Func FormatAnsiSQLCondition(const FieldName,Operator,Val:str;FieldType:TFieldType;Exact:Bool):str
3386: Proc _DBError( const Msg :Str)
3387: Const('TrueExpr','String '0=0
3388: Const('sdfStandard16','String '""'mm'/'dd'/'yyyy'""'
3389: Const('sdfStandard32','String '""'dd/mm/yyyy'""'
3390: Const('sdfOracle','String 'TO_DATE('dd/mm/yyyy',' DD/MM/YYYY')'
3391: Const('sdfInterbase','String 'CAST('mm'/'dd'/'yyyy' AS DATE)'
3392: Const('sdfMSSQL','String 'CONVERT(datetime, 'mm'/'dd'/'yyyy',' 103)'
3393: AddTypeS('Largeint', 'Longint
3394: TIFException', (ErNoError, erCannotImport, erInvalidType, erInternalError, '+
3395: erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3396: erOutOfProcRange, erOutOfRange, erOutOfStackRange, erTypeMismatch, erUnexpectedEof, '+
3397: erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutOfRecordRange, '+

```



```

3398: 'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportedederError);
3399: (*-----*)
3400: Proc SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3401: begin
3402:   Func JIniReadBool( const FileName, Section, Line :Str) :Bool
3403:   Func JIniReadInt( const FileName, Section, Line :Str) : Int
3404:   Func JIniReadString( const FileName, Section, Line :Str) :Str
3405:   Proc JIniWriteBool( const FileName, Section, Line :Str; Value :Bool)
3406:   Proc JIniWriteInt( const FileName, Section, Line :Str; Value : Int)
3407:   Proc JIniWriteString( const FileName, Section, Line, Value :Str)
3408:   Proc JIniReadStrings(IniFile:TCustomIniFile; const Section :Str; Strings TStrings)
3409:   Proc JIniWriteStrings(IniFile: TCustomIniFile; const Section:str; Strings:TStrings)
3410: end;
3411:
3412: (* === compile-time registration functions === *)
3413: (*-----*)
3414: Proc SIRegister_JclDateTime(CL: TPSPascalCompiler);
3415: begin
3416:   'UnixTimeStart','LongInt'( 25569);
3417:   Func JEncodeDate( const Year : Int; Month, Day : Word) : TDateTime
3418:   Proc JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3419:   Proc DecodeDate1( Date : TDateTime; var Year : Int; var Month, Day : Word);
3420:   Proc DecodeDate2( Date : TDateTime; var Year, Month, Day : Int);
3421:   Func CenturyOfDate( const DateTime : TDateTime) : Int
3422:   Func CenturyBaseYear( const DateTime : TDateTime) : Int
3423:   Func DayOfDate( const DateTime : TDateTime) : Int
3424:   Func MonthOfDate( const DateTime : TDateTime) : Int
3425:   Func YearOfDate( const DateTime : TDateTime) : Int
3426:   Func JDayOfTheYear( const DateTime : TDateTime; var Year : Int) : Int;
3427:   Func DayOfTheYear1( const DateTime : TDateTime) : Int;
3428:   Func DayOfTheYearToDateTime( const Year, Day : Int) : TDateTime
3429:   Func HourOfTime( const DateTime : TDateTime) : Int
3430:   Func MinuteOfTime( const DateTime : TDateTime) : Int
3431:   Func SecondOfTime( const DateTime : TDateTime) : Int
3432:   Func GetISOYearNumberOfDays( const Year : Word) : Word
3433:   Func IsISOLongYear( const Year : Word) :Bool;
3434:   Func IsISOLongYear1( const DateTime : TDateTime) :Bool;
3435:   Func ISODayOfWeek( const DateTime : TDateTime) : Word
3436:   Func JISOWeekNumber(DateTime:TDateTime; var YearOfWeekNumber,WeekDay: Int) : Int;
3437:   Func ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Int) : Int;
3438:   Func ISOWeekNumber2( DateTime : TDateTime) : Int;
3439:   Func ISOWeekToDateTime( const Year, Week, Day : Int) : TDateTime
3440:   Func JIsLeapYear( const Year : Int) :Bool;
3441:   Func IsLeapYear1( const DateTime : TDateTime) :Bool;
3442:   Func JDaysInMonth( const DateTime : TDateTime) : Int
3443:   Func Make4DigitYear( Year, Pivot : Int) : Int
3444:   Func JMakeYear4Digit( Year, WindowsillYear : Int) : Int
3445:   Func JEasterSunday( const Year : Int) : TDateTime // TDosDateTime, 'Int
3446:   Func JFormatDateTime( Form :Str; DateTime : TDateTime) :Str
3447:   Func FATDatesEqual( const FileTime1, FileTime2 : Int64) :Bool;
3448:   Func FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) :Bool;
3449:   Func HoursToMsecs( Hours : Int) : Int
3450:   Func MinutesToMsecs( Minutes : Int) : Int
3451:   Func SecondsToMsecs( Seconds : Int) : Int
3452:   Func TimeOfDateTimeToSeconds( DateTime : TDateTime) : Int
3453:   Func TimeOfDateTimeToMsecs( DateTime : TDateTime) : Int
3454:   Func DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3455:   Func LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3456:   Func DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3457:   Func JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3458:   Func JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3459:   Proc DateTimeToSystemTime1( DateTime : TDateTime; var SystemTime:TSystemTime);
3460:   Func LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3461:   Func DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3462:   Func JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3463:   Proc DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3464:   Func DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3465:   Func DosDateTimeToStr( DateTime : Int) :Str
3466:   Func JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3467:   Func FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3468:   Func JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3469:   Proc FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time: Word);
3470:   Func JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3471:   Proc FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3472:   Func FileTimeToStr( const FileTime : TFileTime) :Str
3473:   Func SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3474:   Func JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3475:   Proc SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3476:   Func SystemTimeToStr( const SystemTime : TSystemTime) :Str
3477:   Func CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3478:   Func LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3479:   Func LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3480:   TJclUnixTime32', 'Longword
3481:   Func JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3482:   Func JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3483:   Func FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3484:   Func UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3485:   Func JNullStamp : TTimeStamp
3486:   Func JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp): Int64

```

```

3487: Func JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) :Bool
3488: Func JIsNullTimeStamp( const Stamp : TTimeStamp) :Bool
3489: Func TimeStampDOW( const Stamp : TTimeStamp) : Int
3490: Func FirstWeekDay( const Year, Month : Int; var DOW : Int): Int;
3491: Func FirstWeekDay1( const Year, Month : Int) : Int;
3492: Func LastWeekDay( const Year, Month : Int; var DOW : Int): Int;
3493: Func LastWeekDay1( const Year, Month : Int) : Int;
3494: Func IndexedWeekDay( const Year, Month : Int; Index : Int): Int
3495: Func FirstWeekendDay( const Year, Month : Int; var DOW : Int): Int;
3496: Func FirstWeekendDay1( const Year, Month : Int) : Int;
3497: Func LastWeekendDay( const Year, Month : Int; var DOW : Int): Int;
3498: Func LastWeekendDay1( const Year, Month : Int) : Int;
3499: Func IndexedWeekendDay( const Year, Month : Int; Index : Int): Int
3500: Func FirstDayOfWeek( const Year, Month, DayOfWeek : Int) : Int
3501: Func LastDayOfWeek( const Year, Month, DayOfWeek : Int) : Int
3502: Func IndexedDayOfWeek( const Year, Month, DayOfWeek, Index: Int): Int
3503: FindClass('TOBJECT'), 'EJclDateTimeError
3504: end;
3505:
3506: Proc SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3507: begin
3508:   Func SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3509:   Func CreateDOSProcessRedirected(const CommandLine, InputFile, OutputFile: str): Bool
3510:   Func CreateDOSProcessRedirected2(const CommandLine, InputFile, OutputFile: str): Bool
3511:   Func CreateDOSProcessRedirected3(const CommandLine, InputFile, OutputFile, ErrMsg: Str): Bool;
3512:   Func WinExec32( const Cmd : Str; const CmdShow : Int) : Bool
3513:   Func WinExec32AndWait( const Cmd : Str; const CmdShow : Int) : Card
3514:   Func WinExec32AndRedirectOutput(const Cmd: str; var Output: str; RawOutput: Bool): Card
3515:   TJclKillLevel, '( klNormal, klNoSignal, klTimeout )
3516:   Func ExitWindows( ExitCode : Card) : Bool
3517:   Func LogOffOS( KillLevel : TJclKillLevel) : Bool
3518:   Func PowerOffOS( KillLevel : TJclKillLevel) : Bool
3519:   Func ShutDownOS( KillLevel : TJclKillLevel) : Bool
3520:   Func RebootOS( KillLevel : TJclKillLevel) : Bool
3521:   Func HibernateOS( Force, DisableWakeEvents: Bool) : Bool
3522:   Func SuspendOS( Force, DisableWakeEvents: Bool) : Bool
3523:   Func ShutDownDialog(const DialogMessage: str; TimeOut: DWORD; Force, Reboot: Bool): Bool;
3524:   Func ShutDownDialog1(const MachineName, DialogMessage: str; TimeOut: DWORD; Force, Reboot: Bool): Bool;
3525:   Func AbortShutDown : Bool;
3526:   Func AbortShutDown1( const MachineName : Str) : Bool;
3527:   TJclAllowedPowerOperation, '( apoHibernate, apoShutdown, apoSuspend )
3528:   TJclAllowedPowerOperations, 'set of TJclAllowedPowerOperation
3529:   Func GetAllowedPowerOperations : TJclAllowedPowerOperations
3530:   FindClass('TOBJECT'), 'EJclCreateProcessError
3531:   Proc CreateProcAsUser( const UserDomain, UserName, Password, CommandLine : Str)
3532:   Proc CreateProcAsUserEx(const UserDomain, UserName, Password, CommandLine: str; const Environment: PChar);
3533:   // with Add(EJclCreateProcessError) do
3534: end;
3535:
3536: Proc SIRegister_JclAnsistrs(CL: TPSPascalCompiler);
3537: begin
3538:   // 'AnsiSigns', 'Set').SetSet(['-', '+']);
3539:   'C1_UPPER', 'LongWord( $0001);
3540:   'C1_LOWER', 'LongWord( $0002);
3541:   'C1_DIGIT', 'LongWord').SetUInt( $0004);
3542:   'C1_SPACE', 'LongWord').SetUInt( $0008);
3543:   'C1_PUNCT', 'LongWord').SetUInt( $0010);
3544:   'C1_CNTRL', 'LongWord').SetUInt( $0020);
3545:   'C1_BLANK', 'LongWord').SetUInt( $0040);
3546:   'C1_XDIGIT', 'LongWord').SetUInt( $0080);
3547:   'C1_ALPHA', 'LongWord').SetUInt( $0100);
3548:   AnsiChar, 'Char
3549:   Func StrIsAlpha( const S : Ansistr) : Bool
3550:   Func StrIsAlphaNum( const S : Ansistr) : Bool
3551:   Func StrIsAlphaNumUnderscore( const S : Ansistr) : Bool
3552:   Func StrContainsChars(const S: Ansistr; Chars: TSysCharSet; CheckAll: Boolean): Bool
3553:   Func StrConsistsOfNumberChars( const S : Ansistr) : Bool
3554:   Func StrIsDigit( const S : Ansistr) : Bool
3555:   Func StrIsSubset( const S : Ansistr; const ValidChars : TSysCharSet): Bool
3556:   Func StrSame( const S1, S2 : Ansistr) : Bool
3557:   //Func StrCenter( const S : Ansistr; L : Int; C : AnsiChar) : Ansistr
3558:   Func StrCharPosLower( const S : Ansistr; CharPos : Int) : Ansistr
3559:   Func StrCharPosUpper( const S : Ansistr; CharPos : Int) : Ansistr
3560:   Func StrDoubleQuote( const S : Ansistr) : Ansistr
3561:   Func StrEnsureNoPrefix( const Prefix, Text : Ansistr) : Ansistr
3562:   Func StrEnsureNoSuffix( const Suffix, Text : Ansistr) : Ansistr
3563:   Func StrEnsurePrefix( const Prefix, Text : Ansistr) : Ansistr
3564:   Func StrEnsureSuffix( const Suffix, Text : Ansistr) : Ansistr
3565:   Func StrEscapedToString( const S : Ansistr) : Ansistr
3566:   Func JStrLower( const S : Ansistr) : Ansistr
3567:   Proc StrLowerInPlace( var S : Ansistr)
3568:   //Proc StrLowerBuff( S : PAnsiChar)
3569:   Proc JStrMove( var Dest: Ansistr; const Source: Ansistr; const ToIndex, FromIndex, Count: Int;
3570:   Func StrPadLeft( const S : Ansistr; Len : Int; C : AnsiChar) : Ansistr
3571:   Func StrPadRight( const S : Ansistr; Len : Int; C : AnsiChar) : Ansistr
3572:   Func StrProper( const S : Ansistr) : Ansistr
3573:   //Proc StrProperBuff( S : PAnsiChar)
3574:   Func StrQuote( const S : Ansistr; C : AnsiChar) : Ansistr
3575:   Func StrRemoveChars( const S : Ansistr; const Chars : TSysCharSet) : Ansistr

```

```

3576: Func StrKeepChars( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
3577: Proc JStrReplace(var S:Ansistr;const Search,Replace:Ansistr;Flags: TReplaceFlags)
3578: Func StrReplaceChar( const S : Ansistr; const Source, Replace : AnsiChar) : Ansistr
3579: Func StrReplaceChars(const S:Ansistr;const Chars:TSysCharSet;Replace:AnsiChar):Ansistr
3580: Func StrReplaceButChars(const S:Ansistr;const Chars:TSysCharSet;Replace:AnsiChar):Ansistr;
3581: Func StrRepeat( const S : Ansistr; Count : Int) : Ansistr
3582: Func StrRepeatLength( const S : Ansistr; const L : Int) : Ansistr
3583: Func StrReverse( const S : Ansistr) : Ansistr
3584: Proc StrReverseInPlace( var S : Ansistr)
3585: Func StrSingleQuote( const S : Ansistr) : Ansistr
3586: Func StrSmartCase( const S : Ansistr; Delimiters : TSysCharSet) : Ansistr
3587: Func StrStringToEscaped( const S : Ansistr) : Ansistr
3588: Func StrStripNonNumberChars( const S : Ansistr) : Ansistr
3589: Func StrToHex( const Source : Ansistr) : Ansistr
3590: Func StrTrimCharLeft( const S : Ansistr; C : AnsiChar) : Ansistr
3591: Func StrTrimCharsLeft( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
3592: Func StrTrimCharRight( const S : Ansistr; C : AnsiChar) : Ansistr
3593: Func StrTrimCharsRight( const S : Ansistr; const Chars : TSysCharSet) : Ansistr
3594: Func StrTrimQuotes( const S : Ansistr) : Ansistr
3595: Func JStrUpper( const S : Ansistr) : Ansistr
3596: Proc StrUpperInPlace( var S : Ansistr)
3597: //Proc StrUpperBuff( S : PAnsiChar)
3598: Func StrOemToAnsi( const S : Ansistr) : Ansistr
3599: Func StrAnsiToOem( const S : Ansistr) : Ansistr
3600: Proc StrAddRef( var S : Ansistr)
3601: Func StrAllocSize( const S : Ansistr) : Longint
3602: Proc StrDecRef( var S : Ansistr)
3603: //Func StrLen( S : PAnsiChar) : Int
3604: Func StrLength( const S : Ansistr) : Longint
3605: Func StrRefCount( const S : Ansistr) : Longint
3606: Proc StrResetLength( var S : Ansistr)
3607: Func StrCharCount( const S : Ansistr; C : AnsiChar) : Int
3608: Func StrCharsCount( const S : Ansistr; Chars : TSysCharSet) : Int
3609: Func StrStrCount( const S, SubS : Ansistr) : Int
3610: Func StrCompare( const S1, S2 : Ansistr) : Int
3611: Func StrCompareRange( const S1, S2 : Ansistr; const Index, Count:Int) : Int
3612: //Func StrFillChar( const C : AnsiChar; Count : Int) : Ansistr;
3613: Func StrFillChar1( const C : Char; Count : Int) : Ansistr;
3614: Func StrFillChar(const C: Char; Count: Int):Str;
3615: Func IntFillChar(const I: Int; Count: Int):Str;
3616: Func ByteFillChar(const B: Byte; Count: Int):Str;
3617: Func ArrFillChar(const AC: Char; Count: Int): TCharArray;;
3618: Func ArrByteFillChar(const AB: Char; Count: Int): TByteArray;
3619: Func StrFind( const Substr, S : Ansistr; const Index : Int) : Int
3620: //Func StrHasPrefix( const S : Ansistr; const Prefixes : array of Ansistr) :Bool
3621: Func StrIndex( const S : Ansistr; const List : array of Ansistr) : Int
3622: Func StrILastPos( const SubStr, S : Ansistr) : Int
3623: Func StrIPos( const SubStr, S : Ansistr) : Int
3624: Func StrIsOneOf( const S : Ansistr; const List : array of Ansistr) :Bool
3625: Func StrLastPos( const SubStr, S : Ansistr) : Int
3626: Func StrMatch( const Substr, S : Ansistr; const Index : Int) : Int
3627: Func StrMatches( const Substr, S : Ansistr; const Index : Int) :Bool
3628: Func StrNIPos( const S, SubStr : Ansistr; N : Int) : Int
3629: Func StrNPos( const S, SubStr : Ansistr; N : Int) : Int
3630: Func StrPrefixIndex( const S : Ansistr; const Prefixes : array of Ansistr) : Int
3631: Func StrSearch( const Substr, S : Ansistr; const Index : Int) : Int
3632: //Func StrAfter( const SubStr, S : Ansistr) : Ansistr
3633: //Func StrBefore( const SubStr, S : Ansistr) : Ansistr
3634: Func StrBetween( const S : Ansistr; const Start, Stop : AnsiChar) : Ansistr
3635: Func StrChopRight( const S : Ansistr; N : Int) : Ansistr
3636: Func StrLeft( const S : Ansistr; Count : Int) : Ansistr
3637: Func StrMid( const S : Ansistr; Start, Count : Int) : Ansistr
3638: Func StrRestOf( const S : Ansistr; N : Int) : Ansistr
3639: Func StrRight( const S : Ansistr; Count : Int) : Ansistr
3640: Func CharEqualNoCase( const C1, C2 : AnsiChar) :Bool
3641: Func CharIsAlpha( const C : AnsiChar) :Bool
3642: Func CharIsAlphaNum( const C : AnsiChar) :Bool
3643: Func CharIsBlank( const C : AnsiChar) :Bool
3644: Func CharIsControl( const C : AnsiChar) :Bool
3645: Func CharIsDelete( const C : AnsiChar) :Bool
3646: Func CharIsDigit( const C : AnsiChar) :Bool
3647: Func CharIsLower( const C : AnsiChar) :Bool
3648: Func CharIsNumberChar( const C : AnsiChar) :Bool
3649: Func CharIsPrintable( const C : AnsiChar) :Bool
3650: Func CharIsPunctuation( const C : AnsiChar) :Bool
3651: Func CharIsReturn( const C : AnsiChar) :Bool
3652: Func CharIsSpace( const C : AnsiChar) :Bool
3653: Func CharIsUpper( const C : AnsiChar) :Bool
3654: Func CharIsWhiteSpace( const C : AnsiChar) :Bool
3655: Func CharType( const C : AnsiChar) : Word
3656: Func CharHex( const C : AnsiChar) : Byte
3657: Func CharLower( const C : AnsiChar) : AnsiChar
3658: Func CharUpper( const C : AnsiChar) : AnsiChar
3659: Func CharToggleCase( const C : AnsiChar) : AnsiChar
3660: Func CharPos( const S : Ansistr; const C : AnsiChar; const Index : Int) : Int
3661: Func CharLastPos( const S : Ansistr; const C : AnsiChar; const Index : Int) : Int
3662: Func CharIPos( const S : Ansistr; C : AnsiChar; const Index : Int) : Int
3663: Func CharReplace( var S : Ansistr; const Search, Replace : AnsiChar) : Int
3664: Proc StrIToStrings(S,Sep:Ansistr; const List:TStrings;const AllowEmptyString :Bool)

```

```

3665: Proc StrToStrings(S,Sep: AnsiStr; const List:TStrings;const AllowEmptyString :Bool)
3666: Func StringsToStr(const List:TStrings;const Sep:AnsiStr;const AllowEmptyString:Bool):AnsiStr;
3667: Proc TrimStrings(const List : TStrings; DeleteIfEmpty :Bool)
3668: Proc TrimStringsRight(const List : TStrings; DeleteIfEmpty :Bool)
3669: Proc TrimStringsLeft(const List : TStrings; DeleteIfEmpty :Bool)
3670: Func AddStringToStrings(const S:AnsiStr;Strings:TStrings; const Unique:Boolean):Bool
3671: Func BooleanToStr( B :Bool) : AnsiStr
3672: Func FileToString(const FileName : AnsiStr) : AnsiStr
3673: Proc StringToFile(const FileName, Contents: AnsiStr; Append:Bool) //binary save!
3674: Proc StringToFile2(const S, FileName :Str);
3675: Func StrToken(var S : AnsiStr; Separator : AnsiChar) : AnsiStr
3676: Proc StrTokens(const S : AnsiStr; const List : TStrings)
3677: Proc StrTokenToStrings(S : AnsiStr; Separator : AnsiChar; const List : TStrings)
3678: //Func StrWord(var S : PAnsiChar; out Word : AnsiStr) :Bool
3679: Func StrToFloatSafe(const S : AnsiStr) : Float
3680: Func StrToIntSafe(const S : AnsiStr) : Int
3681: Proc StrNormIndex(const StrLen : Int; var Index : Int; var Count : Int);
3682: Func ArrayOf(List : TStrings) : TDynStringArray;
3683: EJclStringError, 'EJclError
3684: Func IsClass(Address: TObject):Bool;
3685: Func IsObject(Address: TObject):Bool;
3686: // Console Utilities
3687: //Func ReadKey: Char;
3688: Func IntToStrZeroPad(Value, Count: Int): AnsiStr;
3689: Func JclGUIDToString(const GUID: TGUID):Str;
3690: Func JclStringToGUID(const S:Str): TGUID;
3691: end;
3692:
3693: ***** uPSI_JvDBUtil;
3694: Proc ExecuteSQLScript(Base:TDataBase;const Script:sTr; const Commit:TCommit;OnProgress:TOnProgress; const
UserData : Int)
3695: Func GetQueryResult(const DatabaseName, SQL :Str) : Variant
3696: Func GetStoredProcResult(const ADatabaseName,AStoredProcName:str; AParams: array of Variant;const
AResultName:str):Variant
3697: //Func StrFieldDesc(Field : FLDDesc) :Str
3698: Func Var2Type(V : Variant; const VarType : Int) : Variant
3699: Proc CopyRecord(DataSet : TDataSet)
3700: //Proc AddReference(Tbl : TTable; RefName:Str; RefField: Word; MasterTable :Str; MasterField : Word;
ModOp, DelOp : RINTQual)
3701: Proc AddMasterPassword(Table : TTable; pswd :Str)
3702: Proc PackTable(Table : TTable)
3703: Proc PackEncryptedTable(Table : TTable; pswd :Str)
3704: Func EncodeQuotes(const S :Str) :Str
3705: Func Cmp(const S1, S2 :Str) :Bool
3706: Func SubStr(const S :Str; const Index : Int; const Separator :Str) :Str
3707: Func SubStrEnd(const S :Str; const Index : Int; const Separator :Str):Str
3708: Func ReplaceString(S :Str; const OldPattern, NewPattern :Str) :Str
3709: Proc GetXYByPos(const S :Str; const Pos : Int; var X, Y : Int)
3710: *****uPSI_JvJvBDEUtils;*****
3711: //JvBDEUtils
3712: Func CreateDbLocate : TJvLocateObject
3713: //Func CheckOpen(Status : DBIResult) :Bool
3714: Proc FetchAllRecords(DataSet : TBDEDataSet)
3715: Func TransActive(Database : TDatabase) :Bool
3716: Func AsyncQrySupported(Database : TDatabase) :Bool
3717: Func GetQuoteChar(Database : TDatabase) :Str
3718: Proc ExecuteQuery(const DbName, QueryText :Str)
3719: Proc ExecuteQueryEx(const SessName, DbName, QueryText :Str)
3720: Func FieldLogicMap(FldType : TFieldType) : Int
3721: Func FieldSubtypeMap(FldType : TFieldType) : Int Value :Str; Buffer : Pointer)
3722: Func GetAliasPath(const AliasName :Str) :Str
3723: Func IsDirectory(const DatabaseName :Str) :Bool
3724: Func GetBdeDirectory :Str
3725: Func LoginToDatabase(Database : TDatabase; OnLogin : TDatabaseLoginEvent) :Bool
3726: Func DataSetFindValue(ADataset : TBDEDataSet; const Value, FieldName :Str) :Bool
3727: Func DataSetFindLike(ADataset : TBDEDataSet; const Value, FieldName :Str) :Bool
3728: Func DataSetRecNo(DataSet : TDataSet) : Longint
3729: Func DataSetRecordCount(DataSet : TDataSet) : Longint
3730: Func DataSetPositionStr(DataSet : TDataSet) :Str
3731: Proc DataSetShowDeleted(DataSet : TBDEDataSet; Show :Bool)
3732: Func CurrentRecordDeleted(DataSet : TBDEDataSet) :Bool
3733: Func IsFilterApplicable(DataSet : TDataSet) :Bool
3734: Func IsBookmarkStable(DataSet : TBDEDataSet) :Bool
3735: Proc SetIndex(Table : TTable; const IndexFieldNames :Str)
3736: Proc RestoreIndex(Table : TTable)
3737: Proc DeleteRange(Table: TTable; IndexFields:array of const; FieldValues: array of const)
3738: Proc PackTable(Table : TTable)
3739: Proc ReindexTable(Table : TTable)
3740: Proc BdeFlushBuffers
3741: Func GetNativeHandle(Database : TDatabase; Buffer : Pointer; BufSize : Int) : Pointer
3742: Proc ToggleDebugLayer(Active :Bool; const DebugFile :Str)
3743: Proc DbNotSupported
3744: Proc ExportDataSet(Source:TBDEDataSet;DestTable:TTable;TableType:TTableType;const AsciiCharSe
:str;AsciiDelimited Boolean;MaxRecordCount:Longint)
3745: Proc ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
AsciiCharSet:str;AsciiDelimited:Boolean;AsciiDelimiter;AsciiSeparator:Char;MaxRecordCount:Longint);
3746: Proc
ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3747: Proc InitRSRUN(Database:TDatabase;const ConName:str;ConType:Int;const ConServer:str);

```

```

3748: *****uPSI_JvDateUtil;
3749: Func CurrentYear: Word;
3750: Func IsLeapYear(AYear: Int):Bool;
3751: Func DaysPerMonth(AYear, AMonth: Int): Int;
3752: Func FirstDayOfPrevMonth: TDateTime;
3753: Func LastDayOfPrevMonth: TDateTime;
3754: Func FirstDayOfNextMonth: TDateTime;
3755: Func ExtractDay(ADate: TDateTime): Word;
3756: Func ExtractMonth(ADate: TDateTime): Word;
3757: Func ExtractYear(ADate: TDateTime): Word;
3758: Func IncDate(ADate: TDateTime; Days, Months, Years: Int): TDateTime;
3759: Func IncDay(ADate: TDateTime; Delta: Int): TDateTime;
3760: Func IncMonth(ADate: TDateTime; Delta: Int): TDateTime;
3761: Func IncYear(ADate: TDateTime; Delta: Int): TDateTime;
3762: Func ValidDate(ADate: TDateTime):Bool;
3763: Proc DateDiff(Datel, Date2: TDateTime; var Days, Months, Years: Word);
3764: Func MonthsBetween(Datel, Date2: TDateTime): Double;
3765: Func DaysInPeriod(Datel, Date2: TDateTime): Longint;
3766: { Count days between Delat and Date2 + 1, so if Delat = Date2 result = 1 }
3767: Func DaysBetween(Datel, Date2: TDateTime): Longint;
3768: { The same as previous but if Date2 < Delat result = 0 }
3769: Func IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Int): TDateTime;
3770: Func IncHour(ATime: TDateTime; Delta: Int): TDateTime;
3771: Func IncMinute(ATime: TDateTime; Delta: Int): TDateTime;
3772: Func IncSecond(ATime: TDateTime; Delta: Int): TDateTime;
3773: Func IncMSec(ATime: TDateTime; Delta: Int): TDateTime;
3774: Func CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3775: { String to date conversions }
3776: Func GetDateOrder(const DateFormat:Str): TDateOrder;
3777: Func MonthFromName(const S:Str; MaxLen: Byte): Byte;
3778: Func StrToDateDef(const S:Str; Default: TDateTime): TDateTime;
3779: Func StrToDateFmt(const DateFormat, S:Str): TDateTime;
3780: Func StrToDateFmtDef(const DateFormat, S:Str; Default: TDateTime): TDateTime;
3781: Func DefDateFormat(FourDigitYear:Bool):Str;
3782: Func DefDateMask(BlanksChar: Char; FourDigitYear:Bool):Str;
3783: -----
3784: ***** JvUtils;*****
3785: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3786: Func GetWordOnPos(const S:Str; const P: Int):Str;
3787: {GetWordOnPosEx worklike GetWordOnPosfunction,also returns Wordposition in iBeg,iEnd variables}
3788: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
3789: { SubStr returns substring from string, S, separated with Separator string}
3790: Func SubStr(const S:Str; const Index: Int; const Separator:Str):Str;
3791: { SubStrEnd same to previous Func but Index numerated from the end of string }
3792: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
3793: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3794: Func SubWord(P: PChar; var P2: PChar):Str;
3795: { NumberByWord returns the text representation of
3796:   the number, N, in normal russian language. Was typed from Monitor magazine }
3797: Func NumberByWord(const N: Longint):Str;
3798: // Func CurrencyByWord(Value : Currency) :Str; GetLineByPos returns Line number, there
3799: //the symbol Pos is pointed. Lines separated with #13 symbol }
3800: Func GetLineByPos(const S:Str; const Pos: Int): Int;
3801: { GetXYByPos is same to previous function, but returns X position in line too}
3802: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
3803: { ReplaceString searches for all substrings,OldPattern,in a string,S, replaces them with NewPattern }
3804: Func ReplaceString(S:Str; const OldPattern, NewPattern:Str):Str;
3805: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3806: Func ConcatSep(const S, S2, Separator:Str):Str;
3807: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3808: Func ConcatLeftSep(const S, S2, Separator:Str):Str;
3809: { MinimizeString truncs long string,S,appends '...' symbols,if Length of S is more than MaxLen}
3810: Func MinimizeString(const S:Str; const MaxLen: Int):Str;
3811: { Next 4 Func for russian chars transliterating.
3812:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3813: Proc Dos2Win(var S:Str);
3814: Proc Win2Dos(var S:Str);
3815: Func Dos2WinRes(const S:Str):Str;
3816: Func Win2DosRes(const S:Str):Str;
3817: Func Win2Koi(const S:Str):Str;
3818: { Spaces returns string consists on N space chars }
3819: Func Spaces(const N: Int):Str;
3820: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3821: Func AddSpaces(const S:Str; const N: Int):Str;
3822: { Func LastDate for russian users only } { returns date relative to current date: '' }
3823: Func LastDate(const Dat: TDateTime):Str;
3824: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3825: Func CurrencyToStr(const Cur: currency):Str;
3826: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3827: Func Cmp(const S1, S2:Str):Bool;
3828: { StringCat add S2 string to S1 and returns this string }
3829: Func StringCat(var S1:Str; S2:Str):Str;
3830: { HasChar returns True, if Char, Ch, contains in string, S }
3831: Func HasChar(const Ch: Char; const S:Str):Bool;
3832: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
3833: Func CharInSet(const Ch: Char; const SetOfChar: TSetOfChar):Bool;
3834: Func CountOfChar(const Ch: Char; const S:Str): Int;
3835: Func DefStr(const S:Str; Default:Str):Str;
3836: {**** files routines}

```



```

3837: { GetWinDir returns Windows folder name }
3838: Func GetWinDir: TFileName;
3839: Func GetSysDir:Str;
3840: { GetTempDir returns Windows temporary folder name }
3841: Func GetTempDir:Str;
3842: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3843: Func GenTempFileName(FileName:Str):Str;
3844: { GenTempFileNameExt sameto previous function,but returning filename has given extension, FileExt }
3845: Func GenTempFileNameExt(FileName:Str; const FileExt:Str):Str;
3846: { ClearDir clears folder Dir }
3847: Func ClearDir(const Dir:Str):Bool;
3848: { DeleteDir clears and than delete folder Dir }
3849: Func DeleteDir(const Dir:Str):Bool;
3850: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3851: Func FileEquMask(FileName, Mask: TFileName):Bool;
3852: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3853:   Masks must be separated with comma (',' ) }
3854: Func FileEquMasks(FileName, Masks: TFileName):Bool;
3855: Proc DeleteFiles(const Folder: TFileName; const Masks:Str);
3856: { LZFileExpand expand file,FileSource into FileDest.File must be compressed,using MS Compress program }
3857: Func LZFileExpand(const FileSource, FileDest:Str):Bool;
3858: { FileGetInfo fills SearchRec record for specified file attributes}
3859: Func FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec):Bool;
3860: { HasSubFolder returns True, if folder APath contains other folders }
3861: Func HasSubFolder(APath: TFileName):Bool;
3862: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3863: Func IsEmptyFolder(APath: TFileName):Bool;
3864: { AddSlash add slash Char to Dir parameter, if needed }
3865: Proc AddSlash(var Dir: TFileName);
3866: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3867: Func AddSlash2(const Dir: TFileName):Str;
3868: { AddPath returns FileName with Path, if FileName not contain any path }
3869: Func AddPath(const FileName, Path: TFileName): TFileName;
3870: Func AddPaths(const PathList, Path:Str):Str;
3871: Func ParentPath(const Path: TFileName): TFileName;
3872: Func FindInPath(const FileName, PathList:Str): TFileName;
3873: Func FindInPaths(const fileName,paths:Str):Str;
3874: {$IFDEF BCB1}
3875: { BrowseForFolder displays Browse For Folder dialog }
3876: Func BrowseForFolder(const Handle: HWND; const Title:Str; var Folder:Str):Boolean;
3877: {$ENDIF BCB1}
3878: Func BrowseForFolder(const ATitle:Str; AllowCreate :Bool; var ADirectory :str; AHelpContext :
  THelpContext) :Bool
3879: Func BrowseForComputer(const ATitle :Str; AllowCreate :Bool; var ADirectory:str; AHelpContext :
  THelpContext): Bool
3880: Func BrowseDirectory(var AFolderName:str;const DlgText:str;AHelpContext:THelpContext):Bool
3881: Func BrowseComputer(var AComputerName:str;const DlgText:str;AHelpContext:THelpContext):Bool
3882: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3883: Func DeleteReadOnlyFile(const FileName: TFileName):Bool;
3884: { HasParam returns True, if program running with specified parameter, Param }
3885: Func HasParam(const Param:Str):Bool;
3886: Func HasSwitch(const Param:Str):Bool;
3887: Func Switch(const Param:Str):Str;
3888: { ExePath returns ExtractFilePath(ParamStr(0)) }
3889: Func ExePath: TFileName;
3890: Func CopyDir(const SourceDir, DestDir: TFileName):Bool;
3891: Func FileTimeToDateTime(const FT: TFileTime): TDateTime;
3892: Func MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3893: {*** Graphic routines }
3894: { TTFontSelected returns True, if True Type font is selected in specified device context }
3895: Func TTFontSelected(const DC: HDC):Bool;
3896: { TrueInflateRect inflates rect in other method, than InflateRect API Func }
3897: Func TrueInflateRect(const R: TRect; const I: Int): TRect;
3898: {*** Windows routines }
3899: { SetWindowTop put window to top without recreating window }
3900: Proc SetWindowTop(const Handle: HWND; const Top:Bool);
3901: {*** other routines }
3902: { KeyPressed returns True, if Key VK is now pressed }
3903: Func KeyPressed(VK: Int):Bool;
3904: Proc SwapInt(var Int1, Int2: Int);
3905: Func IntPower(Base, Exponent: Int): Int;
3906: Func ChangeTopException(E: TObject): TObject;
3907: Func StrToBool(const S:Str):Bool;
3908: {$IFDEF COMPILER3_UP}
3909: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3910:   Length of MaxLen bytes. The compare operation is controlled by the
3911:   current Windows locale. The return value is the same as for CompareStr. }
3912: Func AnsiStrLIComp(S1, S2: PChar; MaxLen:Card): Int;
3913: Func AnsiStrIComp(S1, S2: PChar): Int;
3914: {$ENDIF}
3915: Func Var2Type(V: Variant; const VarType: Int): Variant;
3916: Func VarToInt(V: Variant): Int;
3917: Func VarToFloat(V: Variant): Double;
3918: {following functions are not documented because they are don't work properly,so don't use them}
3919: Func ReplaceSokrl(S:Str; const Word, Frase:Str):Str;
3920: { ReplaceSokrl is full equal to ReplaceString Func - only for compatibility - don't use }
3921: { GetSubStr is full equal to SubStr Func - only for compatibility - don't use }
3922: Func GetSubStr(const S:Str; const Index: Int; const Separator: Char):Str;
3923: Func GetParameter:Str;

```

```

3924: Func GetLongFileName(FileName:Str):Str;
3925: { * from FileCtrl }
3926: Func DirectoryExists(const Name:Str):Bool;
3927: Proc ForceDirectories(Dir:Str);
3928: { # from FileCtrl }
3929: Func FileNewExt(const FileName, NewExt: TFileName): TFileName;
3930: Func GetComputerID:Str;
3931: Func GetComputerName:Str;
3932: { **** string routines }
3933: { ReplaceAllSokr searches for all substrings, Words, in a string, S, and replaces them with Frases with the
same Index. Also see RAUtilsW.ReplaceSokr1 Func }
3934: Func ReplaceAllSokr(S:Str; Words, Frases: TStrings):Str;
3935: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3936: in the list, Words, and if founds, replaces this Word with string from another list, Frases, with same
Index, and then update NewSelStart variable }
3937: Func ReplaceSokr(S:Str; PosBeg, Len: Int; Words, Frases: TStrings; var NewSelStart: Int): Str;
3938: { CountOfLines calculates linescount in a string, each line separated from another with CrLf sequence }
3939: Func CountOfLines(const S:Str): Int;
3940: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3941: Proc DeleteEmptyLines(Ss: TStrings);
3942: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3943: Note: If strings SQL already contains where-statement, it must be started on begining of any line }
3944: Proc SQLAddWhere(SQL: TStrings; const Where:Str);
3945: { **** files routines - }
3946: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3947: Resource can be compressed using MS Compress program }
3948: Func ResSaveToFile(const Typ, Name: Str; const Compressed: Bool; const FileName: Str): Bool;
3949: Func ResSaveToFileEx(Inst: HINST; Typ, Name: PChar; const Compressed: Bool; const FileName: Str): Bool
3950: Func ResSaveToString(Instance: HINST; const Typ, Name: Str; var S: Str): Bool;
3951: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3952: Func ExecuteJ(const CommandLine, WorkingDirectory: Str): Int;
3953: { IniReadSection read section, Section, from ini-file,
3954: IniFileName, into strings, Ss. This Func reads ALL strings from specified section.
3955: Note: TIniFile.ReadSection Func reads only strings with '=' symbol. }
3956: Func IniReadSection(const IniFileName: TFileName; const Section: Str; Ss: TStrings): Bool;
3957: { LoadTextFile load text file, FileName, into string }
3958: Func LoadTextFile(const FileName: TFileName): Str;
3959: Proc SaveTextFile(const FileName: TFileName; const Source: Str);
3960: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList }
3961: Func ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Int;
3962: Func ReadFolders(const Folder: TFileName; FolderList: TStrings): Int;
3963: { $IFDEF COMPILER3 UP }
3964: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3965: Func TargetFileName(const FileName: TFileName): TFileName;
3966: { return filename ShortCut linked to }
3967: Func ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3968: { $ENDIF COMPILER3 UP }
3969: { **** Graphic routines - }
3970: { LoadIcoToImage loads 2 icons from resource named NameRes, into twoimage lists ALarge and ASmall }
3971: Proc LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: Str);
3972: { RAtextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3973: Proc RAtextOut(Canvas: TCanvas; const R, RClip: TRect; const S: Str);
3974: { RAtextOutEx same with RAtextOut funct, but can calculate needed height for correct output }
3975: Func RAtextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: Str; const CalcHeight: Bool): Int;
3976: { RAtextCalcHeight calculate needed height to correct output, using RAtextOut or RAtextOutEx functions }
3977: Func RAtextCalcHeight(Canvas: TCanvas; const R: TRect; const S: Str): Int;
3978: { Cinema draws some visual effect }
3979: Proc Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3980: { Roughed fills rect with special 3D pattern }
3981: Proc Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Bool);
3982: { 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 }
3983: Func BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Int): TBitmap;
3984: { TextWidth calculate text with for writing using standard desktop font }
3985: Func TextWidth(AStr: Str): Int;
3986: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3987: Func DefineCursor(Identifier: PChar): TCursor;
3988: { **** other routines - }
3989: { FindFormByClass returns first form specified class, FormClass, owned by Application global variable }
3990: Func FindFormByClass(FormClass: TFormClass): TForm;
3991: Func FindFormByClassName(FormClassName: Str): TForm;
3992: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
3993: having Tag property value, equaled to Tag parameter }
3994: Func FindByTag(WinControl: TWinControl; ComponentClass: TComponentClass; const Tag: Int): TComponent;
3995: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3996: Func ControlAtPos2(Parent: TWinControl; X, Y: Int): TControl;
3997: { RBTAG searches WinControl.Controls for checked RadioButton and returns Tag property value }
3998: Func RBTAG(Parent: TWinControl): Int;
3999: { AppMinimized returns True, if Application is minimized }
4000: Func AppMinimized: Bool;
4001: { MessageBox is Application.MessageBox with string (not PChar) parameters.
4002: if Caption parameter = '', it replaced with Application.Title }
4003: Func MessageBoxJ(const Msg: Str; Caption: Str; const Flags: Int): Int;
4004: Func MsgDlg2(const Msg, ACaption: Str; DlgType: TMsgDlgType;
4005: Buttons: TMsgDlgButtons; HelpContext: Int; Control: TWinControl): Int;
4006: Func MsgDlgDef(const Msg, ACaption: Str; DlgType: TMsgDlgType;
4007: Buttons: TMsgDlgButtons; DefButton: TMsgDlgBtn; HelpContext: Int; Control: TWinControl): Int;
4008: { Delay stop program execution to MSec msec }
4009: Proc Delay(MSec: Longword);

```

```

4010: Proc CenterHor(Parent: TControl; MinLeft: Int; Controls: array of TControl);
4011: Proc EnableControls(Control: TWinControl; const Enable: Bool);
4012: Proc EnableMenuItems(MenuItem: TMenuItem; const Tag: Int; const Enable: Bool);
4013: Proc ExpandWidth(Parent: TControl; MinWidth: Int; Controls: array of TControl);
4014: Func PanelBorder(Panel: TCustomPanel): Int;
4015: Func Pixels(Control: TControl; APixels: Int): Int;
4016: Proc SetChildPropOrd(Owner: TComponent; PropName: Str; Value: Longint);
4017: Proc Error(const Msg: Str);
4018: Proc ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: Str;
4019:   const HideSelColor: Bool; var PlainItem: Str; var Width: Int; CalcWidth: Bool);
4020:   {ex.Text parameter: 'Item1<b>bold</b><i>italic ITALIC<c:Red>red<c:Green>green<c:blue>blue</i>' }
4021: Func ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: Str;
4022:   const HideSelColor: Bool): Str;
4023: Func ItemHtWidth(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: Str;
4024:   const HideSelColor: Bool): Int;
4025: Func ItemHtPlain(const Text: Str): Str;
4026: { ClearList - clears list of TObject }
4027: Proc ClearList(List: TList);
4028: Proc MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
4029: Proc ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
4030: { RTTI support }
4031: Func GetPropType(Obj: TObject; const PropName: Str): TTypeKind;
4032: Func GetPropStr(Obj: TObject; const PropName: Str): Str;
4033: Func GetPropOrd(Obj: TObject; const PropName: Str): Int;
4034: Func GetPropMethod(Obj: TObject; const PropName: Str): TMethod;
4035: Proc PrepareIniSection(SS: TStrings);
4036: {following functions are not documented because they are don't work properly, so don't use them}
4037: {$IFDEF COMPILER2}
4038: Func CompareMem(P1, P2: Pointer; Length: Int): Bool; assembler;
4039: {$ENDIF}
4040:
4041: Proc SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
4042: begin
4043:   Proc BoxMoveSelectedItems( SrcList, DstList : TWinControl)
4044:   Proc BoxMoveAllItems( SrcList, DstList : TWinControl)
4045:   Proc BoxDragOver(List: TWinControl; Source: TObject; X, Y: Int; State: TDragState; var Accept: Bool; Sorted: Bool;
4046:   Proc BoxMoveFocusedItem( List : TWinControl; DstIndex : Int)
4047:   Proc BoxMoveSelected( List : TWinControl; Items : TStrings)
4048:   Proc BoxSetItem( List : TWinControl; Index : Int)
4049:   Func BoxGetFirstSelection( List : TWinControl) : Int
4050:   Func BoxCanDropItem( List: TWinControl; X, Y: Int; var DragIndex : Int): Boolean
4051: end;
4052:
4053: Proc SIRegister_JvCsvParse(CL: TPSPascalCompiler);
4054: begin
4055:   Const('MaxInitStrNum', 'LongInt' ( 9);
4056:   Func JvAnsiStrSplit( const InString : Ansistr; const SplitChar, QuoteChar: AnsiChar; var OutStrings : array
4057:   of Ansistr; MaxSplit : Int) : Int
4058:   Func JvStrSplit(const InString: Str; const SplitChar, QuoteChar: Char; var OutStrings: array of
4059:   string; MaxSplit: Int): Int
4060:   Func JvAnsiStrSplitStrings(const InStr: Ansistr; const SplitChar, QuoteChar: AnsiChar; OutStrs: TStrings): Int;
4061:   Func JvAnsiStrStrip( S : Ansistr) : Ansistr
4062:   Func JvStrStrip( S : Str) : Str
4063:   Func GetString( var Source : Ansistr; const Separator : Ansistr) : Ansistr
4064:   Func PadString( const S : Ansistr; Len : Int; PadChar : AnsiChar) : Ansistr
4065:   Func BuildPathName( const PathName, FileName : Ansistr) : Ansistr
4066:   Func StrEatWhiteSpace( const S : Str) : Str
4067:   Func HexToAscii( const S : Ansistr) : Ansistr
4068:   Func AsciiToHex( const S : Ansistr) : Ansistr
4069:   Func StripQuotes( const S1 : Ansistr) : Ansistr
4070:   Func ValidNumericLiteral( S1 : PAnsiChar) : Bool
4071:   Func ValidIntLiteral( S1 : PAnsiChar) : Bool
4072:   Func ValidHexLiteral( S1 : PAnsiChar) : Bool
4073:   Func HexPCharToInt( S1 : PAnsiChar) : Int
4074:   Func ValidStringLiteral( S1 : PAnsiChar) : Bool
4075:   Func StripPCharQuotes( S1 : PAnsiChar) : Ansistr
4076:   Func JvValidIdentifierAnsi( S1 : PAnsiChar) : Bool
4077:   Func JvValidIdentifier( S1 : Str) : Bool
4078:   Func JvEndChar( X : AnsiChar) : Bool
4079:   Proc JvGetToken( S1, S2 : PAnsiChar)
4080:   Func IsExpressionKeyword( S1 : PAnsiChar) : Bool
4081:   Func IsKeyword( S1 : PAnsiChar) : Bool
4082:   Func JvValidVarReference( S1 : PAnsiChar) : Bool
4083:   Func GetParenthesis( S1, S2 : PAnsiChar) : Bool
4084:   Proc JvGetVarReference( S1, S2, SIdx : PAnsiChar)
4085:   Proc JvEatWhitespaceChars( S1 : PAnsiChar);
4086:   Proc JvEatWhitespaceChars1( S1 : PWideChar);
4087:   Func GetTokenCount : Int
4088:   Proc ResetTokenCount
4089: end;
4090:
4091: Proc SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
4092: begin
4093:   SIRegister_TJvQueryParamsDialog(CL);
4094:   Func EditQueryParams(DataSet: TDataSet; List: TParams; AHelpContext: THelpContext): Bool
4095: end;
4096:
4097: ***** JvStrUtil / JvStrUtils; *****
4098: Func FindNotBlankCharPos(const S: Str): Int;

```

```

4097: Func AnsiChangeCase(const S:Str):Str;
4098: Func GetWordOnPos(const S:Str; const P: Int):Str;
4099: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
4100: Func Cmp(const S1, S2:Str):Bool;
4101: { Spaces returns string consists on N space chars }
4102: Func Spaces(const N: Int):Str;
4103: { HasChar returns True, if char, Ch, contains in string, S }
4104: Func HasChar(const Ch: Char; const S:Str):Bool;
4105: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
4106: { SubStr returns substring from string, S, separated with Separator string}
4107: Func SubStr(const S:Str; const Index: Int; const Separator:Str):Str;
4108: { SubStrEnd same to previous Func but Index numerated from the end of string }
4109: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
4110: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces with NewPattern }
4111: Func ReplaceString(S:Str; const OldPattern, NewPattern:Str):Str;
4112: Func CharInSet(const Ch: Char; const SetOfChar: TSetOfChar):Bool;
4113: { GetXYByPos is same to previous function, but returns X position in line too}
4114: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
4115: { AddSlash returns string with added slash char to Dir parameter, if needed }
4116: Func AddSlash2(const Dir: TFileName):Str;
4117: { AddPath returns FileName with Path, if FileName not contain any path }
4118: Func AddPath(const FileName, Path: TFileName): TFileName;
4119: { ExePath returns ExtractFilePath(ParamStr(0)) }
4120: Func ExePath: TFileName;
4121: Func LoadTextFile(const FileName: TFileName):Str;
4122: Proc SaveTextFile(const FileName: TFileName; const Source:Str);
4123: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4124: Func ConcatSep(const S, S2, Separator:Str):Str;
4125: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4126: Func FileEquMask(FileName, Mask: TFileName):Bool;
4127: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4128:   Masks must be separated with comma (',') }
4129: Func FileEquMasks(FileName, Masks: TFileName):Bool;
4130: Func StringEndsWith(const Str, SubStr:Str):Bool;
4131: Func ExtractFilePath2(const FileName:Str):Str;
4132: Func StrToOem(const AnsiStr:Str):Str;
4133: { StrToOem translates a string from the Windows character set into the OEM character set. }
4134: Func OemToAnsiStr(const OemStr:Str):Str;
4135: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4136: Func IsEmptyStr(const S:Str; const EmptyChars: TCharSet):Bool;
4137: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4138: Func ReplaceStr(const S, Srch, Replace:Str):Str;
4139: { Returns string with every occurrence of Srch string replaced with Replace string. }
4140: Func DelSpace(const S:Str):Str;
4141: { DelSpace return a string with all white spaces removed. }
4142: Func DelChars(const S:Str; Chr: Char):Str;
4143: { DelChars return a string with all Chr characters removed. }
4144: Func DelBSpace(const S:Str):Str;
4145: { DelBSpace trims leading spaces from the given string. }
4146: Func DelESpace(const S:Str):Str;
4147: { DelESpace trims trailing spaces from the given string. }
4148: Func DelRSpace(const S:Str):Str;
4149: { DelRSpace trims leading and trailing spaces from the given string. }
4150: Func DelSpace1(const S:Str):Str;
4151: { DelSpace1 return a string with all non-single white spaces removed. }
4152: Func Tab2Space(const S:Str; Numb: Byte):Str;
4153: { Tab2Space converts any tabulation chars in given string to the Numb spaces characters. }
4154: Func NPos(const C:Str; S:Str; N: Int): Int;
4155: { NPos searches for a N-th position of substring C in a given string. }
4156: Func MakeStr(C: Char; N: Int):Str;
4157: Func MS(C: Char; N: Int):Str;
4158: { MakeStr return a string of length N filled with character C. }
4159: Func AddChar(C: Char; const S:Str; N: Int):Str;
4160: { AddChar return a string left-padded to length N with characters C. }
4161: Func AddCharR(C: Char; const S:Str; N: Int):Str;
4162: { AddCharR return a string right-padded to length N with characters C. }
4163: Func LeftStr(const S:Str; N: Int):Str;
4164: { LeftStr return a string right-padded to length N with blanks. }
4165: Func RightStr(const S:Str; N: Int):Str;
4166: { RightStr return a string left-padded to length N with blanks. }
4167: Func CenterStr(const S:Str; Len: Int):Str;
4168: { CenterStr centers the characters in the string based upon the Len specified. }
4169: Func CompStr(const S1, S2:Str): Int;
4170: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1<S2, 0 if S1=S2, or 1 if S1>S2.}
4171: Func CompText(const S1, S2:Str): Int;
4172: {CompText compares S1 to S2, without case-sensitivity. return value is the same as for CompStr.}
4173: Func Copy2Symb(const S:Str; Symb: Char):Str;
4174: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4175: Func Copy2SymbDel(var S:Str; Symb: Char):Str;
4176: {Copy2SymbDel returns substr of str S from first character Symb and removes substring from S.}
4177: Func Copy2Space(const S:Str):Str;
4178: { Copy2Symb returns a substring of a string S from beginning to first white space. }
4179: Func Copy2SpaceDel(var S:Str):Str;
4180: { Copy2SpaceDel returns a substring of a string S from beginning to first
4181:   white space and removes this substring from S. }
4182: Func AnsiProperCase(const S:Str; const WordDelims: TCharSet):Str;
4183: { Returns string, with the first letter of each word in uppercase,
4184:   all other letters in lowercase. Words are delimited by WordDelims. }
4185: Func WordCount(const S:Str; const WordDelims: TCharSet): Int;

```

```

4186: { WordCount given a set of word delimiters, returns number of words in S. }
4187: Func WordPosition(const N: Int; const S:Str; const WordDelims: TCharSet): Int;
4188: { Given a set of word delimiters, returns start position of N'th word in S. }
4189: Func ExtractWord(N: Int; const S:Str; const WordDelims: TCharSet):Str;
4190: Func ExtractWordPos(N: Int; const S:Str; const WordDelims: TCharSet; var Pos: Int):Str;
4191: Func ExtractDelimited(N: Int; const S:Str; const Delims: TCharSet):Str;
4192: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4193:   delimiters, return the N'th word in S. }
4194: Func ExtractSubstr(const S:Str; var Pos: Int; const Delims: TCharSet):Str;
4195: { ExtractSubstr given set of word delimiters, returns the substring from S, that started from position Pos. }
4196: Func IsWordPresent(const W, S:Str; const WordDelims: TCharSet):Bool;
4197: { IsWordPresent given set of word delimiters, returns True if word W is present in string S. }
4198: Func QuotedString(const S:Str; Quote: Char):Str;
4199: { QuotedString returns given string as a quoted string, using the provided Quote character. }
4200: Func ExtractQuotedString(const S:Str; Quote: Char):Str;
4201: { ExtractQuotedString removes the Quote characters from beginning and end of a quoted string,
4202:   and reduces pairs of Quote characters within the quoted string to a single character. }
4203: Func FindPart(const HelpWilds, InputStr:Str): Int;
4204: { FindPart compares a string with '?' and another, returns position of HelpWilds in InputStr. }
4205: Func IsWild(InputStr, Wilds:Str; IgnoreCase:Bool):Bool;
4206: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4207: Func XorString(const Key, Src: ShortString): ShortString;
4208: Func XorEncode(const Key, Source:Str):Str;
4209: Func XorDecode(const Key, Source:Str):Str;
4210: { ** Command line routines ** }
4211: {$IFDEF COMPILER4_UP}
4212: Func FindCmdLineSwitch(const Switch:Str; SwitchChars TCharSet; IgnoreCase: Bool):Bool;
4213: {$ENDIF}
4214: Func GetCmdLineArg(const Switch:Str; SwitchChars: TCharSet):Str;
4215: { ** Numeric string handling routines ** }
4216: Func Numb2USA(const S:Str):Str;
4217: { Numb2USA converts numeric string S to USA-format. }
4218: Func Dec2Hex(N: Longint; A: Byte):Str;
4219: Func D2H(N: Longint; A: Byte):Str;
4220: { Dec2Hex converts the given value to a hexadecimal string representation
4221:   with the minimum number of digits (A) specified. }
4222: Func Hex2Dec(const S:Str): Longint;
4223: Func H2D(const S:Str): Longint;
4224: { Hex2Dec converts the given hexadecimal string to the corresponding Int value. }
4225: Func Dec2Numb(N: Longint; A, B: Byte):Str;
4226: { Dec2Numb converts the given value to a string representation with the
4227:   base equal to B and with the minimum number of digits (A) specified. }
4228: Func Numb2Dec(S:Str; B: Byte): Longint;
4229: { Numb2Dec converts given B-based numeric string to the corresponding Int value. }
4230: Func IntToBin(Value: Longint; Digits, Spaces: Int):Str;
4231: { IntToBin converts given value to bin str representation with min number of digits specified. }
4232: Func IntToRoman(Value: Longint):Str;
4233: { IntToRoman converts the given value to a roman numeric string representation. }
4234: Func RomanToInt(const S:Str): Longint;
4235: { RomanToInt converts the given string to an Int value. If the string
4236:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4237: Proc I64ToCardinals(I: Int64; var LowPart, HighPart:Card);
4238: Proc CardinalsToI64(var I: Int64; const LowPart, HighPart:Card);
4239: ***** JvFileUtil; *****
4240: Proc CopyFileJ(const FileName, DestName:Str; ProgressControl: TControl);
4241: Proc CopyFileEx(const FileName, DestName:Str; OverwriteReadOnly, ShellDialog:Bool; ProgressControl:TControl);
4242: Proc MoveFile(const FileName, DestName: TFileName);
4243: Proc MoveFileEx(const FileName, DestName: TFileName; ShellDialog:Bool);
4244: {$IFDEF COMPILER4_UP}
4245: Func GetFileSize(const FileName:Str): Int64;
4246: {$ELSE}
4247: Func GetFileSize(const FileName:Str): Longint;
4248: {$ENDIF}
4249: Func FileDateTime(const FileName:Str): TDateTime;
4250: Func HasAttr(const FileName:Str; Attr: Int):Bool;
4251: Func DeleteFiles(const FileMask:Str):Bool;
4252: Func DeleteFilesEx(const FileMasks: array of string):Bool;
4253: Func ClearDir(const Path:Str; Delete:Bool):Bool;
4254: Func NormalDir(const DirName:Str):Str;
4255: Func RemoveBackSlash(const DirName:Str):Str;
4256: Func ValidFileName(const FileName:Str):Bool;
4257: Func DirExists(Name:Str):Bool;
4258: Proc ForceDirectories(Dir:Str);
4259: Func FileLock(Handle: Int; Offset, LockSize: Longint): Int;
4260: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4261: {$IFDEF COMPILER4_UP}
4262: Func FileLock(Handle: Int; Offset, LockSize: Int64): Int; overload;
4263: {$ENDIF}
4264: Func FileUnlock(Handle: Int; Offset, LockSize: Longint): Int;
4265: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4266: {$IFDEF COMPILER4_UP}
4267: Func FileUnlock(Handle: Int; Offset, LockSize: Int64): Int; overload;
4268: {$ENDIF}
4269: Func GetTempDir:Str;
4270: Func GetWindowsDir:Str;
4271: Func GetSystemDir:Str;
4272: Func BrowseDirectory(var AFolderName:Str; const DlgText:Str; AHelpContext:THelpContext):Boolean;
4273: {$IFDEF WIN32}
4274: Func BrowseComputer(var ComputerName:Str; const DlgText:Str; AHelpContext:THelpContext):Bool;

```



```

4275: Func ShortToLongFileName(const ShortName:Str):Str;
4276: Func ShortToLongPath(const ShortName:Str):Str;
4277: Func LongToShortFileName(const LongName:Str):Str;
4278: Func LongToShortPath(const LongName:Str):Str;
4279: Proc CreateFileLink(const FileName, DisplayName:Str; Folder: Int);
4280: Proc DeleteFileLink(const DisplayName:Str; Folder: Int);
4281: {SENDIF WIN32}
4282: {IFDEF COMPILER3_UP}
4283: Func IsPathDelimiter(const S:Str; Index: Int):Bool;
4284: {SENDIF}
4285: Func CreateCalculatorForm(AOwner:TComponent;AHelpContext:THelpContext):TJvCalculatorForm
4286: Func CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode ) : TWinControl
4287: Func CreatePopupCalculator( AOwner : TComponent ) : TWinControl
4288: Proc SetupPopupCalculator(PopupCalc:TWinControl;APrecision Byte; ABeepOnError: Bool)
4289:
4290: //*****Proc SIRegister_VarHlpr(CL: TPSPascalCompiler);
4291: Proc VariantClear( var V : Variant)
4292: Proc VariantArrayRedim( var V : Variant; High : Int)
4293: Proc VariantCast( const src : Variant; var dst : Variant; vt : Int)
4294: Proc VariantCpy(const src:Variant;var dst:Variant)
4295: Proc VariantAdd(const src:Variant;var dst:Variant)
4296: Proc VariantSub(const src:Variant;var dst:Variant)
4297: Proc VariantMul(const src:Variant;var dst:Variant)
4298: Proc VariantDiv(const src:Variant;var dst:Variant)
4299: Proc VariantMod(const src:Variant;var dst:Variant)
4300: Proc VariantAnd(const src:Variant;var dst:Variant)
4301: Proc VariantOr( const src:Variant;var dst:Variant)
4302: Proc VariantXor(const src:Variant;var dst:Variant)
4303: Proc VariantShl(const src:Variant;var dst:Variant)
4304: Proc VariantShr(const src:Variant;var dst:Variant)
4305: Func VariantAdd2(const V1:Variant; const V2:Variant):Variant
4306: Func VariantSub2(const V1:Variant; const V2:Variant):Variant
4307: Func VariantMul2(const V1:Variant; const V2:Variant):Variant
4308: Func VariantDiv2(const V1:Variant; const V2:Variant):Variant
4309: Func VariantMod2(const V1:Variant; const V2:Variant):Variant
4310: Func VariantAnd2(const V1:Variant; const V2:Variant):Variant
4311: Func VariantOr2( const V1:Variant; const V2:Variant):Variant
4312: Func VariantXor2(const V1: Variant; const V2: Variant): Variant
4313: Func VariantShl2(const V1: Variant; const V2: Variant): Variant
4314: Func VariantShr2(const V1: Variant; const V2: Variant): Variant
4315: Func VariantNot(const V1: Variant) : Variant
4316: Func VariantNeg(const V1: Variant) : Variant
4317: Func VariantGetElement( const V : Variant; i1 : Int) : Variant;
4318: Func VariantGetElement1(const V : Variant; i1, i2 : Int) : Variant;
4319: Func VariantGetElement2(const V : Variant; i1, i2, i3 : Int) : Variant;
4320: Func VariantGetElement3(const V : Variant; i1, i2, i3, i4 : Int) : Variant;
4321: Func VariantGetElement4(const V : Variant; i1, i2, i3, i4, i5 : Int) : Variant;
4322: Proc VariantPutElement(var V : Variant; const data : Variant; i1 : Int);
4323: Proc VariantPutElement1(var V : Variant; const data : Variant; i1, i2 : Int);
4324: Proc VariantPutElement2(var V : Variant; const data : Variant; i1, i2, i3 : Int);
4325: Proc VariantPutElement3(var V : Variant; const data : Variant; i1, i2, i3, i4 : Int);
4326: Proc VariantPutElement4(var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : Int);
4327: end;
4328:
4329: *****unit uPSI_JvgUtils;*****
4330: Func IsEven(I: Int):Bool;
4331: Func InchesToPixels(DC: HDC; Value: Single; IsHorizontal:Bool): Int;
4332: Func CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal:Bool): Int;
4333: Proc SwapInt(var I1, I2: Int);
4334: Func Spaces(Count: Int):Str;
4335: Func DupStr(const Str:Str; Count: Int):Str;
4336: Func DupChar(C: Char; Count: Int):Str;
4337: Proc Msg(const AMsg:Str);
4338: Func RectW(R: TRect): Int;
4339: Func RectH(R: TRect): Int;
4340: Func IncColor(AColor: Longint; AOffset: Byte): Longint;
4341: Func DecColor(AColor: Longint; AOffset: Byte): Longint;
4342: Func IsItAFilledBitmap(Bmp: TBitmap):Bool;
4343: Proc DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text:Str;
4344: HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4345: Proc DrawTextInRect(DC: HDC; R: TRect; const Text:Str; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4346: Proc ExtTextOutExt(DC: HDC; X, Y: Int; R: TRect; const Text:Str;
4347: Style: TglTextStyle; ALineated, ASuppress3D:Bool; FontColor, DelinColor, HighlightColor, ShadowColor:
TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4348: Proc DrawBox(DC: HDC; var R: TRect; Style: TglBoxStyle; BackgrColor: Longint; ATransparent: Bool);
4349: Func DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4350: BevelInner, BevelOuter: TPanelBevel; Bold: Boolean; BackgrColor: Longint; ATransparent: Bool): TRect;
4351: Proc GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient; PenStyle, PenWidth: Int);
4352: Proc ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4353: Proc DrawBitmapExt(DC: HDC; { DC - background & result}
4354: SourceBitmap: TBitmap; R: TRect; X, Y: Int; {...X,Y_in_rect!
4355: BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4356: ATransparent: Bool; TransparentColor: TColor; DisabledMaskColor: TColor);
4357: Proc CreateBitmapExt(DC: HDC; { DC - background & result}
4358: SourceBitmap: TBitmap; R: TRect; X, Y: Int; {...X,Y_in_rect!
4359: BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4360: ATransparent: Bool; TransparentColor: TColor; DisabledMaskColor: TColor);
4361: Proc BringParentWindowToTop(Wnd: TWinControl);
4362: Func GetParentForm(Control: TControl): TForm;

```

```

4363: Proc GetWindowImageFrom(Control:TWInControl;X,Y:Int;ADrawSelf,ADrawChildWindows:Bool;DC:HDC)
4364: Proc GetWindowImage(Control:TWInControl;ADrawSelf,ADrawChildWindows:Bool;DC:HDC);
4365: Proc GetParentImageRect(Control:TWInControl;Rect:TRect;DC:HDC);
4366: Func CreateRotatedFont(F:TFont;Escapement:Int):HFONT;
4367: Func FindMainWindow(const AWndClass,AWndTitle:Str):THandle;
4368: Proc CalcShadowAndHighlightColors(BaseColor:TColor;Colors:TJvlgLabelColors);
4369: Func CalcMathString(AExpression:Str):Single;
4370: Func IIF(AExpression:Bool;IfTrue,IfFalse:Variant):Variant; overload;
4371: Func IIF(AExpression:Bool;const IfTrue,IfFalse:Str):Str; overload;
4372: Func GetTransparentColor(Bitmap:TBitmap;AutoTrColor:TglAutoTransparentColor):TColor;
4373: Proc TypeStringOnKeyboard(const S:Str);
4374: Func NextStringGridCell(Grid:TStringGrid):Bool;
4375: Proc DrawTextExtAligned(Canvas:TCanvas;const Text:Str;R:TRect;Alignment:TglAlignment;WordWrap:Bool;
4376: Proc LoadComponentFromTextFile(Component:TComponent;const FileName:Str);
4377: Proc SaveComponentToTextFile(Component:TComponent;const FileName:Str);
4378: Func ComponentToString(Component:TComponent):Str;
4379: Proc StringToComponent(Component:TComponent;const Value:Str);
4380: Func PlayWaveResource(const ResName:Str):Bool;
4381: Func UserName:Str;
4382: Func ComputerName:Str;
4383: Func CreateIniFileName:Str;
4384: Func ExpandString(const Str:Str;Len:Int):Str;
4385: Func Transliterate(const Str:Str;RusToLat:Bool):Str;
4386: Func IsSmallFonts:Bool;
4387: Func SystemColorDepth: Int;
4388: Func GetFileTypeJ(const FileName:Str):TglFileType;
4389: Func GetFileType(hFile:THandle):DWORD;
4390: Func FindControlAtPt(Control:TWInControl;Pt:TPoint;MinClass:TClass):TControl;
4391: Func StrPosExt(const Str1,Str2:PChar;Str2Len:DWORD):PChar;
4392:
4393: { *****Utility routines of unit classes}
4394: Func LineStart(Buffer,BufPos:PChar):PChar
4395: Func ExtractStrings(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strings:TStrings):Int
4396: Func TestStreamFormat(Stream:TStream):TStreamOriginalFormat
4397: Proc RegisterClass(AClass:TPersistentClass)
4398: Proc RegisterClasses(AClasses:array of TPersistentClass)
4399: Proc RegisterClassAlias(AClass:TPersistentClass;const Alias:Str)
4400: Proc UnRegisterClass(AClass:TPersistentClass)
4401: Proc UnRegisterClasses(AClasses:array of TPersistentClass)
4402: Proc UnRegisterModuleClasses(Module:HMODULE)
4403: Func FindGlobalComponent(const Name:Str):TComponent
4404: Func IsUniqueGlobalComponentName(const Name:Str):Bool
4405: Func InitInheritedComponent(Instance:TComponent;RootAncestor:TClass):Bool
4406: Func InitComponentRes(const ResName:Str;Instance:TComponent):Bool
4407: Func ReadComponentRes(const ResName:Str;Instance:TComponent):TComponent
4408: Func ReadComponentResEx(HInstance:THandle;const ResName:Str):TComponent
4409: Func ReadComponentResFile(const FileName:Str;Instance:TComponent):TComponent
4410: Proc WriteComponentResFile(const FileName:Str;Instance:TComponent)
4411: Proc GlobalFixupReferences
4412: Proc GetFixupReferenceNames(Root:TComponent;Names:TStrings)
4413: Proc GetFixupInstanceNames(Root:TComponent;const ReferenceRootName:string;Names:TStrings)
4414: Proc RedirectFixupReferences(Root:TComponent;const OldRootName,NewRootName:Str)
4415: Proc RemoveFixupReferences(Root:TComponent;const RootName:Str)
4416: Proc RemoveFixups(Instance:TPersistent)
4417: Func FindNestedComponent(Root:TComponent;const NamePath:Str):TComponent
4418: Proc BeginGlobalLoading
4419: Proc NotifyGlobalLoading
4420: Proc EndGlobalLoading
4421: Func GetUltimateOwner1(ACollection:TCollection):TPersistent;
4422: Func GetUltimateOwner(APersistent:TPersistent):TPersistent;
4423: // AddTypes('TWndMethod','Proc(var Message:TMessage)
4424: //Func MakeObjectInstance(Method:TWndMethod):Pointer
4425: Proc FreeObjectInstance(ObjectInstance:Pointer)
4426: //Func AllocateHWnd(Method:TWndMethod):HWN
4427: Proc DeallocateHWnd(Wnd:HWN)
4428: Func AncestorIsValid(Ancursor:TPersistent;Root,RootAncestor:TComponent):Bool
4429: *****unit uPSI_SqlTimSt and DB;*****
4430: Proc VarSQLTimeStampCreate4(var aDest:Variant;const ASQLTimeStamp:TSQLTimeStamp);
4431: Func VarSQLTimeStampCreate3:Variant;
4432: Func VarSQLTimeStampCreate2(const AValue:Str):Variant;
4433: Func VarSQLTimeStampCreate1(const AValue:TDateTime):Variant;
4434: Func VarSQLTimeStampCreate(const ASQLTimeStamp:TSQLTimeStamp):Variant;
4435: Func VarSQLTimeStamp:TVarType
4436: Func VarIsSQLTimeStamp(const aValue:Variant):Bool;
4437: Func LocalToUTC(var TZInfo:TTimeZone;var Value:TSQLTimeStamp):TSQLTimeStamp //beta
4438: Func UTCToLocal(var TZInfo:TTimeZone;var Value:TSQLTimeStamp):TSQLTimeStamp //beta
4439: Func VarToSQLTimeStamp(const aValue:Variant):TSQLTimeStamp
4440: Func SQLTimeStampToStr(const Format:Str;DateTime:TSQLTimeStamp):Str
4441: Func SQLDayOfWeek(const DateTime:TSQLTimeStamp):Int
4442: Func DateTimeToSQLTimeStamp(const DateTime:TDateTime):TSQLTimeStamp
4443: Func SQLTimeStampToDate(const DateTime:TSQLTimeStamp):TDateTime
4444: Func TryStrToSQLTimeStamp(const S:Str;var TimeStamp:TSQLTimeStamp):Bool
4445: Func StrToSQLTimeStamp(const S:Str):TSQLTimeStamp
4446: Proc CheckSqlTimeStamp(const ASQLTimeStamp:TSQLTimeStamp)
4447: Func ExtractFieldName(const Fields:Str;var Pos:Int):Str;
4448: Func ExtractFieldName(const Fields:WideString;var Pos:Int):WideString;
4449: //Proc RegisterFields(const FieldClasses:array of TFieldClass)
4450: Proc DatabaseError(const Message:WideString;Component:TComponent)
4451: Proc DatabaseErrorFmt(const Message:WideString;const Args:array of const;Component:TComponent)

```

```

4452: Proc DisposeMem( var Buffer, Size : Int)
4453: Func BuffersEqual( Buf1, Buf2 : Pointer; Size : Int) :Bool
4454: Func GetFieldProperty(DataSet:TDataSet;Control:TComponent;const FieldName:WideStr):TField
4455: Func VarTypeToDataType( VarType : Int) : TFieldType
4456: *****unit JvStrings;*****
4457: {template functions}
4458: Func ReplaceFirst(const SourceStr, FindStr, ReplaceStr:Str):Str;
4459: Func ReplaceLast(const SourceStr, FindStr, ReplaceStr:Str):Str;
4460: Func InsertLastBlock(var SourceStr:Str; BlockStr:Str):Bool;
4461: Func RemoveMasterBlocks(const SourceStr:Str):Str;
4462: Func RemoveFields(const SourceStr:Str):Str;
4463: {http functions}
4464: Func URLEncode(const Value:Ansistr):Ansistr; // Converts string To A URLEncoded string
4465: Func URLDecode(const Value:Ansistr):Ansistr; // Converts string From A URLEncoded string
4466: {set functions}
4467: Proc SplitSet(AText:Str; AList: TStringList);
4468: Func JoinSet(AList: TStringList):Str;
4469: Func FirstOfSet(const AText:Str):Str;
4470: Func LastOfSet(const AText:Str):Str;
4471: Func CountOfSet(const AText:Str): Int;
4472: Func SetRotateRight(const AText:Str):Str;
4473: Func SetRotateLeft(const AText:Str):Str;
4474: Func SetPick(const AText:Str; AIndex: Int):Str;
4475: Func SetSort(const AText:Str):Str;
4476: Func SetUnion(const Set1, Set2:Str):Str;
4477: Func SetIntersect(const Set1, Set2:Str):Str;
4478: Func SetExclude(const Set1, Set2:Str):Str;
4479: {replace any <, > etc by &lt; &gt;}
4480: Func XMLSafe(const AText:Str):Str;
4481: {simple hash, Result can be used in Encrypt}
4482: Func Hash(const AText:Str): Int;
4483: { Base64 encode and decode a string }
4484: Func B64Encode(const S: Ansistr): Ansistr;
4485: Func B64Decode(const S: Ansistr): Ansistr;
4486: {Basic encryption from a Borland Example}
4487: Func Encrypt(const InString: Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4488: Func Decrypt(const InString: Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4489: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4490: Func EncryptB64(const InString:Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4491: Func DecryptB64(const InString:Ansistr; StartKey, MultKey, AddKey: Int): Ansistr;
4492: Proc CSVToTags(Src, Dst: TStringList);
4493: // converts a csv list to a tagged string list
4494: Proc TagsToCSV(Src, Dst: TStringList);
4495: // converts a tagged string list to a csv list
4496: // only fieldnames from the first record are scanned ib the other records
4497: Proc ListSelect(Src, Dst: TStringList; const AKey, AValue:Str);
4498: {selects akey=avalue from Src and returns recordset in Dst}
4499: Proc ListFilter(Src: TStringList; const AKey, AValue:Str);
4500: {filters Src for akey=avalue}
4501: Proc ListOrderBy(Src: TStringList; const AKey:Str; Numeric: Bool);
4502: {orders a tagged Src list by akey}
4503: Func PosStr(const FindString, SourceString:Str;
4504:   StartPos: Int = 1): Int;
4505: { PosStr searches the first occurrence of a substring FindString in a string
4506:   given by SourceString with case sensitivity (upper and lower case characters
4507:   are differed). This Func returns the index value of the first character
4508:   of a specified substring from which it occurs in a given string starting with
4509:   StartPos character index. If a specified substring is not found Q_PosStr
4510:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings from www.torry.ru). }
4511: Func PosStrLast(const FindString, SourceString:Str): Int;
4512: {finds the last occurrence}
4513: Func LastPosChar(const FindChar: Char; SourceString:Str): Int;
4514: Func PosText(const FindString, SourceString:Str; StartPos: Int = 1): Int;
4515: { PosText searches the first occurrence of a substring FindString in a string
4516:   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
4517: Func PosTextLast(const FindString, SourceString:Str): Int;
4518: {finds last occurrence}
4519: Func NameValuesToXML(const AText:Str):Str;
4520: {$IFDEF MSWINDOWS}
4521: Proc LoadResourceFile(AFile:Str; MemStream: TMemoryStream);
4522: {$ENDIF MSWINDOWS}
4523: Proc DirFiles(const ADir, AMask:Str; AFileList: TStringList);
4524: Proc RecurseDirFiles(const ADir:Str; var AFileList: TStringList);
4525: Proc RecurseDirProgs(const ADir:Str; var AFileList: TStringList);
4526: Proc SaveString(const AFile, AText:Str);
4527: Proc SaveStringasFile( const AFile, AText :Str)
4528: Func LoadStringJ(const AFile:Str):Str;
4529: Func LoadStringofFile( const AFile :Str ) :Str
4530: Proc SaveStringtoFile( const AFile, AText :Str)
4531: Func LoadStringfromFile( const AFile :Str) :Str
4532: Func HexToColor(const AText:Str): TColor;
4533: Func UppercaseHTMLTags(const AText:Str):Str;
4534: Func LowercaseHTMLTags(const AText:Str):Str;
4535: Proc GetHTMLAnchors(const AFile:Str; AList: TStringList);
4536: Func RelativePath(const ASrc, ADst:Str):Str;
4537: Func GetToken(var Start: Int; const SourceText:Str):Str;
4538: Func PosNonSpace(Start: Int; const SourceText:Str): Int;

```

```

4539: Func PosEscaped(Start: Int; const SourceText, FindText:Str; EscapeChar: Char): Int;
4540: Func DeleteEscaped(const SourceText:Str; EscapeChar: Char):Str;
4541: Func BeginOfAttribute(Start: Int; const SourceText:Str): Int;
4542: // parses the beginning of an attribute: space + alpha character
4543: Func ParseAttribute(var Start: Int; const SourceText:Str; var AName, AValue:Str):Bool;
4544: //parses a name="value" attrib from Start; returns 0 when not found or else position behind attribute
4545: Proc ParseAttributes(const SourceText:Str; Attributes: TStrings);
4546: // parses all name=value attributes to the attributes TStringList
4547: Func HasStrValue(const AText, AName:Str; var AValue:Str):Bool;
4548: // checks if a name="value" pair exists and returns any value
4549: Func GetStrValue(const AText, AName, ADefault:Str):Str;
4550: // retrieves string value from a line like:
4551: // name="jan verhoeven" email="janl dott verhoeven att wxs dott nl"
4552: // returns ADefault when not found
4553: Func GetHTMLColorValue(const AText, AName:Str; ADefault: TColor):TColor;
4554: // same for a color
4555: Func GetIntValue(const AText, AName:Str; ADefault: Int): Int;
4556: // same for an Int
4557: Func GetFloatValue(const AText, AName:Str; ADefault: Extended):Extended;
4558: // same for a float
4559: Func GetBoolValue(const AText, AName:Str):Bool;
4560: // same for Boolean but without default
4561: Func GetValue(const AText, AName:Str):Str;
4562: //retrieves string value from a line like: name="jan verhoeven" email="janl verhoeven att wxs dott nl"
4563: Proc SetValue(var AText:Str; const AName, AValue:Str);
4564: // sets a string value in a line
4565: Proc DeleteValue(var AText:Str; const AName:Str);
4566: // deletes a AName="value" pair from AText
4567: Proc GetNames(AText:Str; AList: TStringList);
4568: // get a list of names from a string with name="value" pairs
4569: Func GetHTMLColor(AColor: TColor):Str;
4570: // converts a color value to the HTML hex value
4571: Func BackPosStr(Start: Int; const FindString, SourceString:Str): Int;
4572: // finds a string backward case sensitive
4573: Func BackPosText(Start: Int; const FindString, SourceString:Str): Int;
4574: // finds a string backward case insensitive
4575: Func PosRangeStr(Start: Int; const HeadString, TailString, SourceString:Str;
4576:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4577: // finds a text range, e.g. <TD>...</TD> case sensitive
4578: Func PosRangeText(Start: Int; const HeadString, TailString, SourceString:Str;
4579:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4580: // finds a text range, e.g. <TD>...</td> case insensitive
4581: Func BackPosRangeStr(Start: Int; const HeadString, TailString, SourceString:Str;
4582:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4583: // finds a text range backward, e.g. <TD>...</TD> case sensitive
4584: Func BackPosRangeText(Start: Int; const HeadString, TailString, SourceString:Str;
4585:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4586: // finds a text range backward, e.g. <TD>...</td> case insensitive
4587: Func PosTag(Start: Int; SourceString:Str; var RangeBegin: Int;
4588:   var RangeEnd: Int):Bool;
4589: // finds a HTML or XML tag: <....>
4590: Func InnerTag(Start: Int; const HeadString, TailString, SourceString:Str;
4591:   var RangeBegin: Int; var RangeEnd: Int):Bool;
4592: // finds the innertext between opening and closing tags
4593: Func Easter(NYear: Int): TDateTime;
4594: // returns the easter date of a year.
4595: Func GetWeekNumber(Today: TDateTime):Str;
4596: //gets a datecode. Returns year and weeknumber in format: YYWW
4597: Func ParseNumber(const S:Str): Int;
4598: // parse number returns the last position, starting from 1
4599: Func ParseDate(const S:Str): Int;
4600: // parse a SQL style data string from positions 1, // starts and ends with #
4601:
4602: *****unit JvJCLUtils;*****
4603: PathDelim, 'String').SetString( '\';
4604: DriveDelim, 'String').SetString( ';;
4605: PathSep, 'String').SetString( ';';
4606: AllFilesMask, 'String').SetString( '*.*';
4607: PathDelim, 'String').SetString( '/';
4608: AllFilesMask, 'String').SetString( '*';
4609: NullHandle, 'LongInt').SetInt( 0);
4610: USDecimalSeparator, 'String').SetString( '.;
4611:   ClassN(CL.FindClass('TOBJECT'), 'EJvConvertError;
4612:   TypeS('TFileTime', 'Int;
4613: // TypeS('TFormatSettings', 'record DecimalSeparator : Char; end;
4614: Func SendRectMessage(Handle: THandle; Msg : Int; wParam: longint; var R : TRect): Int;
4615: //Func SendStructMessage(Handle:THandle;Msg: Int; wParam:WPARAM; var Data): Int;
4616: Func ReadCharsFromStream(Stream: TStream; var Buf: array of PChar; BufSize: Int): Int;
4617: Func WriteStringToStream(Stream: TStream; const Buf: Ansistr; BufSize: Int): Int;
4618: Func UTF8ToString(const S : UTF8String) : Str;
4619: //ConstantN('DefaultDateOrder', '').SetString('doDMY;
4620: ConstantN('CenturyOffset', 'Byte').SetUInt( 60);
4621:
4622: Func VarIsInt(Value: Variant):Bool;
4623: // VarIsInt returns VarIsOrdinal-[varBoolean]
4624: {PosIdx returns index of first appearance of SubStr in Str. search starts at index "Index".}
4625: Func PosIdx(const SubStr, S:Str; Index: Int = 0): Int;
4626: Func PosIdxW(const SubStr, S: WideString; Index: Int = 0): Int;
4627: Func PosLastCharIdx(Ch: Char; const S:Str; Index: Int = 0): Int;

```

```

4628: { GetWordOnPos returns Word from string, S, on the cursor position, P }
4629: Func GetWordOnPos(const S:Str; const P: Int):Str;
4630: Func GetWordOnPosW(const S: WideString; const P: Int): WideString;
4631: Func GetWordOnPos2(const S:Str; P: Int; var iBeg, iEnd: Int):Str;
4632: Func GetWordOnPos2W(const S: WideString; P: Int; var iBeg, iEnd: Int): WideString;
4633: { GetWordOnPosEx working like GetWordOnPos function, but
4634:   also returns Word position in iBeg, iEnd variables }
4635: Func GetWordOnPosEx(const S:Str; const P: Int; var iBeg, iEnd: Int):Str;
4636: Func GetWordOnPosExW(const S: WideString; const P: Int; var iBeg, iEnd: Int): WideString;
4637: Func GetNextWordPosEx(const Text:Str; StartIndex: Int; var iBeg, iEnd: Int):Str;
4638: Func GetNextWordPosExW(const Text:WideString;StartIndex:Int; var iBeg,iEnd:Int):WideString;
4639: Proc GetEndPosCaret(const Text:Str; CaretX, CaretY: Int; var X, Y: Int);
4640: { GetEndPosCaret returns the caret position of the last char. For the position
4641:   after the last char of Text you must add 1 to the returned X value. }
4642: Proc GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Int;var X,Y:Int);
4643: { GetEndPosCaret returns the caret position of the last char. For the position
4644:   after the last char of Text you must add 1 to the returned X value. }
4645: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4646: Func SubStrBySeparator(const S:str;const Index:Int;const Separator:str;StartIndex:Int=1):str;
4647: Func SubStrBySeparatorW(const S:WideString;const Index:Int;const
  Separator:WideString;StartIndex:Int:WideString;
4648: { SubStrEnd same to previous Func but Index numerated from the end of string }
4649: Func SubStrEnd(const S:Str; const Index: Int; const Separator:Str):Str;
4650: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4651: Func SubWord(P: PChar; var P2: PChar):Str;
4652: Func CurrencyByWord(Value: Currency):Str;
4653: { GetLineByPos returns Line number,there symbol Pos is pointed.Lines separated with #13 symbol}
4654: Func GetLineByPos(const S:Str; const Pos: Int): Int;
4655: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4656: Proc GetXYByPos(const S:Str; const Pos: Int; var X, Y: Int);
4657: Proc GetXYByPosW(const S: WideString; const Pos: Int; var X, Y: Int);
4658: { ReplaceString searches for all substrings, OldPattern,
4659:   in a string, S, and replaces them with NewPattern }
4660: Func ReplaceString(S:Str;const OldPattern,NewPattern:str;StartIndex:Int= 1):str;
4661: Func ReplaceStringW(S:WideString;const OldPattern,NewPattern:WideString;StartIndex:Int=1):WideString;
4662: { ConcatSep concatenate S1 and S2 strings with Separator. if S='' then separator not included}
4663: Func ConcatSep(const S1, S2, Separator:Str):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4664: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
4665: Func ConcatLeftSep(const S1,S2, Separator:str):str;{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF SUPPORTS_INLINE}
4666: { Next 4 Func for russian chars transliterating.
4667:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4668: Proc Dos2Win(var S: Ansistr);
4669: Proc Win2Dos(var S: Ansistr);
4670: Func Dos2WinRes(const S: Ansistr): Ansistr; inline; {$ENDIF SUPPORTS_INLINE}
4671: Func Win2DosRes(const S: Ansistr): Ansistr; inline; {$ENDIF SUPPORTS_INLINE}
4672: Func Win2Koi(const S: Ansistr): Ansistr;
4673: { FillString fills the string Buffer with Count Chars }
4674: Proc FillString(var Buffer:Str; Count: Int; const Value: Char); overload;
4675: Proc FillString1(var Buffer:Str; StartIndex,Count:Int; const Value: Char);overload;
4676: { MoveString copies Count Chars from Source to Dest }
4677: Proc MoveString(const Source:Str; var Dest:Str; Count: Int); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE} overload;
4678: Proc MoveString1(const Source:Str; SrcStartIdx: Int; var Dest:Str;
  DstStartIdx: Int;Count:Int);{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}overload;
4680: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4681: Proc FillWideChar(var Buffer; Count: Int; const Value: WideChar);
4682: { MoveWideChar copies Count WideChars from Source to Dest }
4683: Proc MoveWideChar(const Source;var Dest;Count:Int);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF SUPPORTS_INLINE}
4684: { FillNativeChar fills Buffer with Count NativeChars }
4685: Proc FillNativeChar(var Buffer; Count: Int; const Value: Char);// D2009 internal error {$IFDEF
  SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4686: { MoveWideChar copies Count WideChars from Source to Dest }
4687: Proc MoveNativeChar(const Source; var Dest; Count: Int);// D2009 internal error {$IFDEF SUPPORTS_INLINE}
  inline; {$ENDIF SUPPORTS_INLINE}
4688: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4689: Func IsSubString(const S:Str; StartIndex: Int; const SubStr:Str):Bool;
4690: { Spaces returns string consists on N space chars }
4691: Func Spaces(const N: Int):Str;
4692: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4693: Func AddSpaces(const S:Str; const N: Int):Str;
4694: Func SpacesW(const N: Int): WideString;
4695: Func AddSpacesW(const S: WideString; const N: Int): WideString;
4696: { Func LastDateRUS for russian users only }
4697: { returns date relative to current date: 'äää äiy iäqää' }
4698: Func LastDateRUS(const Dat: TDateTime):Str;
4699: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4700: Func CurrencyToStr(const Cur: Currency):Str;
4701: { HasChar returns True, if Char, Ch, contains in string, S }
4702: Func HasChar(const Ch: Char; const S:Str):Bool;
4703: Func HasCharW(const Ch: WideChar; const S: WideString):Bool; inline; {$ENDIF SUPPORTS_INLINE}
4704: Func HasAnyChar(const Chars:Str; const S:Str):Bool;
4705: {$IFDEF COMPILER12_UP}
4706: Func CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4707: {$ENDIF ~COMPILER12_UP}
4708: Func CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF SUPPORTS_INLINE}
4709: Func CountOfChar(const Ch: Char; const S:Str): Int;
4710: Func DefStr(const S:str; Default:str):str; {$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4711: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4712: Func StrLICompW2(S1, S2: PWideChar; MaxLen: Int): Int;

```



```

4713: Func StrPosW(S, SubStr: PWideChar): PWideChar;
4714: Func StrLenW(S: PWideChar): Int;
4715: Func TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4716: Func TrimLeftW(const S:WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4717: Func TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4718: TPixelFormat, '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4719: TMappingMethod, '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4720: Func GetBitmapPixelFormat( Bitmap : TBitmap ) : TPixelFormat
4721: Func GetPaletteBitmapFormat( Bitmap : TBitmap ) : TPixelFormat
4722: Proc SetBitmapPixelFormat(Bitmap: TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod)
4723: Func BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4724: Proc GrayscaleBitmap( Bitmap : TBitmap)
4725: Func BitmapToMemory( Bitmap : TBitmap; Colors : Int ) : TStream
4726: Proc SaveBitmapToFile( const FileName :Str; Bitmap : TBitmap; Colors : Int)
4727: Func ScreenPixelFormat : TPixelFormat
4728: Func ScreenColorCount : Int
4729: Proc TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4730: Func ZoomImage( ImageW, ImageH, MaxW, MaxH : Int; Stretch :Bool): TPoint
4731: // SIRegister_TJvGradient(CL);
4732:
4733: {***** files routines}
4734: Proc SetDelimitedText(List: TStrings; const Text:Str; Delimiter: Char);
4735: const
4736: {$IFDEF MSWINDOWS}
4737: DefaultCaseSensitivity = False;
4738: {$ENDIF MSWINDOWS}
4739: {$IFDEF UNIX}
4740: DefaultCaseSensitivity = True;
4741: {$ENDIF UNIX}
4742: { GenTempFileName returns temporary file name on
4743: drive, there FileName is placed }
4744: Func GenTempFileName(FileName:Str):Str;
4745: { GenTempFileNameExt same to previous function, but
4746: returning filename has given extension, FileExt }
4747: Func GenTempFileNameExt(FileName:Str; const FileExt:Str):Str;
4748: { ClearDir clears folder Dir }
4749: Func ClearDir(const Dir:Str):Bool;
4750: { DeleteDir clears and than delete folder Dir }
4751: Func DeleteDir(const Dir:Str):Bool;
4752: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4753: Func FileEquMask(FileName,Mask:TFileName;CaseSensitive:Boolean=DefaultCaseSensitivity):Bool;
4754: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4755: Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4756: Func FileEquMasks(FileName, Masks: TFileName;
4757: CaseSensitive:Bool = DefaultCaseSensitivity):Bool;
4758: Func DeleteFiles(const Folder: TFileName; const Masks:Str):Bool;
4759: {$IFDEF MSWINDOWS}
4760: { LZFileExpand expand file, FileSource,
4761: into FileDest. Given file must be compressed, using MS Compress program }
4762: Func LZFileExpand(const FileSource, FileDest:Str):Bool;
4763: {$ENDIF MSWINDOWS}
4764: { FileGetInfo fills SearchRec record for specified file attributes}
4765: Func FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec):Bool;
4766: { HasSubFolder returns True, if folder APath contains other folders }
4767: Func HasSubFolder(APath: TFileName):Bool;
4768: { IsEmptyFolder returns True, if there are no files or
4769: folders in given folder, APath}
4770: Func IsEmptyFolder(APath: TFileName):Bool;
4771: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4772: Func AddSlash(const Dir: TFileName):Str; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4773: { AddPath returns FileName with Path, if FileName not contain any path }
4774: Func AddPath(const FileName, Path: TFileName): TFileName;
4775: Func AddPaths(const PathList, Path:Str):Str;
4776: Func ParentPath(const Path: TFileName): TFileName;
4777: Func FindInPath(const FileName, PathList:Str): TFileName;
4778: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4779: Func DeleteReadOnlyFile(const FileName: TFileName):Bool;
4780: { HasParam returns True, if program running with specified parameter, Param }
4781: Func HasParam(const Param:Str):Bool;
4782: Func HasSwitch(const Param:Str):Bool;
4783: Func Switch(const Param:Str):Str;
4784: { ExePath returns ExtractFilePath(ParamStr(0)) }
4785: Func ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4786: Func CopyDir(const SourceDir, DestDir: TFileName):Bool;
4787: //Func FileTimeToDateTime(const FT: TFileTime): TDateTime;
4788: Proc FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4789: Func MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4790: {**** Graphic routines }
4791: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4792: Func IsTTFontSelected(const DC: HDC):Bool;
4793: Func KeyPressed(VK: Int):Bool;
4794: Func isKeyPressed:Bool; //true if key on memo2 (shell output) is pressed
4795: { TrueInflateRect inflates rect in other method, than InflateRect API Func }
4796: Func TrueInflateRect(const R: TRect; const I: Int): TRect;
4797: {**** Color routines }
4798: Proc RGBToHSV(R, G, B: Int; var H, S, V: Int);
4799: Func RGBToBGR(Value:Card):Card;
4800: //Func ColorToPrettyName(Value: TColor):Str;
4801: //Func PrettyNameToColor(const Value:Str): TColor;

```

```

4802: {**** other routines }
4803: Proc SwapInt(var Int1, Int2: Int); { $IFDEF SUPPORTS_INLINE inline; $ENDIF SUPPORTS_INLINE }
4804: Func IntPower(Base, Exponent: Int): Int;
4805: Func ChangeTopException(E: TObject; TOBject; //Linux version writes error message to ErrOutput
4806: Func StrToBool(const S: Str): Bool;
4807: Func Var2Type(V: Variant; const DestVarType: Int): Variant;
4808: Func VarToInt(V: Variant): Int;
4809: Func VarToFloat(V: Variant): Double;
4810: {following functions are not documented cause not work properly sometimes,so do not use them}
4811: // (rom) ReplaceStrings1, GetSubStr removed
4812: Func GetLongFileName(const FileName: Str): Str;
4813: Func FileNewExt(const FileName, NewExt: TFileName): TFileName;
4814: Func GetParameter: Str;
4815: Func GetComputerID: Str;
4816: Func GetComputerName: Str;
4817: {**** string routines }
4818: { ReplaceAllStrings searches for all substrings, Words,
4819:   in a string, S, and replaces them with Frases with the same Index. }
4820: Func ReplaceAllStrings(const S: Str; Words, Frases: TStrings): Str;
4821: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4822:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with same
   Index, and then update NewSelStart variable }
4823: Func ReplaceStrings(const S: str; PosBeg, Len: Int; Words, Frases: TStrings; var NewSelStart: Int): str;
4824: { CountOfLines calculates the lines count in a string, S,
4825:   each line must be separated from another with CrLf sequence }
4826: Func CountOfLines(const S: Str): Int;
4827: { DeleteLines deletes all lines from strings which in the words, words.
4828:   The word of will be deleted from strings. }
4829: Proc DeleteOfLines(Ss: TStrings; const Words: array of string);
4830: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4831:   Lines contained only spaces also deletes. }
4832: Proc DeleteEmptyLines(Ss: TStrings);
4833: { SQLAddWhere addes or modifies existing where-statement, where,
4834:   to the strings, SQL. Note: If strings SQL already contains where-statement,
4835:   it must be started on the beginning of any line }
4836: Proc SQLAddWhere(SQL: TStrings; const Where: Str);
4837: {**** files routines - }
4838: { $IFDEF MSWINDOWS }
4839: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4840:   Resource can be compressed using MS Compress program}
4841: Func ResSaveToFile(const Typ, Name: str; const Compressed: Bool; const FileName: str): Bool;
4842: Func ResSaveToFileEx(Instance: HINST; Typ, Name: PChar; const Compressed: Boolean; const FileName: str): Bool;
4843: Func ResSaveToString(Instance: HINST; const Typ, Name: str; var S: str): Boolean;
4844: { $ENDIF MSWINDOWS }
4845: { IniReadSection read section, Section, from ini-file,
4846:   IniFileName, into strings, Ss. This Func reads ALL strings from specified section.
4847:   Note: TIniFile.ReadSection Func reads only strings with '=' symbol.}
4848: Func IniReadSection(const IniFileName: TFileName; const Section: str; Ss: TStrings): Bool;
4849: { LoadTextFile load text file, FileName, into string }
4850: Func LoadTextFile(const FileName: TFileName): Str;
4851: Proc SaveTextFile(const FileName: TFileName; const Source: Str);
4852: {ReadFolder reads filelist from diskfolder, Folder, that are equal to mask, Mask, into strings, FileList}
4853: Func ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Int;
4854: Func ReadFolders(const Folder: TFileName; FolderList: TStrings): Int;
4855: { RAtextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip.}
4856: Proc RAtextOut(Canvas: TCanvas; const R, RClip: TRect; const S: Str);
4857: { RAtextOutEx same with RAtextOut function, but can calculate needed height for correct output}
4858: Func RAtextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: str; const CalcHeight: Bool): Int;
4859: { RAtextCalcHeight calculate needed height for
4860:   correct output, using RAtextOut or RAtextOutEx functions }
4861: Func RAtextCalcHeight(Canvas: TCanvas; const R: TRect; const S: Str): Int;
4862: { Cinema draws some visual effect }
4863: Proc Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4864: { Roughed fills rect with special 3D pattern }
4865: Proc Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Bool);
4866: { 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 }
4867: Func BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Int): TBitmap;
4868: { TextWidth calculate text with for writing using standard desktop font }
4869: Func TextWidth(const AStr: Str): Int;
4870: { TextHeight calculate text height for writing using standard desktop font }
4871: Func TextHeight(const AStr: Str): Int;
4872: Proc SetChildPropOrd(Owner: TComponent; const PropName: Str; Value: Longint);
4873: Proc Error(const Msg: Str);
4874: Proc ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: str;
4875:   const HideSelColor: Bool; var PlainItem: Str; var Width: Int; CalcWidth: Bool);
4876: {ex.Text parameter: 'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4877: Func ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4878:   const State: TOwnerDrawState; const Text: Str; const HideSelColor: Bool): Str;
4879: Func ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4880:   const State: TOwnerDrawState; const Text: Str; const HideSelColor: Bool): Int;
4881: Func ItemHtPlain(const Text: Str): Str;
4882: { ClearList - clears list of TObject }
4883: Proc ClearList(List: TList);
4884: Proc MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
4885: Proc ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
4886: { RTTI support }
4887: Func GetPropType(Obj: TObject; const PropName: Str): TTypeKind;
4888: Func GetPropStr(Obj: TObject; const PropName: Str): Str;

```

```

4889: Func GetPropOrd(Obj: TObject; const PropName:Str): Int;
4890: Func GetPropMethod(Obj: TObject; const PropName:Str): TMethod;
4891: Proc PrepareIniSection(Ss: TStrings);
4892: {following functions are not documented because they are don't work properly,so don't use them}
4893: // (rom) from JvBandWindows to make it obsolete
4894: Func PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4895: // (rom) from JvBandUtils to make it obsolete
4896: Func iif(const Test:Bool; const ATrue, AFalse: Variant): Variant;
4897: Proc CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4898: Func CreateIconFromClipboard: TIcon;
4899: { begin JvIconClipboardUtils } { Icon clipboard routines }
4900: Func CF_ICON: Word;
4901: Proc AssignClipboardIcon(Icon: TIcon);
4902: { Real-size icons support routines (32-bit only) }
4903: Proc GetIconSize(Icon: HICON; var W, H: Int);
4904: Func CreateRealSizeIcon(Icon: TIcon; HICON;
4905: Proc DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Int);
4906: {end JvIconClipboardUtils }
4907: Func CreateScreenCompatibleDC: HDC;
4908: Func InvalidateRect(hWnd: HWND; const lpRect:TRect; bErase:BOOL):BOOL; overload; {$IFDEF SUPPORTS_INLINE}
inline; {$ENDIF}
4909: Func InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4910: { begin JvRLE } // (rom) changed API for inclusion in JCL
4911: Proc RleCompressTo(InStream, OutStream: TStream);
4912: Proc RleDecompressTo(InStream, OutStream: TStream);
4913: Proc RleCompress(Stream: TStream);
4914: Proc RleDecompress(Stream: TStream);
4915: { end JvRLE } { begin JvDateUtil }
4916: Func CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4917: Func IsLeapYear(AYear: Int):Bool;
4918: Func DaysInAMonth(const AYear, AMonth: Word): Word;
4919: Func DaysPerMonth(AYear, AMonth: Int): Int;
4920: Func FirstDayOfPrevMonth: TDateTime;
4921: Func LastDayOfPrevMonth: TDateTime;
4922: Func FirstDayOfNextMonth: TDateTime;
4923: Func ExtractDay(ADate: TDateTime): Word;
4924: Func ExtractMonth(ADate: TDateTime): Word;
4925: Func ExtractYear(ADate: TDateTime): Word;
4926: Func IncDate(ADate: TDateTime; Days, Months, Years: Int): TDateTime;
4927: Func IncDay(ADate: TDateTime;Delta:Int):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4928: Func IncMonth(ADate: TDateTime; Delta: Int): TDateTime;
4929: Func IncYear(ADate: TDateTime; Delta: Int): TDateTime;
4930: Func ValidDate(ADate: TDateTime):Bool;
4931: Proc DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4932: Func MonthsBetween(Date1, Date2: TDateTime): Double;
4933: Func DaysInPeriod(Date1, Date2: TDateTime): Longint;
4934: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4935: Func DaysBetween(Date1, Date2: TDateTime): Longint;
4936: { The same as previous but if Date2 < Date1 result = 0 }
4937: Func IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Int): TDateTime;
4938: Func IncHour(ATime: TDateTime; Delta: Int): TDateTime;
4939: Func IncMinute(ATime: TDateTime; Delta: Int): TDateTime;
4940: Func IncSecond(ATime: TDateTime; Delta: Int): TDateTime;
4941: Func IncMSec(ATime: TDateTime; Delta: Int): TDateTime;
4942: Func CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4943: { String to date conversions }
4944: Func GetDateOrder(const DateFormat:Str): TDateOrder;
4945: Func MonthFromName(const S:Str; MaxLen: Byte): Byte;
4946: Func StrToDateDef(const S:Str; Default: TDateTime): TDateTime;
4947: Func StrToDateFmt(const DateFormat, S:Str): TDateTime;
4948: Func StrToDateFmtDef(const DateFormat, S:Str; Default: TDateTime): TDateTime;
4949: //Func DefDateFormat(AFourDigitYear:Bool):Str;
4950: //Func DefDateMask(BlanksChar: Char; AFourDigitYear:Bool):Str;
4951: Func FormatLongDate(Value: TDateTime):Str;
4952: Func FormatLongDateTime(Value: TDateTime):Str;
4953: { end JvDateUtil }
4954: Func BufToBinStr(Buf: Pointer; BufSize: Int):Str;
4955: Func BinStrToBuf(Value:Str; Buf: Pointer; BufSize: Int): Int;
4956: { begin JvStrUtils } { ** Common string handling routines ** }
4957: {$IFDEF UNIX}
4958: Func iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes:Card;
const ToCode, FromCode: Ansistr):Bool;
4959: Func iconvString(const S, ToCode, FromCode: Ansistr):Str;
4960: Func iconvWideString(const S: WideString; const ToCode, FromCode:Ansistr):WideString;
4961: Func OemStrToAnsi(const S: Ansistr): Ansistr;
4962: Func AnsiStrToOem(const S: Ansistr): Ansistr;
4963: {$ENDIF UNIX}
4964: Func StrToOem(const Ansistr: Ansistr): Ansistr;
4965: { StrToOem translates a string from the Windows character set into the OEM character set. }
4966: Func OemToAnsiStr(const OemStr: Ansistr): Ansistr;
4967: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4968: Func IsEmptyStr(const S:Str; const EmptyChars: TSysCharSet):Bool;
4969: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4970: Func ReplaceStr(const S, Srch, Replace:Str):Str;
4971: { Returns string with every occurrence of Srch string replaced with Replace string. }
4972: Func DelSpace(const S:Str):Str;
4973: { DelSpace return a string with all white spaces removed. }
4974: Func DelChars(const S:Str; Chr: Char):Str;
4975: { DelChars return a string with all Chr characters removed. }
4976:

```

```

4977: Func DelBSpace(const S:Str):Str;
4978: { DelBSpace trims leading spaces from given string. }
4979: Func DelESpace(const S:Str):Str;
4980: { DelESpace trims trailing spaces from given string. }
4981: Func DelRSpace(const S:Str):Str;
4982: { DelRSpace trims leading and trailing spaces from the given string. }
4983: Func DelSpace1(const S:Str):Str;
4984: { DelSpace1 return a string with all non-single white spaces removed. }
4985: Func Tab2Space(const S:Str; Numb: Byte):Str;
4986: { Tab2Space converts any tabulation character in given string to the
4987:   Numb spaces characters. }
4988: Func NPos(const C:Str; S:Str; N: Int): Int;
4989: { NPos searches for a N-th position of substring C in a given string. }
4990: Func MakeStr(C: Char; N: Int):Str; overload;
4991: { $IFDEF COMPILER12 UP }
4992: Func MakeStr(C: WideChar; N: Int): WideString; overload;
4993: { $ENDIF !COMPILER12 UP }
4994: Func MS(C: Char; N: Int):Str; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF SUPPORTS_INLINE }
4995: { MakeStr return a string of length N filled with character C. }
4996: Func AddChar(C: Char; const S:Str; N: Int):Str;
4997: { AddChar return a string left-padded to length N with characters C. }
4998: Func AddCharR(C: Char; const S:Str; N: Int):Str;
4999: { AddCharR return a string right-padded to length N with characters C. }
5000: Func LeftStr(const S:Str; N: Int):Str;
5001: { LeftStr return a string right-padded to length N with blanks. }
5002: Func RightStr(const S:Str; N: Int):Str;
5003: { RightStr return a string left-padded to length N with blanks. }
5004: Func CenterStr(const S:Str; Len: Int):Str;
5005: { CenterStr centers the characters in the string based upon the Len specified. }
5006: Func CompStr(const S1, S2:Str): Int; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF SUPPORTS_INLINE }
5007: { CompStr compares S1 to S2, with case-sensitivity. The return value is
5008:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
5009: Func CompText(const S1, S2:Str):Int; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF SUPPORTS_INLINE }
5010: { CompText compares S1 to S2, without case-sensitivity. return value is same as for CompStr. }
5011: Func Copy2Symb(const S:Str; Symb: Char):Str;
5012: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
5013: Func Copy2SymbDel(var S:Str; Symb: Char):Str;
5014: { Copy2SymbDel returns a substring of a string S from beginning to first
5015:   character Symb and removes this substring from S. }
5016: Func Copy2Space(const S:Str):Str;
5017: { Copy2Symb returns a substring of a string S from beginning to first white space. }
5018: Func Copy2SpaceDel(var S:Str):Str;
5019: { Copy2SpaceDel returns a substring of a string S from beginning to first
5020:   white space and removes this substring from S. }
5021: Func AnsiProperCase(const S:Str; const WordDelims: TSysCharSet):Str;
5022: { Returns string, with the first letter of each word in uppercase,
5023:   all other letters in lowercase. Words are delimited by WordDelims. }
5024: Func WordCount(const S:Str; const WordDelims: TSysCharSet): Int;
5025: { WordCount given a set of word delimiters, returns number of words in S. }
5026: Func WordPosition(const N: Int; const S:Str; const WordDelims: TSysCharSet):Int;
5027: { Given a set of word delimiters, returns start position of N'th word in S. }
5028: Func ExtractWord(N: Int; const S:Str; const WordDelims: TSysCharSet):Str;
5029: Func ExtractWordPos(N: Int; const S:Str; const WordDelims: TSysCharSet; var Pos: Int):Str;
5030: Func ExtractDelimited(N: Int; const S:Str; const Delims: TSysCharSet):Str;
5031: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
5032:   delimiters, return the N'th word in S. }
5033: Func ExtractSubstr(const S:Str; var Pos: Int; const Delims: TSysCharSet):Str;
5034: { ExtractSubstr given a set of word delimiters, returns the substring from S,
5035:   that started from position Pos. }
5036: Func IsWordPresent(const W, S:Str; const WordDelims: TSysCharSet):Bool;
5037: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
5038: Func QuotedString(const S:Str; Quote: Char):Str;
5039: { QuotedString returns the given string as a quoted string, using provided Quote character. }
5040: Func ExtractQuotedString(const S:Str; Quote: Char):Str;
5041: { ExtractQuotedString removes the Quote characters from beginning and
5042:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character. }
5043: Func FindPart(const HelpWilds, InputStr:Str): Int;
5044: { FindPart compares a string with '?' and another, returns position of HelpWilds in InputStr. }
5045: Func IsWild(InputStr, Wilds:Str; IgnoreCase:Bool):Bool;
5046: { IsWild compares InputString with WildCard string and returns True if corresponds. }
5047: Func XorString(const Key, Src: ShortString): ShortString;
5048: Func XorEncode(const Key, Source:Str):Str;
5049: Func XorDecode(const Key, Source:Str):Str;
5050: { ** Command line routines ** }
5051: Func GetCmdLineArg(const Switch:Str; ASwitchChars: TSysCharSet):Str;
5052: { ** Numeric string handling routines ** }
5053: Func Numb2USA(const S:Str):Str;
5054: { Numb2USA converts numeric string S to USA-format. }
5055: Func Dec2Hex(N: Longint; A: Byte):Str; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF SUPPORTS_INLINE }
5056: { Dec2Hex converts the given value to a hexadecimal string representation
5057:   with the minimum number of digits (A) specified. }
5058: Func Hex2Dec(const S:Str): Longint;
5059: { Hex2Dec converts the given hexadecimal string to the corresponding Int value. }
5060: Func Dec2Numb(N: Int64; A, B: Byte):Str;
5061: { Dec2Numb converts the given value to a string representation with the
5062:   base equal to B and with the minimum number of digits (A) specified. }
5063: Func Numb2Dec(S:Str; B: Byte): Int64;
5064: { Numb2Dec converts the given B-based numeric string to corresponding Int value. }
5065: Func IntToBin(Value: Longint; Digits, Spaces: Int):Str;

```

```

5066: { IntToBin converts the given value to a binary string representation
5067:   with the minimum number of digits specified. }
5068: Func IntToRoman(Value: Longint):Str;
5069: { IntToRoman converts the given value to a roman numeric string representation. }
5070: Func RomanToInt(const S:Str): Longint;
5071: { RomanToInt converts the given string to an Int value. If the string
5072:   doesn't contain a valid roman numeric value, the 0 value is returned. }
5073: Func FindNotBlankCharPos(const S:Str): Int;
5074: Func FindNotBlankCharPosW(const S: WideString): Int;
5075: Func AnsiChangeCase(const S:Str):Str;
5076: Func WideChangeCase(const S:Str):Str;
5077: Func StartsText(const SubStr, S:Str):Bool;
5078: Func EndsText(const SubStr, S:Str):Bool;
5079: Func DequotedStr(const S:Str; QuoteChar: Char = ''' ):Str;
5080: Func AnsiDequotedStr(const S:Str; AQuote:Char):str;//follow Delphi 2009's "Ansi" prefix
5081: {end JvStrUtils}
5082: {$IFDEF UNIX}
5083: Func GetTempFileName(const Prefix: Ansistr):Ansistr;
5084: {$ENDIF UNIX}
5085: { begin JvFileUtil }
5086: Func FileDateTime(const FileName:Str): TDateTime;
5087: Func HasAttr(const FileName:Str; Attr: Int):Bool;
5088: Func DeleteFilesEx(const FileMasks: array of string):Bool;
5089: Func NormlDir(const DirName:Str):Str;
5090: Func RemoveBackSlash(const DirName:Str):Str; //only for Windows/DOS Paths
5091: Func ValidFileName(const FileName:Str):Bool;
5092: {$IFDEF MSWINDOWS}
5093: Func FileLock(Handle: Int; Offset, LockSize: Longint): Int; overload;
5094: Func FileLock(Handle: Int; Offset, LockSize: Int64): Int; overload;
5095: Func FileUnlock(Handle: Int; Offset, LockSize: Longint): Int; overload;
5096: Func FileUnlock(Handle: Int; Offset, LockSize: Int64): Int; overload;
5097: {$ENDIF MSWINDOWS}
5098: Func GetWindowsDir:Str;
5099: Func GetSystemDir:Str;
5100: Func ShortToLongFileName(const ShortName:Str):Str;
5101: Func LongToShortFileName(const LongName:Str):Str;
5102: Func ShortToLongPath(const ShortName:Str):Str;
5103: Func LongToShortPath(const LongName:Str):Str;
5104: {$IFDEF MSWINDOWS}
5105: Proc CreateFileLink(const FileName, DisplayName:Str; Folder: Int);
5106: Proc DeleteFileLink(const DisplayName:Str; Folder: Int);
5107: {$ENDIF MSWINDOWS}
5108: { end JvFileUtil }
5109: // Works like PtInRect but includes all edges in comparision
5110: Func PtInRectInclusive(R: TRect; Pt: TPoint):Bool;
5111: // Works like PtInRect but excludes all edges from comparision
5112: Func PtInRectExclusive(R: TRect; Pt: TPoint):Bool;
5113: Func FourDigitYear:Bool; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5114: Func IsFourDigitYear:Bool;
5115: { moved from JvJVCUtils }
5116: //Open an object with the shell (url or something like that)
5117: Func OpenObject(const Value:Str):Bool; overload;
5118: Func OpenObject(Value: PChar):Bool; overload;
5119: {$IFDEF MSWINDOWS}
5120: //Raise the last Exception
5121: Proc RaiseLastWin32; overload;
5122: Proc RaiseLastWin32(const Text:Str); overload;
5123: //Raise last Exception with a small comment from your part {GetFileVersion returns most significant 32
  bits of a file's binary version number. Typically, this includes major/minor version placed together in
  one 32-bit Int.
5124: Func GetFileVersion(const AFileName:Str):Card;
5125: {$EXTERNALSYM GetFileVersion}
5126: //Get version of Shell.dll
5127: Func GetShellVersion:Card;
5128: {$EXTERNALSYM GetShellVersion}
5129: // CD functions on HW
5130: Proc OpenCdDrive;
5131: Proc CloseCdDrive;
5132: // returns True if Drive is accessible
5133: Func DiskInDrive(Drive: Char):Bool;
5134: {$ENDIF MSWINDOWS}
5135: //Same as linux Func ;)
5136: Proc PError(const Text:Str);
5137: // execute a program without waiting
5138: Proc Exec(const FileName, Parameters, Directory:Str);
5139: // execute a program and wait for it to finish
5140: Func ExecuteAndWait(CmdLine:str;const WorkingDirectory:str;Visibility:Int=SW_SHOW):Int;
5141: // returns True if this is the first instance of the program that is running
5142: Func FirstInstance(const ATitle:Str):Bool;
5143: // restores a window based on it's classname and Caption. Either can be left empty
5144: // to widen the search
5145: Proc RestoreOtherInstance(const MainFormClassName, MainFormCaption:Str);
5146: // manipulate the traybar and start button
5147: Proc HideTraybar;
5148: Proc ShowTraybar;
5149: Proc ShowStartButton(Visible:Bool = True);
5150: //(rom) SC_MONITORPOWER documented as Win 95 only(rom) do some testingset monitor functions
5151: Proc MonitorOn;
5152: Proc MonitorOff;

```



```

5153: Proc LowPower;
5154: // send a key to the window named AppName
5155: Func SendKey(const AppName:Str; Key: Char):Bool;
5156: {$IFDEF MSWINDOWS}
5157: // returns list of all win currently visible, Objects property is filled with window handle
5158: Proc GetVisibleWindows(List: TStrings);
5159: Func GetVisibleWindowsF( List : TStrings):TStrings;
5160: // associates an extension to a specific program
5161: Proc AssociateExtension(const IconPath, ProgramName, Path, Extension:Str);
5162: Proc AddToRecentDocs(const FileName:Str);
5163: Func GetRecentDocs: TStringList;
5164: {$ENDIF MSWINDOWS}
5165: Func CharIsMoney(const Ch: Char):Bool;
5166: //Func StrToCurrDef(const Str:Str; Def: Currency): Currency;
5167: Func IntToExtended(I: Int): Extended;
5168: { GetChangedText works out new text given the current cursor pos & the key pressed
5169:   is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit}
5170: Func GetChangedText(const Text:Str; SelStart, SelLength: Int; Key: Char):Str;
5171: Func MakeYear4Digit(Year, Pivot: Int): Int;
5172: //Func StrIsInt(const S:Str):Bool;
5173: Func StrIsFloatMoney(const Ps:Str):Bool;
5174: Func StrIsDateTime(const Ps:Str):Bool;
5175: Func PreformatDateString(Ps:Str):Str;
5176: Func BooleanToInt(const B:Bool): Int;
5177: Func StringToBoolean(const Ps:Str):Bool;
5178: Func SafeStrToDateTime(const Ps:Str): TDateTime;
5179: Func SafeStrToDate(const Ps:Str): TDateTime;
5180: Func SafeStrToTime(const Ps:Str): TDateTime;
5181: Func StrDelete(const psSub, psMain:Str):Str;
5182: { returns the fractional value of pcValue}
5183: Func TimeOnly(pcValue: TDateTime): TTime;
5184: { returns the integral value of pcValue }
5185: Func DateOnly(pcValue: TDateTime): TDate;
5186: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5187: const { TDateTime value used to signify Null value}
5188:   NullEquivalentDate: TDateTime = 0.0;
5189: Func DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind):Bool;
5190: // Replacement for Win32Check to avoid platform specific warnings in D6
5191: Func OSCheck(RetVal:Bool):Bool;
5192: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5193:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5194:   not be forced to use FileCtrl unnecessarily }
5195: Func MinimizeFileName(const FileName:Str; Canvas: TCanvas; MaxLen: Int):Str;
5196: Func MinimizeText(const Text:Str; Canvas: TCanvas; MaxWidth: Int):Str;
5197: {MinimizeString truncates longstring, S, and appends '...' symbols, if Length of S is more than MaxLen}
5198: Func MinimizeString(const S:Str; const MaxLen: Int):Str;
5199: Proc RunDll32Internal(Wnd:THandle; const DLLName, FuncName, CmdLine:Str; CmdShow: Int= SW_SHOWDEFAULT);
5200: { 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.}
5201: Func GetDLLVersion(const DLLName:Str; var pdwMajor, pdwMinor: Int):Boolean;
5202: {$ENDIF MSWINDOWS}
5203: Proc ResourceNotFound(ResID: PChar);
5204: Func EmptyRect: TRect;
5205: Func RectWidth(R: TRect): Int;
5206: Func RectHeight(R: TRect): Int;
5207: Func CompareRect(const R1, R2: TRect):Bool;
5208: Proc RectNormalize(var R: TRect);
5209: Func RectIsSquare(const R: TRect):Bool;
5210: Func RectSquare(var ARect: TRect; AMaxSize: Int = -1):Bool;
5211: //If AMaxSize = -1, then auto calc Square's max size
5212: {$IFDEF MSWINDOWS}
5213: Proc FreeUnusedOle;
5214: Func GetWindowsVersion:Str;
5215: Func LoadDLL(const LibName:Str): THandle;
5216: Func RegisterServer(const ModuleName:Str): Bool;
5217: Func UnregisterServer(const ModuleName:Str): Bool;
5218: {$ENDIF MSWINDOWS}
5219: { String routines }
5220: Func GetEnvVar(const VarName:Str):Str;
5221: Func AnsiUpperFirstChar(const S:Str):Str; //follow Delphi2009's ex. with "Ansi" prefix
5222: Func StringToPChar(var S:Str): PChar;
5223: Func StrPAlloc(const S:Str): PChar;
5224: Proc SplitCommandLine(const CmdLine:Str; var ExeName, Params:Str);
5225: Func DropT(const S:Str):Str;
5226: { Memory routines }
5227: Func AllocMemo(Size: Longint): Pointer;
5228: Func ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5229: Proc FreeMemo(var fpBlock: Pointer);
5230: Func GetMemoSize(fpBlock: Pointer): Longint;
5231: Func CompareMem(fpBlock1, fpBlock2: Pointer; Size:Card):Bool;
5232: { Manipulate huge pointers routines }
5233: Proc HugeInc(var HugePtr: Pointer; Amount: Longint);
5234: Proc HugeDec(var HugePtr: Pointer; Amount: Longint);
5235: Func HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5236: Proc HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5237: Proc HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5238: Func WindowClassName(Wnd: THandle):Str;
5239: Proc SwitchToWindow(Wnd: THandle; Restore:Bool);
5240: Proc ActivateWindow(Wnd: THandle);

```

```

5241: Proc ShowWinNoAnimate(Handle: THandle; CmdShow: Int);
5242: Proc KillMessage(Wnd: THandle; Msg:Card);
5243: { SetWindowTop put window to top without recreating window }
5244: Proc SetWindowTop(const Handle: THandle; const Top:Bool);
5245: Proc CenterWindow(Wnd: THandle);
5246: Func MakeVariant(const Values: array of Variant): Variant;
5247: { Convert dialog units to pixels and backwards }
5248: { $IFDEF MSWINDOWS }
5249: Func DialogUnitsToPixelsX(DlgUnits:Word):Word;
5250: Func DialogUnitsToPixelsY(DlgUnits:Word):Word;
5251: Func PixelsToDialogUnitsX(PixUnits:Word):Word;
5252: Func PixelsToDialogUnitsY(PixUnits:Word):Word;
5253: { $ENDIF MSWINDOWS }
5254: Func GetUniqueFileNameInDir(const Path, FileNameMask:Str):Str;
5255: { $IFDEF BCB }
5256: Func FindPrevInstance(const MainFormClass: ShortString;const ATitle:Str): THandle;
5257: Func ActivatePrevInstance(const MainFormClass: ShortString;const ATitle:Str):Bool;
5258: { $ELSE }
5259: Func FindPrevInstance(const MainFormClass, ATitle:Str): THandle;
5260: Func ActivatePrevInstance(const MainFormClass, ATitle:Str):Bool;
5261: { $ENDIF BCB }
5262: { $IFDEF MSWINDOWS }
5263: { BrowseForFolderNative displays Browse For Folder dialog }
5264: Func BrowseForFolderNative(const Handle:THandle;const Title:str;var Folder:str):Bool;
5265: { $ENDIF MSWINDOWS }
5266: Proc AntiAlias(Clip: TBitmap);
5267: Proc AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Int);
5268: Proc CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5269:   ABitmap: TBitmap; const SourceRect: TRect);
5270: Func IsTrueType(const FontName:Str):Bool;
5271: // Removes all non-numeric characters from AValue and returns the resulting string
5272: Func TextToValText(const AValue:Str):Str;
5273: Func ExecRegExpr( const ARegExpr, AInputStr : RegExprString) :Bool
5274: Proc SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5275: Func ReplaceRegExpr(const ARegExpr,AInputStr,
  AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5276: Func QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5277: Func RegExprSubExpressions(const ARegExpr:str;ASubExprs:TStrings;AExtendedSyntax:bool):
5278:
5279: *****unit uPSI_JvTFUtils;
5280: Func JExtractYear( ADate : TDateTime) : Word
5281: Func JExtractMonth( ADate : TDateTime) : Word
5282: Func JExtractDay( ADate : TDateTime) : Word
5283: Func ExtractHours( ATime : TDateTime) : Word
5284: Func ExtractMins( ATime : TDateTime) : Word
5285: Func ExtractSecs( ATime : TDateTime) : Word
5286: Func ExtractMSecs( ATime : TDateTime) : Word
5287: Func FirstOfMonth( ADate : TDateTime) : TDateTime
5288: Func GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5289: Func GetWeeksInMonth( Year, Month : Word; StartOfWeek : Int) : Word
5290: Proc IncBorlDOW( var BorlDOW : Int; N : Int)
5291: Proc IncDOW( var DOW : TTFDayOfWeek; N : Int)
5292: Proc IncDays( var ADate : TDateTime; N : Int)
5293: Proc IncWeeks( var ADate : TDateTime; N : Int)
5294: Proc IncMonths( var ADate : TDateTime; N : Int)
5295: Proc IncYears( var ADate : TDateTime; N : Int)
5296: Func EndOfMonth( ADate : TDateTime) : TDateTime
5297: Func IsFirstOfMonth( ADate : TDateTime) :Bool
5298: Func IsEndOfMonth( ADate : TDateTime) :Bool
5299: Proc EnsureMonth( Month : Word)
5300: Proc EnsureDOW( DOW : Word)
5301: Func EqualDates( D1, D2 : TDateTime) :Bool
5302: Func Lesser( N1, N2 : Int) : Int
5303: Func Greater( N1, N2 : Int) : Int
5304: Func GetDivLength( TotalLength, DivCount, DivNum : Int) : Int
5305: Func GetDivNum( TotalLength, DivCount, X : Int) : Int
5306: Func GetDivStart( TotalLength, DivCount, DivNum : Int) : Int
5307: Func DOWToBorl( ADOW : TTFDayOfWeek) : Int
5308: Func BorlToDOW( BorlDOW : Int) : TTFDayOfWeek
5309: Func DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5310: Proc CalcTextPos(HostRect:TRect;var TextLeft,TextTop:Int;var
  TextBounds:TRect;AFont:TFont;AAngle:Int;HAlign:TAlignment;VAlign:TJvTFVAlignment;ATxt:str)
5311: Proc DrawAngleText(ACanvas:TCanvas;HostRect:TRect;var TextBounds:TRect;AAngle:Int;HAlign:
  TAlignment;VAlign:TJvTFVAlignment;ATxt:str)
5312: Func JRectWidth( ARect : TRect) : Int
5313: Func JRectHeight( ARect : TRect): Int
5314: Func JEmptyRect : TRect
5315: Func IsClassByName( Obj : TObject; ClassName:str):Boolean
5316:
5317: Proc SIRegister_MSysUtils(CL: TPSPascalCompiler);
5318: begin
5319: Proc HideTaskBarButton( hWindow : HWND)
5320: Func msLoadStr( ID : Int) :Str
5321: Func msFormat( fmt :Str; params : array of const) :Str
5322: Func msFileExists( const FileName :Str) :Bool
5323: Func msIntToStr( Int : Int64) :Str
5324: Func msStrPas( const Str : PChar) :Str
5325: Func msRenameFile( const OldName, NewName :Str) :Bool
5326: Func CutFileName( s :Str) :Str

```

```

5327: Func GetVersionInfo( var VersionString :Str) : DWORD
5328: Func FormatTime( t :Card) :Str
5329: Func msCreateDir( const Dir :Str) :Bool
5330: Func SetAutoRun( NeedAutoRun :Bool; AppName :Str) :Bool
5331: Func SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5332: Func msStrLen( Str : PChar) : Int
5333: Func msDirectoryExists( const Directory :Str) :Bool
5334: Func GetFolder( hWnd : hWnd; RootDir : Int; Caption :Str) :Str
5335: Func SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5336: Func EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) :LRESULT
5337: Proc SetEditWndProc( hWnd : HWND; ptr : TObject)
5338: Func GetTextFromFile( Filename :Str) :Str
5339: Func IsTopMost(hWnd:HWND):Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5340: Func msStrToIntDef( const s :Str; const i : Int) : Int
5341: Func msStrToInt( s :Str) : Int
5342: Func GetItemText( hDlg : THandle; ID : DWORD):str
5343: end;
5344:
5345: Proc SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5346: begin
5347:   //TDynFloatArray', 'array of Extended
5348:   TDynLWordArray', 'array of LongWord
5349:   TDynLIntArray', 'array of LongInt
5350:   TDynFloatMatrix', 'array of TDynFloatArray
5351:   TDynLWordMatrix', 'array of TDynLWordArray
5352:   TDynLIntMatrix', 'array of TDynLIntArray
5353:   Func SquareAll( const X : TDynFloatArray) : TDynFloatArray
5354:   Func InverseAll( const X : TDynFloatArray) : TDynFloatArray
5355:   Func LnAll( const X : TDynFloatArray) : TDynFloatArray
5356:   Func Log10All( const X : TDynFloatArray) : TDynFloatArray
5357:   Func LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5358:   Func AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5359:   Func SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5360:   Func MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5361:   Func DotProduct( const X, Y : TDynFloatArray) : Extended
5362:   Func MNorm( const X : TDynFloatArray) : Extended
5363:   Func MatrixIsRectangular( const X : TDynFloatMatrix) :Bool
5364:   Proc MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Bool;
5365:   Func MatrixIsSquare( const X : TDynFloatMatrix) :Bool
5366:   Func MatricesSameDimensions( const X, Y : TDynFloatMatrix) :Bool
5367:   Func AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5368:   Proc AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5369:   Func SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5370:   Proc SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5371:   Func MultiplyMatrixByConst( const X: TDynFloatMatrix; const K: Extended):TDynFloatMatrix
5372:   Proc MultiplyMatrixByConst2( var X: TDynFloatMatrix; const K: Extended);
5373:   Func MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5374:   Func TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5375:   Func GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5376: end;
5377:
5378: Proc SIRegister_ESBMaths(CL: TPSPascalCompiler);
5379: begin
5380:   'ESBMinSingle','Single').setExtended( 1.5e-45);
5381:   'ESBMaxSingle','Single').setExtended( 3.4e+38);
5382:   'ESBMinDouble','Double').setExtended( 5.0e-324);
5383:   'ESBMaxDouble','Double').setExtended( 1.7e+308);
5384:   'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5385:   'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5386:   'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5387:   'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5388:   'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5389:   'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5390:   'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5391:   'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5392:   'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5393:   'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5394:   'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5395:   'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5396:   'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5397:   'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5398:   'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5399:   'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5400:   'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5401:   'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5402:   'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5403:   'ESBe','Extended').setExtended( 2.7182818284590452354);
5404:   'ESBe2','Extended').setExtended( 7.3890560989306502272);
5405:   'ESBePi','Extended').setExtended( 23.140692632779269006);
5406:   'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5407:   'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5408:   'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5409:   'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5410:   'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5411:   'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5412:   'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5413:   'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5414:   'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5415:   'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);

```

```

5416: 'ESBPi','Extended').setExtended( 3.1415926535897932385);
5417: 'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5418: 'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5419: 'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5420: 'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5421: 'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5422: 'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5423: 'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5424: 'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5425: 'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5426: 'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5427: 'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5428: 'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5429: 'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5430: 'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5431: 'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5432: 'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5433: 'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5434: //LongWord', 'Cardinal
5435: TBitList', 'Word
5436: Func UMul( const Num1, Num2 : LongWord) : LongWord
5437: Func UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5438: Func UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5439: Func UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5440: Func SameFloat( const X1, X2 : Extended) : Bool
5441: Func FloatIsZero( const X : Extended) : Bool
5442: Func FloatIsPositive( const X : Extended) : Bool
5443: Func FloatIsNegative( const X : Extended) : Bool
5444: Proc IncLim( var B : Byte; const Limit : Byte)
5445: Proc IncLimSI( var B : ShortInt; const Limit : ShortInt)
5446: Proc IncLimW( var B : Word; const Limit : Word)
5447: Proc IncLimI( var B : Int; const Limit : Int)
5448: Proc IncLimL( var B : LongInt; const Limit : LongInt)
5449: Proc DecLim( var B : Byte; const Limit : Byte)
5450: Proc DecLimSI( var B : ShortInt; const Limit : ShortInt)
5451: Proc DecLimW( var B : Word; const Limit : Word)
5452: Proc DecLimI( var B : Int; const Limit : Int)
5453: Proc DecLimL( var B : LongInt; const Limit : LongInt)
5454: Func MaxB( const B1, B2 : Byte) : Byte
5455: Func MinB( const B1, B2 : Byte) : Byte
5456: Func MaxSI( const B1, B2 : ShortInt) : ShortInt
5457: Func MinSI( const B1, B2 : ShortInt) : ShortInt
5458: Func MaxW( const B1, B2 : Word) : Word
5459: Func MinW( const B1, B2 : Word) : Word
5460: Func esbMaxI( const B1, B2 : Int) : Int
5461: Func esbMinI( const B1, B2 : Int) : Int
5462: Func MaxL( const B1, B2 : LongInt) : LongInt
5463: Func MinL( const B1, B2 : LongInt) : LongInt
5464: Proc SwapB( var B1, B2 : Byte)
5465: Proc SwapSI( var B1, B2 : ShortInt)
5466: Proc SwapW( var B1, B2 : Word)
5467: Proc SwapI( var B1, B2 : SmallInt)
5468: Proc SwapL( var B1, B2 : LongInt)
5469: Proc SwapI32( var B1, B2 : Int)
5470: Proc SwapC( var B1, B2 : LongWord)
5471: Proc SwapInt64( var X, Y : Int64)
5472: Func esbSign( const B : LongInt) : ShortInt
5473: Func Max4Word( const X1, X2, X3, X4 : Word) : Word
5474: Func Min4Word( const X1, X2, X3, X4 : Word) : Word
5475: Func Max3Word( const X1, X2, X3 : Word) : Word
5476: Func Min3Word( const X1, X2, X3 : Word) : Word
5477: Func MaxBArray( const B : array of Byte) : Byte
5478: Func MaxWArray( const B : array of Word) : Word
5479: Func MaxSIArray( const B : array of ShortInt) : ShortInt
5480: Func MaxIArray( const B : array of Int) : Int
5481: Func MaxLArray( const B : array of LongInt) : LongInt
5482: Func MinBArray( const B : array of Byte) : Byte
5483: Func MinWArray( const B : array of Word) : Word
5484: Func MinSIArray( const B : array of ShortInt) : ShortInt
5485: Func MinIArray( const B : array of Int) : Int
5486: Func MinLArray( const B : array of LongInt) : LongInt
5487: Func SumBArray( const B : array of Byte) : Byte
5488: Func SumBArray2( const B : array of Byte) : Word
5489: Func SumSIArray( const B : array of ShortInt) : ShortInt
5490: Func SumSIArray2( const B : array of ShortInt) : Int
5491: Func SumWArray( const B : array of Word) : Word
5492: Func SumWArray2( const B : array of Word) : LongInt
5493: Func SumIArray( const B : array of Int) : Int
5494: Func SumLArray( const B : array of LongInt) : LongInt
5495: Func SumLWArray( const B : array of LongWord) : LongWord
5496: Func ESBDigits( const X : LongWord) : Byte
5497: Func BitsHighest( const X : LongWord) : Int
5498: Func ESBBitsNeeded( const X : LongWord) : Int
5499: Func esbGCD( const X, Y : LongWord) : LongWord
5500: Func esbLCM( const X, Y : LongInt) : Int64
5501: //Func esbLCM( const X, Y : LongInt) : LongInt
5502: Func RelativePrime( const X, Y : LongWord) : Bool
5503: Func Get87ControlWord : TBitList
5504: Proc Set87ControlWord( const CWord : TBitList)

```

```

5505: Proc SwapExt( var X, Y : Extended)
5506: Proc SwapDbl( var X, Y : Double)
5507: Proc SwapSing( var X, Y : Single)
5508: Func esbSgn( const X : Extended) : ShortInt
5509: Func Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5510: Func ExtMod( const X, Y : Extended) : Extended
5511: Func ExtRem( const X, Y : Extended) : Extended
5512: Func CompMOD( const X, Y : Comp) : Comp
5513: Proc Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5514: Proc XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5515: Func DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5516: Proc Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5517: Func MaxExt( const X, Y : Extended) : Extended
5518: Func MinExt( const X, Y : Extended) : Extended
5519: Func MaxEArray( const B : array of Extended) : Extended
5520: Func MinEArray( const B : array of Extended) : Extended
5521: Func MaxSArray( const B : array of Single) : Single
5522: Func MinSArray( const B : array of Single) : Single
5523: Func MaxCArray( const B : array of Comp) : Comp
5524: Func MinCArray( const B : array of Comp) : Comp
5525: Func SumSArray( const B : array of Single) : Single
5526: Func SumEArray( const B : array of Extended) : Extended
5527: Func SumSqEArray( const B : array of Extended) : Extended
5528: Func SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5529: Func SumXYEArray( const X, Y : array of Extended) : Extended
5530: Func SumCArray( const B : array of Comp) : Comp
5531: Func FactorialX( A : LongWord) : Extended
5532: Func PermutationX( N, R : LongWord) : Extended
5533: Func esbBinomialCoeff( N, R : LongWord) : Extended
5534: Func IsPositiveEArray( const X : array of Extended) : Bool
5535: Func esbGeometricMean( const X : array of Extended) : Extended
5536: Func esbHarmonicMean( const X : array of Extended) : Extended
5537: Func ESbMean( const X : array of Extended) : Extended
5538: Func esbSampleVariance( const X : array of Extended) : Extended
5539: Func esbPopulationVariance( const X : array of Extended) : Extended
5540: Proc esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5541: Proc esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5542: Func GetMedian( const SortedX : array of Extended) : Extended
5543: Func GetMode( const SortedX : array of Extended; var Mode : Extended) : Bool
5544: Proc GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5545: Func ESbMagnitude( const X : Extended) : Int
5546: Func ESbTan( Angle : Extended) : Extended
5547: Func ESbCot( Angle : Extended) : Extended
5548: Func ESbCosec( const Angle : Extended) : Extended
5549: Func ESbSec( const Angle : Extended) : Extended
5550: Func ESbArcTan( X, Y : Extended) : Extended
5551: Proc ESbSinCos( Angle : Extended; var SinX, CosX : Extended)
5552: Func ESbArcCos( const X : Extended) : Extended
5553: Func ESbArcSin( const X : Extended) : Extended
5554: Func ESbArcSec( const X : Extended) : Extended
5555: Func ESbArcCosec( const X : Extended) : Extended
5556: Func ESbLog10( const X : Extended) : Extended
5557: Func ESbLog2( const X : Extended) : Extended
5558: Func ESbLogBase( const X, Base : Extended) : Extended
5559: Func Pow2( const X : Extended) : Extended
5560: Func IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5561: Func ESbIntPower( const X : Extended; const N : LongInt) : Extended
5562: Func XtoY( const X, Y : Extended) : Extended
5563: Func esbTenToY( const Y : Extended) : Extended
5564: Func esbTwoToY( const Y : Extended) : Extended
5565: Func LogXtoBaseY( const X, Y : Extended) : Extended
5566: Func esbISqrt( const I : LongWord) : LongWord
5567: Func ILog2( const I : LongWord) : LongWord
5568: Func IGreatestPowerOf2( const N : LongWord) : LongWord
5569: Func ESbArCosh(X:Extended):Extended
5570: Func ESbArSinh(X:Extended):Extended
5571: Func ESbArTanh(X:Extended):Extended
5572: Func ESbCosh(X:Extended):Extended
5573: Func ESbSinh(X:Extended):Extended
5574: Func ESbTanh(X:Extended):Extended
5575: Func InverseGamma(const X: Extended) : Extended
5576: Func esbGamma(const X : Extended) : Extended
5577: Func esbLnGamma(const X: Extended) : Extended
5578: Func esbBeta(const X, Y: Extended) : Extended
5579: Func IncompleteBeta(X :Extended; P, Q: Extended) : Extended
5580: end;
5581:
5582: *****Int Huge Cardinal Utils*****
5583: Func Add_uint64_WithCarry( x, y : uint64; var Carry :Bool) : uint64
5584: Func Add_uint32_WithCarry( x, y : uint32; var Carry :Bool) : uint32
5585: Func Subtract_uint64_WithBorrow( x, y : uint64; var Borrow :Bool) : uint64
5586: Func Subtract_uint32_WithBorrow( x, y : uint32; var Borrow :Bool) : uint32
5587: Func BitCount_8( Value : byte) : Int
5588: Func BitCount_16( Value : uint16) : Int
5589: Func BitCount_32( Value : uint32) : Int
5590: Func BitCount_64( Value : uint64) : Int
5591: Func CountSetBits_64( Value : uint64) : Int TPrimalityTestNoticeProc',
5592: Proc ( CountPrimalityTests : Int)
5593: Func gcdhuge( a, b : THugeCardinal) : THugeCardinal

```



```

5594: Func lcmhuge( a, b : THugeCardinal) : THugeCardinal
5595: Func isCoPrime( a, b : THugeCardinal) : Bool
5596: Func isProbablyPrime(p:THugeCardinal;OnProgress:TProgress;var wasAborted:bool):boolean
5597: Func hasSmallFactor( p : THugeCardinal) :Bool
5598: Func GeneratePrime(NumBits:
Int;OnProgress:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;PassCount:Int;Pool1:TMemoryStreamPool;var
Prime:THugeCardinal;var NumbersTested:Int):bool
5599: Func Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5600: Const('StandardExponent','LongInt'( 65537);
5601: Proc Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:Int;Fixed_e:uint64;var N,e,d,
Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPool;var
Numbers
5602: Func Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal:Bool')
5603: Func AnsiBytesOf(const S:Str): TBytes;
5604: Func GetBytes(value:Str): TBytes;
5605: Func GetBytesInt(value: Smallint): TBytes;
5606: Func HugeToBase10(const AHuge: THugeCardinal):Str;
5607:
5608: (*-----*)
5609: Proc SIRegister_uTPLb_StreamUtils(CL: TPSPascalCompiler);
5610: begin
5611: SIRegister_TDesalinationWriteStream(CL);
5612: // CL.AddTypeS('TByteDynArray', 'array of Byte
5613: Func CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
5614: Proc CopyMemoryStream( Source, Destination : TMemoryStream)
5615: Proc BurnMemoryStream( Destructo : TMemoryStream)
5616: Proc BurnMemory( var Buff, BuffLen : integer)
5617: Proc BurnMemoryString( var Buff:Str; BuffLen : integer)
5618: Proc BurnMemoryInteger( var Buff: Integer; BuffLen : integer)
5619: Proc BurnMemoryString2( Buff:Str; BuffLen : integer)
5620: Proc BurnMemoryInteger2( Buff: Integer; BuffLen : integer)
5621: (Proc BurnMemoryByteArray( var array of Byte; BuffLen : integer)
5622: Proc BurnMemoryByteArray2( var buff: TByteDynArray; BuffLen : integer)
5623: Proc BurnMemoryByteArray3( buff: TByteDynArray; BuffLen : integer)
5624: Proc ZeroFillStream( Stream : TMemoryStream)
5625: Proc RandomFillStream( Stream : TMemoryStream)
5626: Proc XOR_Streams2( Dest, Srce : TMemoryStream)
5627: Proc XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
5628: Func CompareMemoryStreams( S1, S2 : TMemoryStream) :Bool
5629: Func Stream_to_Base64( Source : TStream) : Ansistr
5630: Func Stream_to_Base642( Source : TStream) : Ansistr
5631: Proc Base64_to_stream2( const Base64 : Ansistr; Destin : TStream)
5632: Func Stream_to_Ansistr( Source : TStream) : Ansistr
5633: Func StreamtoAnsistr( Source : TStream) : Ansistr
5634: Func StreamToString( Source : TStream) :Str
5635: Func StreamToString2( Source : TStream) :Str
5636: Func StreamToString3( Source : TStream) :Str
5637: Func StreamToString4( Source : TStream) :Str
5638: Ansistr_to_stream( const Value : Ansistr; Destin : TStream)
5639: Func CompareFiles(const FN1,FN2:str;Breathe:TNotifyEvent;BreathingSender:TObject):bool;
5640: Func FileSize( const FileName :Str) : int64
5641: Func Stream_to_decimalbytes( Source : TStream) :Str
5642: Func StreamToDecimalbytes( Source : TStream) :Str
5643: Func StreamtoOrd( Source : TStream) :Str
5644: Func StreamToByte( Source : TStream) :Str
5645: Func DisplayStream( Stream : TStream) :Str
5646: Proc Base64_to_streamBytes( const Base64: TBytes; Destin: TStream);
5647: Func Stream_to_Base64Bytes(ASource: TStream; const ATransform: TBytes): TBytes;
5648: Func Stream_To_Hex(ASource: TStream): Ansistr;
5649: // from unit uPSI_uTPLb_PointerArithmetic;
5650: Proc ClearMemory( Stream : TMemoryStream; Offset, CountBytes : integer)
5651: Func ReadMem(Source:TStream;Destin:TMemoryStream;DestinOffset,CountBytes:integer):int);
5652: Func WriteMem(Source:TMemoryStream;SourceOffset:int;Destin:TStream;CountBytes:int):int;
5653: end;
5654:
5655: Proc SIRegister_EwbCoreTools(CL: TPSPascalCompiler);
5656: begin
5657: //CL.AddDelphiFunction('Func IsWinXPSP2OrLater( ) :Bool
5658: Func ColorToHTML2( const Color : TColor) :Str
5659: //Func WideStringToLPOLESTR( const Source : WideString) : POLEStr
5660: //Func XPath4Node( node : IHTMLElement) :Str
5661: //Func TaskAllocWideString( const S :Str) : PWChar
5662: Func AnsiIndexStr2( const AText :Str; const AValues : array of string) : Integer
5663: Func GetPos( const SubSt, Text :Str; StartPos : Integer) : Integer
5664: Func _CharPos( const C : Char; const S :Str) : Integer
5665: Func CutString(var Text:str; const Delimiter:str; const Remove:Boolean) :Str
5666: Proc FormatPath( Path :Str)
5667: Func GetWinText( WinHandle : THandle) :Str
5668: Func GetWinClass( Handle : Hwnd) : WideString
5669: Func GetParentWinByClass( ChildHandle : HWND; const ClassName :Str) : HWND
5670: Func DirectoryExists2( const Directory :Str) :Bool
5671: //CL.AddDelphiFunction('Func VarSupports(const V:Variant;const IID:TGUID; out Intf): Bool);
5672: Func CharInSet3( C : Char; const CharSet : TSysCharSet) :Bool
5673: Func AddBackslash2( const S :Str) :Str
5674: CL.AddConstantN('WM_SETWBFOCUS','LongWord').SetUInt( $0400 + $44);
5675: end;
5676:
5677: Proc SIRegister_EwbUrl(CL: TPSPascalCompiler);
5678: begin

```

```

5679: 'TEMP_SIZE_EWB','LongInt').SetInt( 1024);
5680: 'MAX_BUFFER_EWB','LongInt').SetInt( 256);
5681: 'WebDelim_EWB','String').SetString( '/');
5682: 'ProtocolDelim_EWB','String').SetString( '://');
5683: 'QueryDelim_EWB','String').SetString( '?');
5684: 'BookmarkDelim_EWB','String').SetString( '#');
5685: 'EqualDelim_EWB','String').SetString( '=');
5686: 'DriveDelim_EWB','String').SetString( ':');
5687: CL.AddTypeS('TQueryOption_EWB', 'ULONG');
5688: TOnError_EWB,'Proc (Sender:TObject;ErrorCode:integer;ErrMessage:str);
5689: SIRegister_TUrl(CL);
5690: end;
5691:
5692: Func IS_INTRESOURCE(lpszType: PChar): BOOL; //as of windows.pas
5693: begin
5694:   Result:= ULONG_PTR(lpszType) shr 16 = 0;
5695: end;
5696:
5697: Func ResourceNameToString(lpszName: PChar):Str;
5698: Func ResourceTypeToString(lpszType: PChar):Str;
5699: Proc storeRCDataResourceToFile(aresname, afilename:Str);
5700: Func InternalFindNextUrlToDownload(aNode: TPageNotYetDownloadedBinTreeNode; '+
5701:   allowDeepLevel: Integer): TPageNotYetDownloadedBinTreeNode;'); //ALWebSpider
5702:
5703: Proc SIRegister_xrtl_math_Int(CL: TPSPascalCompiler);
5704: begin
5705:   AddTypeS('TXRTLInt', 'array of Int
5706:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5707:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5708:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5709:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5710:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5711:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5712:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5713:   'BitsPerByte','LongInt'( 8);
5714:   BitsPerDigit','LongInt'( 32);
5715:   SignBitMask','LongWord( $80000000);
5716:   Func XRTLAdjustBits( const ABits : Int) : Int
5717:   Func XRTLLength( const AInt : TXRTLInt) : Int
5718:   Func XRTLDataBits( const AInt : TXRTLInt) : Int
5719:   Proc XRTLBitPosition( const BitIndex : Int; var Index, Mask : Int)
5720:   Proc XRTLBitSet( var AInt : TXRTLInt; const BitIndex : Int)
5721:   Proc XRTLBitReset( var AInt : TXRTLInt; const BitIndex : Int)
5722:   Func XRTLBitGet( const AInt : TXRTLInt; const BitIndex : Int) : Int
5723:   Func XRTLBitGetBool( const AInt : TXRTLInt; const BitIndex : Int) :Bool
5724:   Func XRTLExtend(const AInt:TXRTLInt;ADataBits:Int;Sign:Int;var AResult:TXRTLInt):Int;
5725:   Func XRTLZeroExtend(const AInt:TXRTLInt;ADataBits:Int; var AResult:TXRTLInt):Int;
5726:   Func XRTLSignExtend(const AInt:TXRTLInt; ADataBits:Int;var AResult:TXRTLInt):Int;
5727:   Func XRTLSignStrip(const AInt:TXRTLInt;var AResult:TXRTLInt;const AMinDataBits:Int):Int;
5728:   Proc XRTLNot( const AInt : TXRTLInt; var AResult : TXRTLInt)
5729:   Proc XRTLOr( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5730:   Proc XRTLAnd( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5731:   Proc XRTLXor( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5732:   Func XRTLSign( const AInt : TXRTLInt) : Int
5733:   Proc XRTLZero( var AInt : TXRTLInt)
5734:   Proc XRTLOne( var AInt : TXRTLInt)
5735:   Proc XRTLMOne( var AInt : TXRTLInt)
5736:   Proc XRTLTwo( var AInt : TXRTLInt)
5737:   Func XRTLNeg( const AInt : TXRTLInt; var AResult : TXRTLInt) : Int
5738:   Func XRTLAbs( const AInt : TXRTLInt; var AResult : TXRTLInt) : Int
5739:   Proc XRTLFullSum( const A, B, C : Int; var Sum, Carry : Int)
5740:   Func XRTLAdd( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int;
5741:   Func XRTLAdd1(const AInt1:TXRTLInt;const AInt2:Int64;var AResult:TXRTLInt):Int;
5742:   Func XRTLSub( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int;
5743:   Func XRTLSub1(const AInt1:TXRTLInt;const AInt2:Int64;var AResult:TXRTLInt):Int;
5744:   Func XRTLCompare( const AInt1, AInt2 : TXRTLInt) : Int;
5745:   Func XRTLCompare1( const AInt1 : TXRTLInt; const AInt2 : Int64) : Int;
5746:   Func XRTLUmul( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int
5747:   Func XRTLMulAdd(const AInt1,AInt2,AInt3:TXRTLInt; var AResult:TXRTLInt):Int
5748:   Func XRTLMul( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt) : Int
5749:   Proc XRTLDivMod( const AInt1, AInt2 : TXRTLInt; var QResult, RResult : TXRTLInt)
5750:   Proc XRTLSqr( const AInt : TXRTLInt; var AResult : TXRTLInt)
5751:   Proc XRTLSqrt( const AInt : TXRTLInt; var AResult : TXRTLInt)
5752:   Proc XRTLRoot( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5753:   Proc XRTLRootApprox(const AInt1,AInt2:TXRTLInt;var ALowApproxResult,AHighApproxResult:TXRTLInt)
5754:   Proc XRTLURootApprox(const AInt1,AInt2:TXRTLInt;var ALowApproxResult,AHighApproxResult:TXRTLInt);
5755:   Proc XRTLExp( const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5756:   Proc XRTLExpMod( const AInt1, AInt2, AInt3 : TXRTLInt;var AResult: TXRTLInt)
5757:   Proc XRTLSLBL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5758:   Proc XRTLSABL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5759:   Proc XRTLRCBL(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5760:   Proc XRTLSLDL(const AInt:TXRTLInt;const DigitCount:Int; var AResult:TXRTLInt)
5761:   Proc XRTLSADL(const AInt: TXRTLInt; const DigitCount:Int;var AResult: TXRTLInt)
5762:   Proc XRTLRCDL(const AInt:TXRTLInt; const DigitCount:Int; var AResult: TXRTLInt)
5763:   Proc XRTLSLBR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5764:   Proc XRTLSABR(const AInt: TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5765:   Proc XRTLRCBR(const AInt:TXRTLInt; const BitCount:Int; var AResult: TXRTLInt)
5766:   Proc XRTLSLDR(const AInt:TXRTLInt; const DigitCount:Int; var AResult: TXRTLInt)
5767:   Proc XRTLSADR(const AInt: TXRTLInt; const DigitCount:Int;var AResult: TXRTLInt)

```

```

5768: Proc XRTLRCDDR(const AInt: TXRTLInt;const DigitCount:Int;var AResult: TXRTLInt)
5769: Func XRTLToHex(const AInt : TXRTLInt; Digits : Int) :Str
5770: Func XRTLToBin(const AInt : TXRTLInt; Digits : Int) :Str
5771: Func XRTLToString(const AInt : TXRTLInt; Radix : Int; Digits : Int) :Str
5772: Proc XRTLFromHex(const Value :Str; var AResult : TXRTLInt)
5773: Proc XRTLFromBin(const Value :Str; var AResult : TXRTLInt)
5774: Proc XRTLFromString(const Value :Str; var AResult : TXRTLInt; Radix : Int)
5775: Proc XRTLAssign(const AInt : TXRTLInt; var AResult : TXRTLInt);
5776: Proc XRTLAssign1(const Value : Int; var AResult : TXRTLInt);
5777: Proc XRTLAssign2(const Value : Int64; var AResult : TXRTLInt);
5778: Proc XRTLAppend(const AInt, AHigh : TXRTLInt; var AResult : TXRTLInt)
5779: Proc XRTLSplit(const AInt: TXRTLInt; var ALow,AHigh: TXRTLInt;LowDigits: Int)
5780: Func XRTLGetMSBitIndex(const AInt : TXRTLInt) : Int
5781: Proc XRTLMinMax(const AInt1, AInt2:TXRTLInt;var AMinResult,AMaxResult: TXRTLInt)
5782: Proc XRTLMin(const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt);
5783: Proc XRTLMin1(const AInt1:TXRTLInt;const AInt2:Int;var AResult : TXRTLInt);
5784: Proc XRTLMax(const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt);
5785: Proc XRTLMax1(const AInt1:TXRTLInt; const AInt2:Int; var AResult:TXRTLInt);
5786: Proc XRTLGCD(const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5787: Proc XRTLSwap(var AInt1, AInt2 : TXRTLInt)
5788: Proc XRTLFactorial(const AInt : TXRTLInt; var AResult : TXRTLInt)
5789: Proc XRTLFactorialMod(const AInt1, AInt2 : TXRTLInt; var AResult : TXRTLInt)
5790: end;
5791:
5792: Proc SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5793: begin
5794: Func JvXPMMethodsEqual(const Method1, Method2 : TMethod) :Bool
5795: Proc JvXPDrawLine(const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Int)
5796: Proc JvXPCreateGradientRect(const AWidth, AHeight : Int; const StartColor, EndColor : TColor;const
Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Bool;var Bitmap:TBitmap);
5797: Proc JvXPAdjustBoundRect(const BorderWidth:Byte;const ShowBoundLines:Boolean; const
BoundLines:TJvXPBoundLines; var Rect: TRect)
5798: Proc JvXPDrawBoundLines(const ACanvas:TCanvas;const BoundLns:TJvXPBoundLns;const AColor:TColor;Rect:TRect;
5799: Proc JvXPConvertToGray2(Bitmap : TBitmap)
5800: Proc JvXPRenderText(const AParent:TControl;const ACanvas:TCanvas;ACaption:TCaption;const
AFont:TFont;const AEnabled,AShowAccelChar:Bool;var ARect:TRect;AFlags:Int)
5801: Proc JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const Swapped:Bool;
5802: Proc JvXPColorizeBitmap(Bitmap : TBitmap; const AColor : TColor)
5803: Proc JvXPSetDrawFlags(const AAlignment:TAlignment;const AWordWrap:Bool;var Flags: Int)
5804: 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)
5805: end;
5806:
5807: Proc SIRegister_uwinstr(CL: TPSPascalCompiler);
5808: begin
5809: Func StrDec(S :Str) :Str
5810: Func uIsNumeric(var S :Str; var X : Float) :Bool
5811: Func ReadNumFromEdit(Edit : TEdit) : Float
5812: Proc WriteNumToFile(var F : Text; X : Float)
5813: end;
5814:
5815: Proc SIRegister_utexplot(CL: TPSPascalCompiler);
5816: begin
5817: Func TeX_InitGraphics(FileName:str;PgWidth,PgHeight:Int; Header:Boolean):Boolean
5818: Proc TeX_SetWindow(X1, X2, Y1, Y2 : Int; GraphBorder :Bool)
5819: Proc TeX_LeaveGraphics( Footer :Bool)
5820: Proc TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float)
5821: Proc TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float)
5822: Proc TeX_SetGraphTitle( Title :Str)
5823: Proc TeX_SetOxTitle( Title :Str)
5824: Proc TeX_SetOyTitle( Title :Str)
5825: Proc TeX_PlotOxAxis
5826: Proc TeX_PlotOyAxis
5827: Proc TeX_PlotGrid(Grid : TGrid)
5828: Proc TeX_WriteGraphTitle
5829: Func TeX_SetMaxCurv(NCurv : Byte) :Bool
5830: Proc TeX_SetPointParam(CurvIndex, Symbol, Size : Int)
5831: Proc TeX_SetLineParam(CurvIndex, Style : Int; Width : Float; Smooth :Bool)
5832: Proc TeX_SetCurvLegend(CurvIndex : Int; Legend :Str)
5833: Proc TeX_SetCurvStep(CurvIndex, Step : Int)
5834: Proc TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Int)
5835: Proc TeX_PlotCurveWithErrorBars(X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Int)
5836: Proc TeX_PlotFunc(Func : TFunc; X1, X2 : Float; Npt : Int; CurvIndex : Int)
5837: Proc TeX_WriteLegend(NCurv : Int; ShowPoints, ShowLines :Bool)
5838: Proc TeX_ConRec(Nx, Ny, Nc : Int; X, Y, Z : TVector; F : TMatrix)
5839: Func Xcm(X : Float) : Float
5840: Func Ycm(Y : Float) : Float
5841: end;
5842:
5843: *-----*)
5844: Proc SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5845: begin
5846: TConstArray', 'array of TVarRec
5847: Func CopyVarRec(const Item : TVarRec) : TVarRec
5848: Func CreateConstArray(const Elements : array of const) : TConstArray
5849: Proc FinalizeVarRec(var Item : TVarRec)
5850: Proc FinalizeConstArray(var Arr : TConstArray)
5851: end;

```

```

5852:
5853: Proc SRegister_StStrS(CL: TPSPascalCompiler);
5854: begin
5855:   Func HexBS( B : Byte) : ShortString
5856:   Func HexWS( W : Word) : ShortString
5857:   Func HexLS( L : LongInt) : ShortString
5858:   Func HexPtrS( P : Pointer) : ShortString
5859:   Func BinaryBS( B : Byte) : ShortString
5860:   Func BinaryWS( W : Word) : ShortString
5861:   Func BinaryLS( L : LongInt) : ShortString
5862:   Func OctalBS( B : Byte) : ShortString
5863:   Func OctalWS( W : Word) : ShortString
5864:   Func OctalLS( L : LongInt) : ShortString
5865:   Func Str2Int16S( const S : ShortString; var I : SmallInt) : Bool
5866:   Func Str2WordS( const S : ShortString; var I : Word) : Bool
5867:   Func Str2LongS( const S : ShortString; var I : LongInt) : Bool
5868:   Func Str2RealS( const S : ShortString; var R : Double) : Bool
5869:   Func Str2RealS( const S : ShortString; var R : Real) : Bool
5870:   Func Str2ExtS( const S : ShortString; var R : Extended) : Bool
5871:   Func Long2StrS( L : LongInt) : ShortString
5872:   Func Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5873:   Func Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5874:   Func ValPrepS( const S : ShortString) : ShortString
5875:   Func CharStrS( C : AnsiChar; Len : Card) : ShortString
5876:   Func PadChS( const S : ShortString; C : AnsiChar; Len : Card) : ShortString
5877:   Func PadS( const S : ShortString; Len : Card) : ShortString
5878:   Func LeftPadChS( const S : ShortString; C : AnsiChar; Len : Card) : ShortString
5879:   Func LeftPadS( const S : ShortString; Len : Card) : ShortString
5880:   Func TrimLeadS( const S : ShortString) : ShortString
5881:   Func TrimTrailS( const S : ShortString) : ShortString
5882:   Func TrimS( const S : ShortString) : ShortString
5883:   Func TrimSpacesS( const S : ShortString) : ShortString
5884:   Func CenterChS( const S : ShortString; C : AnsiChar; Len : Card) : ShortString
5885:   Func CenterS( const S : ShortString; Len : Card) : ShortString
5886:   Func EntabS( const S : ShortString; TabSize : Byte) : ShortString
5887:   Func DetabS( const S : ShortString; TabSize : Byte) : ShortString
5888:   Func ScrambleS( const S, Key : ShortString) : ShortString
5889:   Func SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5890:   Func FilterS( const S, Filters : ShortString) : ShortString
5891:   Func CharExistsS( const S : ShortString; C : AnsiChar) : Bool
5892:   Func CharCountS( const S : ShortString; C : AnsiChar) : Byte
5893:   Func WordCountS( const S, WordDelims : ShortString) : Card
5894:   Func WordPositions( N:Card; const S, WordDelims:ShortString; var Pos:Card) : Bool
5895:   Func ExtractWords( N :Card; const S, WordDelims : ShortString) : ShortString
5896:   Func AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Card
5897:   Func AsciiPositionS( N:Card; const S, WordDelims:ShortStr; Quote:AnsiChar; var Pos:Card) : Bool
5898:   Func ExtractAsciiS( N:Card; const S, WordDelims:ShortString; Quote:AnsiChar) : ShortString
5899:   Proc WordWrapS( const InStr:ShortString; var OutStr,Overlap:ShortString; Margin:Card; PadToMargin:Bool)
5900:   Func CompStringS( const S1, S2 : ShortString) : Int
5901:   Func CompUCStringS( const S1, S2 : ShortString) : Int
5902:   Func SoundexS( const S : ShortString) : ShortString
5903:   Func MakeLetterSetS( const S : ShortString) : Longint
5904:   Proc BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5905:   Func BMSearchS( var Buffer, BufLength:Card; var BT:BTable; const MatchString:ShortString; var Pos:Card) : Bool;
5906:   Func BMSearchUCS( var Buffer, BufLength:Card; var BT:BTable; const MatchStr:ShortStr; var Pos:Card) : Bool;
5907:   Func DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5908:   Func ForceExtensionS( const Name, Ext : ShortString) : ShortString
5909:   Func JustFilenameS( const PathName : ShortString) : ShortString
5910:   Func JustNameS( const PathName : ShortString) : ShortString
5911:   Func JustExtensionS( const Name : ShortString) : ShortString
5912:   Func JustPathNameS( const PathName : ShortString) : ShortString
5913:   Func AddBackSlashS( const DirName : ShortString) : ShortString
5914:   Func CleanPathNameS( const PathName : ShortString) : ShortString
5915:   Func HasExtensionS( const Name : ShortString; var DotPos :Card) : Bool
5916:   Func CommaizeS( L : LongInt) : ShortString
5917:   Func CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5918:   Func FloatFormS( const Mask:ShortStr; R:TstFloat; const LtCurr, RtCurr:SString; Sep, DecPt:Char) : ShortStr;
5919:   Func LongIntFormS( const Mask:ShortString; L:LongInt; const LtCurr,
RtCurr:ShortString; Sep:AnsiChar) : ShortString;
5920:   Func StrChPosS( const P : ShortString; C : AnsiChar; var Pos :Card) : Bool
5921:   Func StrStPosS( const P, S : ShortString; var Pos :Card) : Bool
5922:   Func StrStCopyS( const S : ShortString; Pos, Count :Card) : ShortString
5923:   Func StrChInsertS( const S : ShortString; C : AnsiChar; Pos :Card) : ShortString
5924:   Func StrStInsertS( const S1, S2 : ShortString; Pos :Card) : ShortString
5925:   Func StrChDeleteS( const S : ShortString; Pos :Card) : ShortString
5926:   Func StrStDeleteS( const S : ShortString; Pos, Count :Card) : ShortString
5927:   Func ContainsOnlyS( const S, Chars : ShortString; var BadPos :Card) : Bool
5928:   Func ContainsOtherThanS( const S, Chars : ShortString; var BadPos :Card) : Bool
5929:   Func CopyLeftS( const S : ShortString; Len :Card) : ShortString
5930:   Func CopyMidS( const S : ShortString; First, Len :Card) : ShortString
5931:   Func CopyRightS( const S : ShortString; First :Card) : ShortString
5932:   Func CopyRightAbsS( const S : ShortString; NumChars :Card) : ShortString
5933:   Func CopyFromNthWords( const S, WordDelims:str; const AWord:str; N:Card; var SubString:ShortString) : Bool;
5934:   Func DeleteFromNthWords( const S, WordDelims:str; AWord:ShortString; N:Card; var SubStr:ShortString) : Bool;
5935:   Func CopyFromToWordsS( const S, WordDelims, Word1, Word2:ShortString; N1, N2:Card; var
SubString:ShortString) : Bool;
5936:   Func DeleteFromToWordsS( const S, WordDelims, Wrd1, Wrd2:ShortString; N1, N2:Card; var
SubString:ShortString) : Bool;
5937:   Func CopyWithinS( const S, Delimiter : ShortString; Strip :Bool) : ShortString

```

```

5938: Func DeleteWithinS( const S, Delimiter : ShortString ) : ShortString
5939: Func ExtractTokensS( const S, Delims: ShortString; QuoteChar: AnsiChar; AllowNulls: Bool; Tokens: TStrings ) : Card
5940: Func IsChAlphaS( C : Char ) : Bool
5941: Func IsChNumericS( C : Char; const Numbers : ShortString ) : Bool
5942: Func IsChAlphaNumericS( C : Char; const Numbers : ShortString ) : Bool
5943: Func IsStrAlphaS( const S : Shortstring ) : Bool
5944: Func IsStrNumericS( const S, Numbers : ShortString ) : Bool
5945: Func IsStrAlphaNumericS( const S, Numbers : ShortString ) : Bool
5946: Func LastWordS( const S, WordDelims, AWord: ShortString; var Position : Card ) : Bool
5947: Func LastWordAbsS( const S, WordDelims : ShortString; var Position : Card ) : Bool
5948: Func LastStringS( const S, AString : ShortString; var Position : Card ) : Bool
5949: Func LeftTrimCharsS( const S, Chars : ShortString ) : ShortString
5950: Func KeepCharsS( const S, Chars : ShortString ) : ShortString
5951: Func RepeatStringS( const RepeatString: ShortStr; var Repetitions: Card; MaxLen: Card ) : ShortStr;
5952: Func ReplaceStringS( const S, OldStr, NewStr: ShortString; N: Card; var Replacements: Card ) : ShortString;
5953: Func ReplaceStringAllS( const S, OldString, NewString: ShortString; var Replacements: Card ) : ShortString;
5954: Func ReplaceWordS( const S, WordDelims, OldWord, NewW: SString; N: Card; var Replacements: Card ) : ShortString
5955: Func ReplaceWordAllS( const S, WordDelims, OldWord, NewWord: ShortString; var Replacements: Card ) : ShortString
5956: Func RightTrimCharsS( const S, Chars : ShortString ) : ShortString
5957: Func StrWithinS( const S, SearchStr: ShortString; Start: Card; var Position: Card ) : bool
5958: Func TrimCharsS( const S, Chars : ShortString ) : ShortString
5959: Func WordPosS( const S, WordDelims, AWord: ShortString; N: Card; var Position: Card ) : Bool
5960: end;
5961:
5962: *****unit uPSI_StUtils; from Systools4*****
5963: Func SignL( L : LongInt ) : Int
5964: Func SignF( F : Extended ) : Int
5965: Func MinWord( A, B : Word ) : Word
5966: Func MidWord( W1, W2, W3 : Word ) : Word
5967: Func MaxWord( A, B : Word ) : Word
5968: Func MinLong( A, B : LongInt ) : LongInt
5969: Func MidLong( L1, L2, L3 : LongInt ) : LongInt
5970: Func MaxLong( A, B : LongInt ) : LongInt
5971: Func MinFloat( F1, F2 : Extended ) : Extended
5972: Func MidFloat( F1, F2, F3 : Extended ) : Extended
5973: Func MaxFloat( F1, F2 : Extended ) : Extended
5974: Func MakeInt16( H, L : Byte ) : SmallInt
5975: Func MakeWordS( H, L : Byte ) : Word
5976: Func SwapNibble( B : Byte ) : Byte
5977: Func SwapWord( L : LongInt ) : LongInt
5978: Proc SetFlag( var Flags : Word; FlagMask : Word )
5979: Proc ClearFlag( var Flags : Word; FlagMask : Word )
5980: Func FlagIsSet( Flags, FlagMask : Word ) : Bool
5981: Proc SetByteFlag( var Flags : Byte; FlagMask : Byte )
5982: Proc ClearByteFlag( var Flags : Byte; FlagMask : Byte )
5983: Func ByteFlagIsSet( Flags, FlagMask : Byte ) : Bool
5984: Proc SetLongFlag( var Flags : LongInt; FlagMask : LongInt )
5985: Proc ClearLongFlag( var Flags : LongInt; FlagMask : LongInt )
5986: Func LongFlagIsSet( Flags, FlagMask : LongInt ) : Bool
5987: Proc ExchangeBytes( var I, J : Byte )
5988: Proc ExchangeWords( var I, J : Word )
5989: Proc ExchangeLongInts( var I, J : LongInt )
5990: Proc ExchangeStructs( var I, J, Size : Card )
5991: Proc FillWord( var Dest, Count : Card; Filler : Word )
5992: Proc FillStruct( var Dest, Count : Card; var Filler, FillerSize : Card )
5993: Func AddWordToPtr( P : __Pointer; W : Word ) : __Pointer
5994: //*****uPSI_StFIN;*****
5995: Func AccruedInterestMaturity( Issue, Maturity: TStDate; Rate, Par: Extended; Basis: TStBasis ) : Extended
5996: Func AccruedInterestPeriodic( Issue, Settlement, Maturity: TStDate; Rate, Par: Extended; Frequency: TStFrequency;
Basis : TStBasis ) : Extended
5997: Func BondDuration( Settlement, Maturity: TStDate; Rate,
Yield: Ext; Frequency: TStFrequency; Basis: TStBasis ) : Extended;
5998: Func BondPrice( Settlement, Maturity: TStDate; Rate, Yield,
Redempt: Ext; Freq: TStFrequency; Basis: TStBasis ) : Extended
5999: Func CumulativeInterest( Rate: Extended; NPeriods: Int; PV: Extended; StartPeriod,
EndPeriod: Int; Frequency: TStFrequency; Timing: TStPaymentTime ) : Extended
6000: Func CumulativePrincipal( Rate: Extended; NPeriods: Int; PV: Extended; StartPeriod,
EndPeriod: Int; Frequency: TStFrequency; Timing: TStPaymentTime ) : Extended
6001: Func DayCount( Day1, Day2 : TStDate; Basis : TStBasis ) : LongInt
6002: Func DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Int ) : Extended
6003: Func DiscountRate( Settlement, Maturity: TStDate; Price, Redemption: Extended; Basis: TStBasis ) : Extended;
6004: Func DollarToDecimal( FracDollar : Extended; Fraction : Int ) : Extended
6005: Func DollarToDecimalText( DecDollar : Extended ) : Str
6006: Func DollarToFraction( DecDollar : Extended; Fraction : Int ) : Extended
6007: Func DollarToFractionStr( FracDollar : Extended; Fraction : Int ) : Str
6008: Func EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency ) : Extended
6009: Func FutureValueS( Rate: Extended; NPeriods: Int; Pmt, PV: Extended; Freq: TStFreq; Timing: TStPaymentTime ) : Extended;
6010: Func FutureValueSchedule( Principal : Extended; const Schedule : array of Double ) : Extended
6011: Func FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Int ) : Extended
6012: Func InterestRateS( NPeriods: Int; Pmt, PV,
FV: Extended; Freq: TStFrequency; Timing: TStPaymentTime; Guess: Extended ) : Extended;
6013: Func InternalRateOfReturn( const Values : array of Double; Guess : Extended ) : Extended
6014: Func InternalRateOfReturn16( const Values, NValues : Int; Guess : Extended ) : Extended
6015: Func IsCardValid( const S : Str ) : Bool
6016: Func ModifiedDuration( Settlement, Maturity: TStDate; Rate,
Yield: Ext; Freq: TStFrequency; Basis: TStBasis ) : Extended;
6017: Func ModifiedIRR( const Values: array of Double; FinanceRate, ReinvestRate: Extended ) : Extended
6018: Func ModifiedIRR16( const Values, NValues: Int; FinanceRate, ReinvestRate: Extended ) : Extended
6019: Func NetPresentValueS( Rate : Extended; const Values : array of Double ) : Extended

```



```

6020: Func NetPresentValue16( Rate : Extended; const Values, NValues : Int) : Extended
6021: Func NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
6022: Func NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
6023: Func NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
6024: Func Payment(Rate:Extended;NPeriods:Int;PV
    FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
6025: Func Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Int;
6026: Func PresentValueS(Rate:Extended;NPeriods:Int;Pmt,
    FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Extended
6027: Func ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
6028: Func RoundToDecimal( Value : Extended; Places : Int; Bankers :Bool) : Extended
6029: Func TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
6030: Func TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
6031: Func TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
6032: Func VariableDecliningBalance(Cost,Salvage:Extended;Life:Int;StartPeriod,EndPeriod,
    Factor:Extended;NoSwitch:bool):Extended
6033: Func YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Exted;Basis:TStBasis):Extended;
6034: Func YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
    Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
6035: Func YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
6036:
6037: //*****unit uPSI_StAstroP;
6038: Proc PlanetsPos( JD : Double; var PA : TStPlanetsArray)
6039: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****
6040: Func AveDev( const Data : array of Double) : Double
6041: Func AveDev16( const Data, NData : Int) : Double
6042: Func Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
6043: Func Correlation( const Data1, Data2 : array of Double) : Double
6044: Func Correlation16( const Data1, Data2, NData : Int) : Double
6045: Func Covariance( const Data1, Data2 : array of Double) : Double
6046: Func Covariance16( const Data1, Data2, NData : Int) : Double
6047: Func DevSq( const Data : array of Double) : Double
6048: Func DevSq16( const Data, NData : Int) : Double
6049: Proc Frequency(const Data:array of Double;const Bins:array of Double;var Counts:array of LongInt);
6050: //Proc Frequency16( const Data, NData : Int; const Bins, NBins : Int; var Counts)
6051: Func GeometricMeanS( const Data : array of Double) : Double
6052: Func GeometricMean16( const Data, NData : Int) : Double
6053: Func HarmonicMeanS( const Data : array of Double) : Double
6054: Func HarmonicMean16( const Data, NData : Int) : Double
6055: Func Largest( const Data : array of Double; K : Int) : Double
6056: Func Largest16( const Data, NData : Int; K : Int) : Double
6057: Func MedianS( const Data : array of Double) : Double
6058: Func Median16( const Data, NData : Int) : Double
6059: Func Mode( const Data : array of Double) : Double
6060: Func Model16( const Data, NData : Int) : Double
6061: Func Percentile( const Data : array of Double; K : Double) : Double
6062: Func Percentile16( const Data, NData : Int; K : Double) : Double
6063: Func PercentRank( const Data : array of Double; X : Double) : Double
6064: Func PercentRank16( const Data, NData : Int; X : Double) : Double
6065: Func Permutations( Number, NumberChosen : Int) : Extended
6066: Func Combinations( Number, NumberChosen : Int) : Extended
6067: Func FactorialS( N : Int) : Extended
6068: Func Rank( Number : Double; const Data : array of Double; Ascending: Bool) : Int
6069: Func Rank16( Number : Double; const Data, NData : Int; Ascending : Bool) : Int
6070: Func Smallest( const Data : array of Double; K : Int) : Double
6071: Func Smallest16( const Data, NData : Int; K : Int) : Double
6072: Func TrimMean( const Data : array of Double; Percent : Double) : Double
6073: Func TrimMean16( const Data, NData : Int; Percent : Double) : Double
6074: AddTypeS('TSTLinEst', 'record B0:Double; B1 : double; seB0 : double; seB'
6075: +1 : Double; R2: Double;sigma :Double;SSr:double;SSe: Double; F0:Double;df:Int;end
6076: Proc LinEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TSTLinEst;ErrorStats:Bool;
6077: Proc LogEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TSTLinEst;ErrorStats:Bool;
6078: Func Forecast(X: Double;const KnownY: array of Double;const KnownX:array of Double):Double
6079: Func ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double
6080: Func Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
6081: Func RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
6082: Func Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
6083: Func StandardErrorY(const KnownY: array of Double;const KnownX:array of Double):Double
6084: Func BetaDist( X, Alpha, Beta, A, B : Single) : Single
6085: Func BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
6086: Func BinomDist(Numbers,Trials: Int; ProbabilityS : Single;Cumulative:Bool):Single
6087: Func CritBinom( Trials : Int; ProbabilityS, Alpha : Single) : Int
6088: Func ChiDist( X : Single; DegreesFreedom: Int) : Single
6089: Func ChiInv( Probability : Single; DegreesFreedom: Int) : Single
6090: Func ExponDist( X, Lambda : Single; Cumulative :Bool) : Single
6091: Func FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Int) : Single
6092: Func FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Int) : Single
6093: Func LogNormDist( X, Mean, StandardDev : Single) : Single
6094: Func LogInv( Probability, Mean, StandardDev : Single) : Single
6095: Func NormDist( X, Mean, StandardDev : Single; Cumulative :Bool) : Single
6096: Func NormInv( Probability, Mean, StandardDev : Single) : Single
6097: Func NormSDist( Z : Single) : Single
6098: Func NormSInv( Probability : Single) : Single
6099: Func Poisson( X : Int; Mean : Single; Cumulative :Bool) : Single
6100: Func TDist( X : Single; DegreesFreedom : Int; TwoTails :Bool) : Single
6101: Func TInv( Probability : Single; DegreesFreedom : Int) : Single
6102: Func Erfc( X : Single) : Single
6103: Func GammaLn( X : Single) : Single
6104: Func LargestSort( const Data : array of Double; K : Int) : Double

```

```

6105: Func SmallestSort( const Data: array of double; K : Int) : Double
6106:
6107: Proc SIRegister_TstSorter(CL: TPSPascalCompiler);
6108:   Func OptimumHeapToUse( RecLen :Card; NumRecs : LongInt) : LongInt
6109:   Func MinimumHeapToUse( RecLen :Card) : LongInt
6110:   Func MergeInfo( MaxHeap : LongInt; RecLen :Card; NumRecs : LongInt) : TMergeInfo
6111:   Func DefaultMergeName( MergeNum : Int) :Str
6112:   Proc ArraySort( var A, RecLen, NumRecs :Card; Compare:TUntypedCompareFunc)
6113: Proc SIRegister_StAstro(CL: TPSPascalCompiler);
6114: Func AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
6115: Func FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
6116: Func SunPos( UT : TStDateTimeRec) : TStPosRec
6117: Func SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
6118: Func SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
6119: Func Twilight(LD:TStDate; Longitude,Latitude:Double;Twilight:TStTwilight):TStRiseSetRec
6120: Func LunarPhase( UT : TStDateTimeRec) : Double
6121: Func MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
6122: Func MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
6123: Func FirstQuarter( D : TStDate) : TStLunarRecord
6124: Func FullMoon( D : TStDate) : TStLunarRecord
6125: Func LastQuarter( D : TStDate) : TStLunarRecord
6126: Func NewMoon( D : TStDate) : TStLunarRecord
6127: Func NextFirstQuarter( D : TStDate) : TStDateTimeRec
6128: Func NextFullMoon( D : TStDate) : TStDateTimeRec
6129: Func NextLastQuarter( D : TStDate) : TStDateTimeRec
6130: Func NextNewMoon( D : TStDate) : TStDateTimeRec
6131: Func PrevFirstQuarter( D : TStDate) : TStDateTimeRec
6132: Func PrevFullMoon( D : TStDate) : TStDateTimeRec
6133: Func PrevLastQuarter( D : TStDate) : TStDateTimeRec
6134: Func PrevNewMoon( D : TStDate) : TStDateTimeRec
6135: Func SiderealTime( UT : TStDateTimeRec) : Double
6136: Func Solstice( Y, Epoch : Int; Summer :Bool) : TStDateTimeRec
6137: Func Equinox( Y, Epoch : Int; Vernal :Bool) : TStDateTimeRec
6138: Func SEaster( Y, Epoch : Int) : TStDate
6139: Func DateTimeToAJD( D : TDateTime) : Double
6140: Func HoursMin( RA : Double) : ShortString
6141: Func DegsMin( DC : Double) : ShortString
6142: Func AJDToDateTime( D : Double) : TDateTime
6143:
6144: Proc SIRegister_StDate(CL: TPSPascalCompiler);
6145: Func CurrentDate : TStDate
6146: Func StValidDate( Day, Month, Year, Epoch : Int) :Bool
6147: Func DMYtoStDate( Day, Month, Year, Epoch : Int) : TStDate
6148: Proc StDateToDMY( Julian : TStDate; var Day, Month, Year : Int)
6149: Func StIncDate( Julian : TStDate; Days, Months, Years : Int) : TStDate
6150: Func IncDateTrunc( Julian : TStDate; Months, Years : Int) : TStDate
6151: Proc StDateDiff( Dater1, Date2 : TStDate; var Days, Months, Years : Int)
6152: Func BondDateDiff( Dater1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
6153: Func WeekOfYear( Julian : TStDate) : Byte
6154: Func AstJulianDate( Julian : TStDate) : Double
6155: Func AstJulianDateToStDate( AstJulian : Double; Truncate :Bool) : TStDate
6156: Func AstJulianDatePrim( Year, Month, Date : Int; UT : TStTime) : Double
6157: Func StDayOfWeek( Julian : TStDate) : TStDayType
6158: Func DayOfWeekDMY( Day, Month, Year, Epoch : Int) : TStDayType
6159: Func StIsLeapYear( Year : Int) :Bool
6160: Func StDaysInMonth( Month : Int; Year, Epoch : Int) : Int
6161: Func ResolveEpoch( Year, Epoch : Int) : Int
6162: Func ValidTime( Hours, Minutes, Seconds : Int) :Bool
6163: Proc StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
6164: Func HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
6165: Func CurrentTime : TStTime
6166: Proc TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
6167: Func StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
6168: Func DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
6169: Func RoundToNearestHour( T : TStTime; Truncate :Bool) : TStTime
6170: Func RoundToNearestMinute( const T : TStTime; Truncate :Bool) : TStTime
6171: Proc DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
6172: Proc IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Int;Secs:LongInt)
6173: Func DateTimeToStDate(DT : TDateTime) : TStDate
6174: Func DateTimeToStTime(DT : TDateTime) : TStTime
6175: Func StDateToDateTime(D : TStDate) : TDateTime
6176: Func StTimeToDateTime(T : TStTime) : TDateTime
6177: Func Convert2ByteDate(TwoByteDate : Word) : TStDate
6178: Func Convert4ByteDate(FourByteDate : TStDate) : Word
6179:
6180: Proc SIRegister_StDateSt(CL: TPSPascalCompiler);
6181: Func DateStringHMStoAstJD( const Picture, DS :Str; H, M, S, Epoch : Int) : Double
6182: Func MonthToString( const Month : Int) :Str
6183: Func DateStringToStDate( const Picture, S :Str; Epoch : Int) : TStDate
6184: Func DateStringToDMY(const Picture,S:str; Epoch:Int; var D, M, Y : Int):Boolean
6185: Func StDateToDateString( const Picture:str;const Julian:TStDate;Pack:Bool):str
6186: Func DayOfWeekToString( const WeekDay : TStDayType) :Str
6187: Func DMYtoDateString(const Picture:str;Day,Month,Year,Epoch:Int;Pack:Boolean):str;
6188: Func CurrentDateString( const Picture :Str; Pack :Bool) :Str
6189: Func CurrentTimeString( const Picture :Str; Pack :Bool) :Str
6190: Func TimeStringToHMS( const Picture, St :Str; var H, M, S : Int) :Bool
6191: Func TimeStringToStTime( const Picture, S :Str) : TStTime
6192: Func StTimeToAmPmString(const Picture:str; const T:TStTime; Pack:Bool) :Str
6193: Func StTimeToTimeString(const Picture:Str; const T:TStTime;Pack :Bool) :Str

```

```

6194: Func DateStringIsBlank( const Picture, S :Str) :Bool
6195: Func InternationalDate( ForceCentury :Bool) :Str
6196: Func InternationalLongDate( ShortNames :Bool; ExcludeDOW :Bool) :Str
6197: Func InternationalTime( ShowSeconds :Bool) :Str
6198: Proc ResetInternationalInfo
6199:
6200: Proc SIRegister_StBase(CL: TPSPascalCompiler);
6201: Func DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) :Bool
6202: Func AnsiUpperCaseShort32( const S :Str) :Str
6203: Func AnsiCompareTextShort32( const S1, S2 :Str) : Int
6204: Func AnsiCompareStrShort32( const S1, S2 :Str) : Int
6205: Func HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
6206: Func HugeDecompressRLE(const InBuffer, InLen:Longint;var OutBuffer, OutLen:LongInt):Longint
6207: Proc HugeFillChar( var Dest, Count : Longint; Value : Byte)
6208: Proc HugeFillStruc( var Dest, Count : Longint; const Value, ValSize :Card)
6209: Func Upcase( C : AnsiChar) : AnsiChar
6210: Func LoCase( C : AnsiChar) : AnsiChar
6211: Func CompareLetterSets( Set1, Set2 : LongInt) :Card
6212: Func CompStruct( const S1, S2, Size :Card) : Int
6213: Func Search(const Buffer, BufLength:Card;const Match, MatLength:Card;var Pos:Cardi):Bool;
6214: Func StSearch(const Buff, BufLength:Card;const Match, MatLength:Card;var Pos:Cardi):Bool
6215: Func SearchUC(const Buff, BufLength:Card;const Match, MatLength:Card;var Pos:Cardi):Bool
6216: Func IsOrInheritsFrom( Root, Candidate : TClass) :Bool
6217: Proc RaiseContainerError( Code : longint)
6218: Proc RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6219: Func ProductOverflow( A, B : LongInt) :Bool
6220: Func StNewStr( S :Str) : PShortString
6221: Proc StDisposeStr( PS : PShortString)
6222: Proc ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : Int)
6223: Proc ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : Int)
6224: Proc ValWord( const S : ShortString; var Wd : word; var ErrorCode : Int)
6225: Proc RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6226: Proc RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6227: Proc RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info :Str)
6228:
6229: Proc SIRegister_usvd(CL: TPSPascalCompiler);
6230: begin
6231: Proc SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Int; S : TVector; V : TMatrix)
6232: Proc SV_SetZero( S : TVector; Lb, Ub : Int; Tol : Float)
6233: Proc SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Int;X:TVector);
6234: Proc SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Int; A : TMatrix)
6235: Proc RKF45(F:TDiffEqs;Negn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6236: end;
6237:
6238: //*****unit unit ; StMath Package of SysTools*****
6239: Func IntPowerS( Base : Extended; Exponent : Int) : Extended
6240: Func PowerS( Base, Exponent : Extended) : Extended
6241: Func StInvCos( X : Double) : Double
6242: Func StInvSin( Y : Double) : Double
6243: Func StInvTan2( X, Y : Double) : Double
6244: Func StTan( A : Double) : Double
6245: Proc DumpException; //unit StExpEng;
6246: Func HexifyBlock( var Buffer, BufferSize : Int):Str
6247:
6248: //*****unit unit ; StCRC Package of SysTools*****
6249: Func Adler32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
6250: Func Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6251: Func Adler32OfFile( FileName : Ansistr) : LongInt
6252: Func Crc16Prim( var Data, DataSize, CurCrc :Card) :Card
6253: Func Crc16OfStream( Stream : TStream; CurCrc :Card) :Card
6254: Func Crc16OfFile( FileName : Ansistr) :Card
6255: Func Crc32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
6256: Func Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6257: Func Crc32OfFile( FileName : Ansistr) : LongInt
6258: Func InternetSumPrim( var Data, DataSize, CurCrc :Card) :Card
6259: Func InternetSumOfStream( Stream : TStream; CurCrc :Card) :Card
6260: Func InternetSumOfFile( FileName : Ansistr) :Card
6261: Func Kermit16Prim( var Data, DataSize, CurCrc :Card) :Card
6262: Func Kermit16OfStream( Stream : TStream; CurCrc :Card) :Card
6263: Func Kermit16OfFile( FileName : Ansistr) :Card
6264:
6265: //*****unit unit ; StBCD Package of SysTools*****
6266: Func AddBcd( const B1, B2 : TbcdS) : TbcdS
6267: Func SubBcd( const B1, B2 : TbcdS) : TbcdS
6268: Func MulBcd( const B1, B2 : TbcdS) : TbcdS
6269: Func DivBcd( const B1, B2 : TbcdS) : TbcdS
6270: Func ModBcd( const B1, B2 : TbcdS) : TbcdS
6271: Func NegBcd( const B : TbcdS) : TbcdS
6272: Func AbsBcd( const B : TbcdS) : TbcdS
6273: Func FracBcd( const B : TbcdS) : TbcdS
6274: Func IntBcd( const B : TbcdS) : TbcdS
6275: Func RoundDigitsBcd( const B : TbcdS; Digits :Card) : TbcdS
6276: Func RoundPlacesBcd( const B : TbcdS; Places :Card) : TbcdS
6277: Func ValBcd( const S :Str) : TbcdS
6278: Func LongBcd( L : LongInt) : TbcdS
6279: Func ExtBcd( E : Extended) : TbcdS
6280: Func ExpBcd( const B : TbcdS) : TbcdS
6281: Func LnBcd( const B : TbcdS) : TbcdS
6282: Func IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS

```

```

6283: Func PowBcd( const B, E : TbcdS ) : TbcdS
6284: Func SqrtBcd( const B : TbcdS ) : TbcdS
6285: Func CmpBcd( const B1, B2 : TbcdS ) : Int
6286: Func EqDigitsBcd( const B1, B2 : TbcdS; Digits :Card) :Bool
6287: Func EqPlacesBcd( const B1, B2 : TbcdS; Digits :Card) :Bool
6288: Func IsIntBcd( const B : TbcdS ) :Bool
6289: Func TruncBcd( const B : TbcdS ) : LongInt
6290: Func BcdExt( const B : TbcdS ) : Extended
6291: Func RoundBcd( const B : TbcdS ) : LongInt
6292: Func StrBcd( const B : TbcdS; Width, Places :Card) :Str
6293: Func StrExpBcd( const B : TbcdS; Width :Card) :Str
6294: Func FormatBcd( const Format :Str; const B : TbcdS ) :Str
6295: Func StrGeneralBcd( const B : TbcdS ) :Str
6296: Func FloatFormBcd(const Mask:str;B:TbcdS;const LtCurr,RtCurr:str;Sep,DecPt:AnsiChar):str
6297: Proc ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6298:
6299: ////*****unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*****
6300: Proc StParseLine( const Data : Ansistr; Schema : TStTextDataSchema; Result : TStrings)
6301: Func StFieldTypeToStr( FieldType : TStSchemaFieldType) : Ansistr
6302: Func StStrToFieldType( const S : Ansistr) : TStSchemaFieldType
6303: Func StDeEscape( const EscStr : Ansistr) : Char
6304: Func StDoEscape( Delim : Char) : Ansistr
6305: Func StTrimTrailingChars( const S : Ansistr; Trailer : Char) : Ansistr
6306: Func AnsiHashText( const S :Str; Size : Int) : Int
6307: Func AnsiHashStr( const S :Str; Size : Int) : Int
6308: Func AnsiELFHashText( const S :Str; Size : Int) : Int
6309: Func AnsiELFHashStr( const S :Str; Size : Int) : Int
6310:
6311: ////*****unit unit ; StNetCon Package of SysTools*****
6312: with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6313: Constructor Create( AOwner : TComponent)
6314: Func Connect : DWord
6315: Func Disconnect : DWord
6316: RegisterProperty('Password', 'String', iptrw);
6317: Property(UserName', 'String', iptrw);
6318: Property(ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6319: Property(DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6320: Property(LocalDevice', 'String', iptrw);
6321: Property(ServerName', 'String', iptrw);
6322: Property(ShareName', 'String', iptrw);
6323: Property(OnConnect', 'TNotifyEvent', iptrw);
6324: Property(OnConnectFail', 'TOnConnectFailEvent', iptrw);
6325: Property(OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6326: Property(OnDisconnect', 'TNotifyEvent', iptrw);
6327: Property(OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6328: Property(OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6329: end;
6330: ////*****Thread Functions Context of Win API --- more objects in SyncObjs.pas
6331: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6332: Proc InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6333: Proc EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6334: Proc LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6335: Func InitializeCriticalSectionAndSpinCount( var
lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6336: Func SetCriticalSectionSpinCount( var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6337: Func TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6338: Proc DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6339: Func GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6340: Func SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6341: Func SuspendThread( hThread : THandle) : DWORD
6342: Func ResumeThread( hThread : THandle) : DWORD
6343: Func CreateThread2( ThreadFunc: TThreadFunction2; thrId: DWord) : THandle
6344: Func GetCurrentThread : THandle
6345: Proc ExitThread( dwExitCode : DWORD)
6346: Func TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6347: Func GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6348: Proc EndThread(ExitCode: Int);
6349: Func WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6350: Func MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6351: Proc FreeProcInstance( Proc : FARPROC)
6352: Proc FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6353: Func DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6354: Proc ParallelJob( ASelf:TObject;ATarget:Pointer;AParam:Pointer; ASafeSection:Bool);
6355: Proc ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection :Bool);
6356: Proc ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6357: Proc ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: bool;
6358: Func CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6359: Func CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection :Bool) : TParallelJob;
6360: Func CurrentParallelJobInfo : TParallelJobInfo
6361: Func ObtainParallelJobInfo : TParallelJobInfo
6362: Proc GetSystemInfo( var lpSystemInfo : TSystemInfo);
6363: Func IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL;
6364: Func SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL;
6365: Func
DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL;
6366: Func SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL;
6367: Func DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL;dwOptions:DWORD):BOOL;

```

```

6368: Func GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL;
6369: Func SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL;
6370:
6371: *****unit uPSI_JclMime;
6372: Func MimeEncodeString( const S : Ansistr) : Ansistr
6373: Func MimeDecodeString( const S : Ansistr) : Ansistr
6374: Proc MimeEncodeStream(const InputStream:TStream; const OutputStream : TStream)
6375: Proc MimeDecodeStream(const InputStream:TStream; const OutputStream : TStream)
6376: Func MimeEncodedSize( const I :Card) :Card
6377: Func MimeDecodedSize( const I :Card) :Card
6378: Proc MimeEncode(var InputBuffer:Str;const InputByteCount:Card;var OutputBuffer)
6379: Func MimeDecode(var InputBuffer:Str;const InputBytesCount:Card;var OutputBuffer):Card;
6380: Func MimeDecodePartial(var InputBuffer:Str;const InputBytesCount:Card;var OutputBuffer:Str;var
ByteBuffer:Card;var ByteBufferSpace:Card):Card
6381: Func MimeDecodePartialEnd(var OutputBuf:Str;const ByteBuf:Card;const ByteBufferSpace:Card):Card;
6382:
6383: *****unit uPSI_JclPrint;
6384: Proc DirectPrint( const Printer, Data :Str)
6385: Proc SetPrinterPixelsPerInch
6386: Func GetPrinterResolution : TPoint
6387: Func CharFitsWithinDots( const Text :Str; const Dots : Int) : Int
6388: Proc PrintMemo( const Memo : TMemo; const Rect : TRect)
6389:
6390: //*****unit uPSI_ShLwApi;*****
6391: Func StrChr( lpStart : PChar; wMatch : WORD) : PChar
6392: Func StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6393: Func StrCmpN( lpStr1, lpStr2 : PChar; nChar : Int) : Int
6394: Func StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Int) : Int
6395: Func StrCSpn( lpStr, lpSet : PChar) : Int
6396: Func StrCSpnI( lpStr1, lpSet : PChar) : Int
6397: Func StrDup( lpSrch : PChar) : PChar
6398: Func StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6399: Func StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6400: Func StrFromTimeInterval(pszOut:PChar cchMax:UINT;dwTimeMS:DWORD;digits:Int):Int
6401: Func StrIsIntEqual(fCaseSens: BOOL;lpString1,lpString2:PChar; nChar:Int): BOOL
6402: Func StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
6403: Func StrPBrk( psz, pszSet : PChar) : PChar
6404: Func StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6405: Func StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6406: Func StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6407: Func StrSpn( psz, pszSet : PChar) : Int
6408: Func StrStr( lpFirst, lpSrch : PChar) : PChar
6409: Func StrStrI( lpFirst, lpSrch : PChar) : PChar
6410: Func StrToInt( lpSrch : PChar) : Int
6411: Func StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Int) : BOOL
6412: Func StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6413: Func ChrCmpI( w1, w2 : WORD) : BOOL
6414: Func ChrCmpIA( w1, w2 : WORD) : BOOL
6415: Func ChrCmpIW( w1, w2 : WORD) : BOOL
6416: Func StrIntEqN( s1, s2 : PChar; nChar : Int) : BOOL
6417: Func StrIntEqNI( s1, s2 : PChar; nChar : Int) : BOOL
6418: Func StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Int) : PChar
6419: Func StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
6420: Func IntlStrEqWorker(fCaseSens:BOOL;lpString1,lpString2:PChar;nChar:Int):BOOL
6421: Func IntlStrEqN( s1, s2 : PChar; nChar : Int) : BOOL
6422: SZ_CONTENTTYPE_HTMLA,'String 'text/html
6423: SZ_CONTENTTYPE_HTMLW,'String 'text/html
6424: SZ_CONTENTTYPE_HTML,'string SZ_CONTENTTYPE_HTMLA);
6425: SZ_CONTENTTYPE_CDF,'String 'application/x-cdf
6426: SZ_CONTENTTYPE_CDFW,'String 'application/x-cdf
6427: SZ_CONTENTTYPE_CDF,'string SZ_CONTENTTYPE_CDF);
6428: Func PathIsHTMLFile( pszPath : PChar) : BOOL
6429: STIF_DEFAULT,'LongWord( $00000000);
6430: STIF_SUPPORT_HEX,'LongWord( $00000001);
6431: Func StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Int) : Int
6432: Func StrNCpy( psz1, psz2 : PChar; cchMax : Int) : PChar
6433: Func StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Int) : PChar
6434: Func PathAddBackslash( pszPath : PChar) : PChar
6435: Func PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6436: Func PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6437: Func PathBuildRoot( szRoot : PChar; iDrive : Int) : PChar
6438: Func PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6439: Func PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6440: Func PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6441: Func PathCompactPathEx(pszOut:PChar;pszSrc:PChar;cchMax:UINT;dwFlags:DWORD):BOOL
6442: Func PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Int
6443: Func PathFileExists( pszPath : PChar) : BOOL
6444: Func PathFindExtension( pszPath : PChar) : PChar
6445: Func PathFindFileName( pszPath : PChar) : PChar
6446: Func PathFindNextComponent( pszPath : PChar) : PChar
6447: Func PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6448: Func PathGetArgs( pszPath : PChar) : PChar
6449: Func PathFindSuffixArray(pszPath:PChar;const apszSuffix:PChar;iArraySize:Int): PChar
6450: Func PathIsLFNFileSpec( lpName : PChar) : BOOL
6451: Func PathGetCharType( ch : Char) : UINT
6452: GCT_INVALID,'LongWord( $0000);
6453: GCT_LFNCHAR,'LongWord( $0001);
6454: GCT_SHORTCHAR,'LongWord( $0002);
6455: GCT_WILD,'LongWord( $0004);

```



```

6456: GCT_SEPARATOR', 'LongWord( $0008);
6457: Func PathGetDriveNumber( pszPath : PChar ) : Int
6458: Func PathIsDirectory( pszPath : PChar ) : BOOL
6459: Func PathIsDirectoryEmpty( pszPath : PChar ) : BOOL
6460: Func PathIsFileSpec( pszPath : PChar ) : BOOL
6461: Func PathIsPrefix( pszPrefix, pszPath : PChar ) : BOOL
6462: Func PathIsRelative( pszPath : PChar ) : BOOL
6463: Func PathIsRoot( pszPath : PChar ) : BOOL
6464: Func PathIsSameRoot( pszPath1, pszPath2 : PChar ) : BOOL
6465: Func PathIsUNC( pszPath : PChar ) : BOOL
6466: Func PathIsNetworkPath( pszPath : PChar ) : BOOL
6467: Func PathIsUNCServer( pszPath : PChar ) : BOOL
6468: Func PathIsUNCServerShare( pszPath : PChar ) : BOOL
6469: Func PathIsContentType( pszPath, pszContentType : PChar ) : BOOL
6470: Func PathIsURL( pszPath : PChar ) : BOOL
6471: Func PathMakePretty( pszPath : PChar ) : BOOL
6472: Func PathMatchSpec( pszFile, pszSpec : PChar ) : BOOL
6473: Func PathParseIconLocation( pszIconFile : PChar ) : Int
6474: Proc PathQuoteSpaces( lpsz : PChar)
6475: Func PathRelativePathTo( pszPath: PChar; pszFrom: PChar; dwAttrFrom: DWORD; pszTo: PChar; dwAttrTo: DWORD ) : BOOL;
6476: Proc PathRemoveArgs( pszPath : PChar)
6477: Func PathRemoveBackslash( pszPath : PChar ) : PChar
6478: Proc PathRemoveBlanks( pszPath : PChar)
6479: Proc PathRemoveExtension( pszPath : PChar)
6480: Func PathRemoveFileSpec( pszPath : PChar ) : BOOL
6481: Func PathRenameExtension( pszPath : PChar; pszExt : PChar ) : BOOL
6482: Func PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6483: Proc PathSetDlgItemPath( hDlg : HWND; id : Int; pszPath : PChar)
6484: Func PathSkipRoot( pszPath : PChar ) : PChar
6485: Proc PathStripPath( pszPath : PChar)
6486: Func PathStripToRoot( pszPath : PChar ) : BOOL
6487: Proc PathUnquoteSpaces( lpsz : PChar)
6488: Func PathMakeSystemFolder( pszPath : PChar ) : BOOL
6489: Func PathUnmakeSystemFolder( pszPath : PChar ) : BOOL
6490: Func PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD ) : BOOL
6491: Proc PathUndecorate( pszPath : PChar)
6492: Func PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6493: URL_SCHEME_INVALID, 'LongInt'( - 1);
6494: URL_SCHEME_UNKNOWN, 'LongInt'( 0);
6495: URL_SCHEME_FTP, 'LongInt'( 1);
6496: URL_SCHEME_HTTP, 'LongInt'( 2);
6497: URL_SCHEME_GOPHER, 'LongInt'( 3);
6498: URL_SCHEME_MAILTO, 'LongInt'( 4);
6499: URL_SCHEME_NEWS, 'LongInt'( 5);
6500: URL_SCHEME_NNTP, 'LongInt'( 6);
6501: URL_SCHEME_TELNET, 'LongInt'( 7);
6502: URL_SCHEME_WAIS, 'LongInt'( 8);
6503: URL_SCHEME_FILE, 'LongInt'( 9);
6504: URL_SCHEME_MK, 'LongInt'( 10);
6505: URL_SCHEME_HTTPS, 'LongInt'( 11);
6506: URL_SCHEME_SHELL, 'LongInt'( 12);
6507: URL_SCHEME_SNEWS, 'LongInt'( 13);
6508: URL_SCHEME_LOCAL, 'LongInt'( 14);
6509: URL_SCHEME_JAVASCRIPT, 'LongInt'( 15);
6510: URL_SCHEME_VBSCRIPT, 'LongInt'( 16);
6511: URL_SCHEME_ABOUT, 'LongInt'( 17);
6512: URL_SCHEME_RES, 'LongInt'( 18);
6513: URL_SCHEME_MAXVALUE, 'LongInt'( 19);
6514: URL_SCHEME, 'Int
6515: URL_PART_NONE, 'LongInt'( 0);
6516: URL_PART_SCHEME, 'LongInt'( 1);
6517: URL_PART_HOSTNAME, 'LongInt'( 2);
6518: URL_PART_USERNAME, 'LongInt'( 3);
6519: URL_PART_PASSWORD, 'LongInt'( 4);
6520: URL_PART_PORT, 'LongInt'( 5);
6521: URL_PART_QUERY, 'LongInt'( 6);
6522: URL_PART, 'DWORD
6523: URLIS_URL, 'LongInt'( 0);
6524: URLIS_OPAQUE, 'LongInt'( 1);
6525: URLIS_NOHISTORY, 'LongInt'( 2);
6526: URLIS_FILEURL, 'LongInt'( 3);
6527: URLIS_APPLIABLE, 'LongInt'( 4);
6528: URLIS_DIRECTORY, 'LongInt'( 5);
6529: URLIS_HASQUERY, 'LongInt'( 6);
6530: TURLIS, 'DWORD
6531: URL_UNESCAPE, 'LongWord( $10000000);
6532: URL_ESCAPE_UNSAFE, 'LongWord( $20000000);
6533: URL_PLUGGABLE_PROTOCOL, 'LongWord( $40000000);
6534: URL_WININET_COMPATIBILITY, 'LongWord( DWORD ( $80000000 ));
6535: URL_DONT_ESCAPE_EXTRA_INFO, 'LongWord( $02000000);
6536: URL_ESCAPE_SPACES_ONLY, 'LongWord( $04000000);
6537: URL_DONT_SIMPLIFY, 'LongWord( $08000000);
6538: URL_NO_META, 'longword( URL_DONT_SIMPLIFY);
6539: URL_UNESCAPE_INPLACE, 'LongWord( $00100000);
6540: URL_CONVERT_IF_DOSPATH, 'LongWord( $00200000);
6541: URL_UNESCAPE_HIGH_ANSI_ONLY, 'LongWord( $00400000);
6542: URL_INTERNAL_PATH, 'LongWord( $00800000);
6543: URL_FILE_USE_PATHURL, 'LongWord( $00010000);
6544: URL_ESCAPE_PERCENT, 'LongWord( $00001000);

```

```

6545: URL_ESCAPE_SEGMENT_ONLY', 'LongWord( $00002000);
6546: URL_PARTFLAG_KEEPScheme', 'LongWord( $00000001);
6547: URL_APPLY_DEFAULT', 'LongWord( $00000001);
6548: URL_APPLY_GUESSScheme', 'LongWord( $00000002);
6549: URL_APPLY_GUESSFILE', 'LongWord( $00000004);
6550: URL_APPLY_FORCEAPPLY', 'LongWord( $00000008);
6551: Func UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL ) : Int
6552: Func UrlCombine( pszBase, pszRelative: PChar; pszCombin: PChar; out pcchCombin: DWORD; dwFlags: DWORD ) : HRESULT;
6553: Func UrlCanonicalize( pszUrl: PChar; pszCanonicalized: PChar; pcchCanonic: DWORD; dwFlags: DWORD ) : HRESULT;
6554: Func UrlIsOpaque( pszURL : PChar ) : BOOL
6555: Func UrlIsNoHistory( pszURL : PChar ) : BOOL
6556: Func UrlIsFileUrl( pszURL : PChar ) : BOOL
6557: Func UrlIs( pszUrl : PChar; UrlIs : TUrlIs ) : BOOL
6558: Func UrlGetLocation( psz1 : PChar ) : PChar
6559: Func UrlUnescape( pszUrl, pszUnescaped: PChar; pcchUnescaped: DWORD; dwFlags: DWORD ) : HRESULT
6560: Func UrlEscape( pszUrl: PChar; pszEscaped: PChar; pcchEscaped: DWORD; dwFlags: DWORD ) : HRESULT
6561: Func UrlCreateFromPath( pszPath: PChar; pszUrl: PChar; pcchUrl: DWORD; dwFlags: DWORD ) : HRESULT
6562: Func PathCreateFromUrl( pszUrl: PChar; pszPath: PChar; pcchPath: DWORD; dwFlags: DWORD ) : HRESULT
6563: Func UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6564: Func UrlGetPart( pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart, dwFlags: DWORD ) : HRESULT
6565: Func UrlApplyScheme( pszIn: PChar; pszOut: PChar; pcchOut : DWORD; dwFlags: DWORD ) : HRESULT
6566: Func HashData( pbData: BYTE; cbData: DWORD; pbHash: BYTE; cbHash: DWORD ) : HRESULT
6567: Func UrlEscapeSpaces( pszUrl: PChar; pszEscaped: PChar; pcchEscaped: WORD ) : HRESULT
6568: Func UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD ) : HRESULT
6569: Func SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6570: Func SHDeleteKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6571: Func SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar ) : DWORD
6572: Func SHEnumKeyEx( hKey: HKEY; dwIndex : DWORD; pszName: PChar; var pcchName: DWORD ) : Longint
6573: Func SHEnumValue( hKey: HKEY; dwIndex: DWORD; pszValueName: PChar; var pcchValueName: DWORD; pdwType: DWORD; pvData:
    Pointer; pcbData: DWORD ) : Longint
6574: Func SHQueryInfoKey( hKey: HKEY; pcSubKeys, pcchMaxSubKeyLen, pcVal, pcchMaxValueNameLen: DWORD ) : Longint;
6575: Func SHCopyKey( hKeySrc: HKEY; szSrcSubKey: PChar; hKeyDest: HKEY; fReserved: DWORD ) : DWORD
6576: Func SHRegGetPath( hKey: HKEY; pcszSubKey, pcszValue: PChar; pszPath: PChar; dwFlags: DWORD ) : DWORD
6577: Func SHRegSetPath( hKey: HKEY; pcszSubKey, pcszValue, pcszPath: PChar; dwFlags: DWORD ) : DWORD
6578: SHREGDEL_DEFAULT', 'LongWord( $00000000);
6579: SHREGDEL_HKCU', 'LongWord( $00000001);
6580: SHREGDEL_HKLM', 'LongWord( $00000010);
6581: SHREGDEL_BOTH', 'LongWord( $00000011);
6582: SHREGENUM_DEFAULT', 'LongWord( $00000000);
6583: SHREGENUM_HKCU', 'LongWord( $00000001);
6584: SHREGENUM_HKLM', 'LongWord( $00000010);
6585: SHREGENUM_BOTH', 'LongWord( $00000011);
6586: SHREGSET_HKCU', 'LongWord( $00000001);
6587: SHREGSET_FORCE_HKCU', 'LongWord( $00000002);
6588: SHREGSET_HKLM', 'LongWord( $00000004);
6589: SHREGSET_FORCE_HKLM', 'LongWord( $00000008);
6590: TSHRegDelFlags', 'DWORD
6591: TSHRegEnumFlags', 'DWORD
6592: HUSKEY', 'THandle
6593: ASSOCF_INIT_NOREMAPCLSID', 'LongWord( $00000001);
6594: ASSOCF_INIT_BYEXENAME', 'LongWord( $00000002);
6595: ASSOCF_OPEN_BYEXENAME', 'LongWord( $00000002);
6596: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord( $00000004);
6597: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord( $00000008);
6598: ASSOCF_NOUSERSETTINGS', 'LongWord( $00000010);
6599: ASSOCF_NOTRUNCATE', 'LongWord( $00000020);
6600: ASSOCF_VERIFY', 'LongWord( $00000040);
6601: ASSOCF_REMAPRUNDLL', 'LongWord( $00000080);
6602: ASSOCF_NOFIXUPS', 'LongWord( $00000100);
6603: ASSOCF_IGNOREBASECLASS', 'LongWord( $00000200);
6604: ASSOCF', 'DWORD
6605: ASSOCSTR_COMMAND', 'LongInt'( 1);
6606: ASSOCSTR_EXECUTABLE', 'LongInt'( 2);
6607: ASSOCSTR_FRIENDLYDOCNAME', 'LongInt'( 3);
6608: ASSOCSTR_FRIENDLYAPPNAME', 'LongInt'( 4);
6609: ASSOCSTR_NOOPEN', 'LongInt'( 5);
6610: ASSOCSTR_SHELLNEWVALUE', 'LongInt'( 6);
6611: ASSOCSTR_DDECOMMAND', 'LongInt'( 7);
6612: ASSOCSTR_DDEIFEXEC', 'LongInt'( 8);
6613: ASSOCSTR_DDEAPPLICATION', 'LongInt'( 9);
6614: ASSOCSTR_DDETOPIC', 'LongInt'( 10);
6615: ASSOCSTR_INFOTIP', 'LongInt'( 11);
6616: ASSOCSTR_MAX', 'LongInt'( 12);
6617: ASSOCSTR', 'DWORD
6618: ASSOCKEY_SHELLEXECCLASS', 'LongInt'( 1);
6619: ASSOCKEY_APP', 'LongInt'( 2);
6620: ASSOCKEY_CLASS', 'LongInt'( 3);
6621: ASSOCKEY_BASECLASS', 'LongInt'( 4);
6622: ASSOCKEY_MAX', 'LongInt'( 5);
6623: ASSOCKEY', 'DWORD
6624: ASSOCDATA_MSIDESCRIPTOR', 'LongInt'( 1);
6625: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt'( 2);
6626: ASSOCDATA_QUERYCLASSSTORE', 'LongInt'( 3);
6627: ASSOCDATA_HASPERUSERASSOC', 'LongInt'( 4);
6628: ASSOCDATA_MAX', 'LongInt'( 5);
6629: ASSOCDATA', 'DWORD
6630: ASSOCENUM_NONE', 'LongInt'( 0);
6631: ASSOCENUM', 'DWORD
6632: SID_IQueryAssociations', 'String' '{c46ca590-3c3f-11d2-bee6-0000f805ca57}

```

```

6633: SHACF_DEFAULT $00000000);
6634: SHACF_FILESYSTEM', 'LongWord( $00000001);
6635: SHACF_URLHISTORY', 'LongWord( $00000002);
6636: SHACF_URLMRU', 'LongWord( $00000004);
6637: SHACF_USETAB', 'LongWord( $00000008);
6638: SHACF_FILESYS_ONLY', 'LongWord( $00000010);
6639: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord( $10000000);
6640: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord( $20000000);
6641: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord( $40000000);
6642: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord( DWORD ( $80000000 ));
6643: Func SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6644: Proc SHSetThreadRef( punk : IUnknown)
6645: Proc SHGetThreadRef( out ppunk : IUnknown)
6646: CTF_INSIST', 'LongWord( $00000001);
6647: CTF_THREAD_REF', 'LongWord( $00000002);
6648: CTF_PROCESS_REF', 'LongWord( $00000004);
6649: CTF_COINIT', 'LongWord( $00000008);
6650: Func SHCreateShellPalette( hdc : HDC) : HPALETTE
6651: Proc ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6652: Func ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6653: Func ColorAdjustLuma( clrRGB : TColorRef; n : Int; fScale : Bool) : TColorRef
6654: Func GetSysColorBrush( nIndex : Int) : HBRUSH
6655: Func DrawFocusRect( hdc : HDC; const lprc : TRect) : BOOL
6656: Func FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Int
6657: Func FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Int
6658: Func InvertRect( hdc : HDC; const lprc : TRect) : BOOL
6659: Func SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Int) : BOOL
6660: Func SetRectEmpty( var lprc : TRect) : BOOL
6661: Func CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6662: Func InflateRect( var lprc : TRect; dx, dy : Int) : BOOL
6663: Func IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6664: Func SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6665: Func InitializeFlatSB( hWnd : HWND) : Bool
6666: Proc UninitializeFlatSB( hWnd : HWND)
6667: Func FlatSB_GetScrollProp( p1 : HWND; propIndex : Int; p3 : PInt) : Bool
6668: Func FlatSB_SetScrollProp( p1 : HWND; index : Int; newValue : Int; p4 : Bool) : Bool
6669: Func GET_APPCOMMAND_LPARAM( lParam : Int) : Word //of JvWin32
6670: Func GET_DEVICE_LPARAM( lParam : Int) : Word
6671: Func GET_MOUSEORKEY_LPARAM( lParam : Int) : Int
6672: Func GET_FLAGS_LPARAM( lParam : Int) : Word
6673: Func GET_KEYSTATE_LPARAM( lParam : Int) : Word
6674:
6675: // ***** 204 unit uPSI_ShellAPI;
6676: Func DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6677: Func DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6678: Proc DragFinish( Drop : HDROP)
6679: Proc DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6680: Func ShellExecute( hWnd : HWND; Operation, FileName, Parameters, Directory : PChar; ShowCmd : Int) : HINST
6681: Func FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6682: Func ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Int
6683: Func DuplicateIcon( hinst : HINST; Icon : HICON) : HICON
6684: Func ExtractAssociatedIcon( hinst : HINST; lpIconPath : PChar; var lpIcon : Word) : HICON
6685: Func ExtractIcon( hinst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6686: Func SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6687: Func DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6688: Func ExtractIconEx( lpszFile : PChar; nIconIndex : Int; var phiconLarge, phiconSmall : HICON; nIcons : UINT) : UINT;
6689: Proc SHFreeNameMappings( hNameMappings : THandle)
6690:
6691: DLLVER_PLATFORM_WINDOWS', 'LongWord( $00000001);
6692: DLLVER_PLATFORM_NT', 'LongWord( $00000002);
6693: DLLVER_MAJOR_MASK', 'LongWord( Int64 ( $FFFF000000000000 ));
6694: DLLVER_MINOR_MASK', 'LongWord( Int64 ( $0000FFFF00000000 ));
6695: DLLVER_BUILD_MASK', 'LongWord( Int64 ( $00000000FFFF0000 ));
6696: DLLVER_QFE_MASK', 'LongWord( Int64 ( $000000000000FFFF ));
6697: Func MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6698: Func SimpleXMLEncode( const S : Str) : Str
6699: Proc SimpleXMLDecode( var S : Str; TrimBlanks : Bool)
6700: Func XMLEncode( const S : Str) : Str
6701: Func XMLDecode( const S : Str) : Str
6702: Func EntityEncode( const S : Str) : Str
6703: Func EntityDecode( const S : Str) : Str
6704:
6705: Proc RRegister_CPort_Routines( S : TPSExec);
6706: Proc EnumComPorts( Ports : TStrings)
6707: Proc ListComPorts( Ports : TStrings)
6708: Proc ComPorts( Ports : TStrings) //Alias to Arduino
6709: Func GetComPorts : TStringList;
6710: Func StrToBaudRate( Str : Str) : TBaudRate
6711: Func StrToStopBits( Str : Str) : TStopBits
6712: Func StrToDataBits( Str : Str) : TDataBits
6713: Func StrToParity( Str : Str) : TParityBits
6714: Func StrToFlowControl( Str : Str) : TFlowControl
6715: Func BaudRateToStr( BaudRate : TBaudRate) : Str
6716: Func StopBitsToStr( StopBits : TStopBits) : Str
6717: Func DataBitsToStr( DataBits : TDataBits) : Str
6718: Func ParityToStr( Parity : TParityBits) : Str
6719: Func FlowControlToStr( FlowControl : TFlowControl) : Str
6720: Func ComErrorsToStr( Errors : TComErrors) : Str
6721: Func GetMessage( var lpMsg : TMsg; hWnd : HWND; wParamFilterMin, wParamFilterMax : UINT) : BOOL

```

```

6722: Func DispatchMessage( const lpMsg : TMsg ) : Longint
6723: Func TranslateMessage( const lpMsg : TMsg ) : BOOL
6724: Func SetMessageQueue( cMessagesMax : Int ) : BOOL
6725: Func PeekMessage( var lpMsg: TMsg; hWnd: HWND; wMsgFilterMin, wMsgFilterMax, wRemoveMsg: UINT ) : BOOL
6726: Func GetMessagePos : DWORD
6727: Func GetMessageTime : Longint
6728: Func GetMessageExtraInfo : Longint
6729: Func GetSpecialFolderPath( const FolderName : Str; CanCreate : Bool ) : Str
6730: Proc JAddToRecentDocs( const Filename : Str )
6731: Proc ClearRecentDocs
6732: Func ExtractIconFromFile( FileName : Str; Index : Int ) : HICON
6733: Func CreateShellLink( const AppName, Desc : Str; Dest : Str ) : Str
6734: Proc GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo )
6735: Proc SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo )
6736: Func RecycleFile( FileToRecycle : Str ) : Bool
6737: Func JCopyFile( FromFile, ToDir : Str ) : Bool
6738: Func ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType ) : UINT
6739: Func ShellObjectTypeConstToEnum( ShellObjectType : UINT ) : TShellObjectType
6740: Func QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel: DWORD; lpBuffer: LPBYTE; cbBufSize: DWORD;
var pcbBytesNeeded : DWORD ) : BOOL
6741: Func QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel: DWORD; lpBuffer: LPBYTE; cbBufSize: DWORD;
var pcbBytesNeeded : DWORD ) : BOOL
6742: Func QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel: DWORD; lpBuffer: LPBYTE; cbBufSize: DWORD;
var pcbBytesNeeded : DWORD ) : BOOL
6743: Func EnumServicesStatusExA( hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
dwServiceState: DWORD; lpServices: LPBYTE; cbBufSize: DWORD; var pcbBytesNeeded, lpServicesReturned,
lpResumeHandle: DWORD; pszGroupName: LP
6744: Func EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
dwServiceState : DWORD; lpServices: LPBYTE; cbBufSize: DWORD; var pcbBytesNeeded, lpServicesReturned,
lpResumeHandle: DWORD; pszGroupName
6745: Func EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
dwServiceState: DWORD; lpServices: LPBYTE; cbBufSize: DWORD; var pcbBytesNeeded, lpServicesReturned,
lpResumeHandle: DWORD; pszGroupName
6746: Func ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR ) : BOOL
6747:
6748: ***** unit uPSI_JclPeImage;
6749: Func IsValidPeFile( const FileName : TFileName ) : Bool
6750: // Func PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders ) : Bool
6751: Func PeCreateNameHintTable( const FileName : TFileName ) : Bool
6752: Func PeRebaseImage( const ImageName: TFileName; NewBase: DWORD; TimeStamp : DWORD; MaxNewSize: DWORD ) :
TJclRebaseImageInfo
6753: Func PeVerifyChecksum( const FileName : TFileName ) : Bool
6754: Func PeClearChecksum( const FileName : TFileName ) : Bool
6755: Func PeUpdateChecksum( const FileName : TFileName ) : Bool
6756: Func PeDoesExportFunction( const FileName: TFileName; const FunctionName: str; Options: TJclSmartCompOptions ) : Bool;
6757: Func PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName: str; var
ForwardedName: str; Options: TJclSmartCompOptions ) : Boolean
6758: Func PeIsExportFunctionForwarded( const FileName: TFileName; const
FunctionName: str; Options: TJclSmartCompOptions ) : Bool
6759: Func PeDoesImportFunction( const FileName: TFileName; const FunctionName: str; const LibraryName: str; Options:
TJclSmartCompOptions ) : Bool
6760: Func PeDoesImportLibrary( const FileName: TFileName; const LibraryName: str; Recursive: Boolean ) : Boolean;
6761: Func PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive : Bool;
FullPathName : Bool ) : Bool
6762: Func PeImportedFunctions( const FileName: TFileName; const FunctionsList: TStrings; const LibraryName: str;
IncludeLibNames : Bool ) : Bool
6763: Func PeExportedFunctions( const FileName: TFileName; const FunctionsList: TStrings ) : Boolean
6764: Func PeExportedNames( const FileName: TFileName; const FunctionsList: TStrings ) : Bool
6765: Func PeExportedVariables( const FileName: TFileName; const FunctionsList: TStrings ) : Boolean
6766: Func PeResourceKindNames( const FileN: TFileName; ResourceType: TJclPeResourceKind; const
NamesList: TStrings ) : Bool
6767: Func PeBorFormNames( const FileName: TFileName; const NamesList: TStrings : Boolean
6768: Func PeBorDependedPackages( const FileName: TFileName; PackagesList: TStrings; FullPathName,
Descript: Bool ) : Bool;
6769: Func PeFindMissingImports( const FileName: TFileName; MissingImportsList: TStrings ) : Bool;
6770: Func PeFindMissingImports1( RequiredImportsList, MissingImportsList: TStrings ) : Bool;
6771: Func PeCreateRequiredImportList( const FileName: TFileName; RequiredImportsList: TStrings ) : Bool;
6772: // Func PeMapImgNtHeaders( const BaseAddress : Pointer ) : PImageNtHeaders
6773: // Func PeMapImgLibraryName( const BaseAddress : Pointer ) : Str
6774: // Func PeMapImgSections( const NtHeaders : PImageNtHeaders ) : PImageSectionHeader
6775: // Func PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : Str ) :
PImageSectionHeader
6776: // Func PeMapImgExportedVariables( const Module: HMODULE; const VariablesList: TStrings ) : Bool
6777: // Func PeMapImgResolvePackageThunk( Address : Pointer ) : Pointer
6778: Func PeMapFindResource( const Module: HMODULE; const ResourceType: PChar; const ResourceName: str ) : ___Pointer;
6779: SIRegister_TJclPeSectionStream( CL );
6780: SIRegister_TJclPeMapImgHookItem( CL );
6781: SIRegister_TJclPeMapImgHooks( CL );
6782: // Func PeDbgImgNtHeaders( ProcessHandle: THandle; BaseAddress: Pointer; var NtHeaders: TImageNtHeaders ) : Boolean
6783: // Func PeDbgImgLibraryName( ProcessHandle: THandle; BaseAddress: Pointer; var Name: str ) : Bool
6784: Type TJclBorUmSymbolKind, ' ( skData, skFunction, skConstructor, skDestructor, skRTTI, skVTable )
6785: TJclBorUmSymbolModifier, ' ( smQualified, smLinkProc )
6786: TJclBorUmSymbolModifiers, ' set of TJclBorUmSymbolModifier
6787: TJclBorUmDescription, ' record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6788: TJclBorUmResult, ' ( urOk, urNotMangled, urMicrosoft, urError )
6789: TJclPeUmResult, ' ( umNotMangled, umBorland, umMicrosoft )
6790: Func PeBorUnmangleName( const Name: Str; var Unmangled: Str; var Description: TJclBorUmDescription; var
BasePos: Int ) : TJclBorUmResult;
6791: Func PeBorUnmangleName1( const Name: str; var Unmangled: str; var
Descript: TJclBorUmDescription ) : TJclBorUmResult;

```

```

6792: Func PeBorUnmangleName2( const Name :Str; var Unmangled :Str) : TJclBorUmResult;
6793: Func PeBorUnmangleName3( const Name :Str) :Str;
6794: Func PeIsNameMangled( const Name :Str) : TJclPeUmResult
6795: Func PeUnmangleName( const Name :Str; var Unmangled :Str) : TJclPeUmResult
6796:
6797:
6798: //***** SysTools uPSI_StSystem; *****
6799: Func StCopyFile( const SrcPath, DestPath : Ansistr) :Card
6800: Func CreateTempFile( const aFolder : Ansistr; const aPrefix : Ansistr): Ansistr
6801: Func DeleteVolumeLabel( Drive : Char) :Card
6802: //Proc EnumerateDirectories(const StartDir:Ansistr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6803: //Proc EnumerateFiles(const StartDir:Ansistr;FL:TStrings;SubDirs:Bool IncludeItem:TIncludeItemFunc);
6804: Func FileHandlesLeft( MaxHandles :Card) :Card
6805: Func FileMatchesMask( const FileName, FileMask : Ansistr) :Bool
6806: Func FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6807: Func FindNthSlash( const Path : Ansistr; n : Int) : Int
6808: Func FlushOsBuffers( Handle : Int) :Bool
6809: Func GetCurrentUser : Ansistr
6810: Func GetDiskClass( Drive : Char) : DiskClass
6811: Func GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,SectorsPerCluster:Card):Bool;
6812: Func GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
DiskSize:Double):Bool;
6813: Func GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var DiskSize:Comp):Bool;
6814: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6815: Func GetDiskSpace2(const path:Str; index: Int): int64;
6816: Func GetFileCreateDate( const FileName : Ansistr) : TDateTime
6817: Func StGetFileLastAccess( const FileName : Ansistr) : TDateTime
6818: Func GetFileLastModify( const FileName : Ansistr) : TDateTime
6819: Func GetHomeFolder( aForceSlash :Bool) : Ansistr
6820: Func GetLongPath( const APath : Ansistr) : Ansistr
6821: Func GetMachineName : Ansistr
6822: Func GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) :Card
6823: Func GetParentFolder( const APath : Ansistr; aForceSlash :Bool): Ansistr
6824: Func GetShortPath( const APath : Ansistr) : Ansistr
6825: Func GetSystemFolder( aForceSlash :Bool) : Ansistr
6826: Func GetTempFolder( aForceSlash :Bool) : Ansistr
6827: Func StGetWindowsFolder( aForceSlash :Bool) : Ansistr
6828: Func GetWorkingFolder( aForceSlash :Bool) : Ansistr
6829: Func GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset: Int): TStDateTimeRec
6830: Func StIsDirectory( const DirName : Ansistr) :Bool
6831: Func IsDirectoryEmpty( const S : Ansistr) : Int
6832: Func IsDriveReady( Drive : Char) :Bool
6833: Func IsFile( const FileName : Ansistr) :Bool
6834: Func IsFileArchive( const S : Ansistr) : Int
6835: Func IsFileHidden( const S : Ansistr) : Int
6836: Func IsFileReadOnly( const S : Ansistr) : Int
6837: Func IsFileSystem( const S : Ansistr) : Int
6838: Func LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Int) : TStDateTimeRec
6839: Func ReadVolumeLabel( var VolName : Ansistr; Drive : Char) :Card
6840: Func SameFile( const FilePath1, FilePath2 : Ansistr; var ErrorCode : Int) :Bool
6841: Func SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) :Card
6842: Proc SplitPath( const APath : Ansistr; Parts : TStrings)
6843: Func StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6844: Func StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6845: Func UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6846: Func ValidDrive( Drive : Char) :Bool
6847: Func WriteVolumeLabel( const VolName : Ansistr; Drive : Char) :Card
6848:
6849: //*****unit uPSI_JclLANMan;*****
6850: Func CreateAccount( const Server,Username,Fullname>Password,Description,Homedir,Script :str; const
PasswordNeverExpires : Bool) : Bool
6851: Func CreateLocalAccount( const Username,Fullname>Password,Description, Homedir, Script :Str;const
PasswordNeverExpires :Bool) :Bool
6852: Func DeleteAccount( const Servername, Username :Str) :Bool
6853: Func DeleteLocalAccount( Username :Str) :Bool
6854: Func CreateLocalGroup( const Server, Groupname, Description :Str) :Bool
6855: Func CreateGlobalGroup( const Server, Groupname, Description :Str) :Bool
6856: Func DeleteLocalGroup( const Server, Groupname :Str) :Bool
6857: Func GetLocalGroups( const Server :Str; const Groups : TStrings) :Bool
6858: Func GetGlobalGroups( const Server :Str; const Groups : TStrings) :Bool
6859: Func LocalGroupExists( const Group :Str) :Bool
6860: Func GlobalGroupExists( const Server, Group :Str) :Bool
6861: Func AddAccountToLocalGroup( const Accountname, Groupname :Str) :Bool
6862: Func LookupGroupName( const Server :Str; const RID : TNetWellKnownRID) :Str
6863: Proc ParseAccountName( const QualifiedName:Str; var Domain, UserName :Str)
6864: Func IsLocalAccount( const AccountName :Str) :Bool
6865: Func TimeStampInterval( StartStamp, EndStamp : TDateTime) : Int
6866: Func GetRandomString( NumChar :Card) :Str
6867:
6868: //*****unit uPSI_cUtils;*****
6869: TypeS(TUnitType', ( utSrc, utHead, utRes, utPrj, utOther )
6870: Func cIsWinNT :Bool
6871: Proc cFilesFromWildcard(Directory,Mask:Str;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Boolean;
6872: Func cExecuteFile( const FileName, Params, DefaultDir :Str; ShowCmd : Int) : THandle
6873: Func cRunAndGetOutput(Cmd,WorkDir:str; ErrFunc:TErrFunc;LineOutputFunc:TLineOutputFunc;
CheckAbortFunc:TCheckAbortFunc;ShowReturnValue:Bool):str
6874: Func cGetShortName( FileName :Str) :Str
6875: Proc cShowError( Msg :Str)
6876: Func cCommaStrToStr( s :Str; formatstr :Str) :Str

```



```

6877: Func cIncludeQuoteIfSpaces( s :Str) :Str
6878: Func cIncludeQuoteIfNeeded( s :Str) :Str
6879: Proc cLoadFilefromResource( const FileName :Str; ms : TMemoryStream)
6880: Func cValidateFile(const FileName:str;const WorkPath:str;const CheckDirs:bool):str;
6881: Func cBuildFilter( var value :Str; const FLTStyle : TFILTERSET) :Bool;
6882: Func cBuildFilter1( var value :Str; const _filters : array of string) :Bool;
6883: Func cCodeInstoStr( s :Str) :Str
6884: Func cStrtoCodeIns( s :Str) :Str
6885: Proc cStrtoAttr( var Attr : TSynHighlighterAttributes; Value :Str)
6886: Func cAttrtoStr( const Attr : TSynHighlighterAttributes) :Str
6887: Proc cStrtoPoint( var pt : TPoint; value :Str)
6888: Func cPointtoStr( const pt : TPoint) :Str
6889: Func cListtoStr( const List : TStrings) :Str
6890: Func ListtoStr( const List : TStrings) :Str
6891: Proc StrtoList( s :Str; const List : TStrings; const delimiter : char)
6892: Proc cStrtoList( s :Str; const List : TStrings; const delimiter : char)
6893: Func cGetFileType( const FileName :Str) : TUnitType
6894: Func cGetExTyp( const FileName :Str) : TExUnitType
6895: Proc cSetPath( Add :Str; const UseOriginal :Bool)
6896: Func cExpandFileto( const FileName :Str; const BasePath :Str) :Str
6897: Func cFileSamePath( const FileName :Str; const TestPath :Str) :Bool
6898: Proc cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6899: Func cGetLastPos( const SubStr :Str; const S :Str) : Int
6900: Func cGenMakePath( FileName :Str) :Str;
6901: Func cGenMakePath2( FileName :Str) :Str
6902: Func cGenMakePath1(FileName:Str; EscapeSpaces,EncloseInQuotes:Bool):str;
6903: Func cGetRealPath( BrokenFileName :Str; Directory :Str) :Str
6904: Func cCalcMod( Count : Int) : Int
6905: Func cGetVersionString( FileName :Str) :Str
6906: Func cCheckChangeDir( var Dir :Str) :Bool
6907: Func cGetAssociatedProgram(const Extension:str;var Filename,Description:str):bool
6908: Func cIsNumeric( s :Str) :Bool
6909: Proc StrtoAttr( var Attr : TSynHighlighterAttributes; Value :Str)
6910: Func AttrtoStr( const Attr : TSynHighlighterAttributes) :Str
6911: Func GetFileType( const FileName :Str) : TUnitType
6912: Func Atoi(const aStr:Str): Int
6913: Func Itoa(const aint: Int):Str
6914: Func Atof(const aStr:Str): double;
6915: Func Atol(const aStr:Str): longint;
6916:
6917: Proc SRegister_cHTTPUtils(CL: TPSPascalCompiler);
6918: begin
6919:   FindClass('TOBJECT'),'EHTTP
6920:   FindClass('TOBJECT'),'EHTTPParser
6921:   //AnsiCharSet', 'set of AnsiChar
6922:   AnsistrArray', 'array of Ansistr
6923:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6924:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6925:   THTTPVersion', 'record Value : THTTPVersionEnum; Protocol : TH'
6926:   + 'TTPProtocolEnum; CustomProtocol : Ansistr; CustomMajVersion : Int; '
6927:   + 'CustomMinVersion : Int; end
6928:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6929:   + 'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6930:   + 'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6931:   + 'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6932:   + 'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6933:   + 'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6934:   + 'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6935:   + ' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6936:   + 'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6937:   + 'nection, hntOrigin, hntKeepAlive )
6938:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: Ansistr; end
6939:   THTTPCustomHeader', 'record FieldName : Ansistr; FieldValue : ' Ansistr; end
6940:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6941:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6942:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount: Int64; end
6943:   //PHTTPContentLength', '^THTTPContentLength // will not work
6944:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6945:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6946:   + 'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6947:   + 'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6948:   + 'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6949:   + 'ationCustom, hctAudioCustom, hctVideoCustom )
6950:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6951:   + 'ajor: Ansistr; CustomMinor: Ansistr; Parameters: AnsistrArray; CustomStr: Ansistr; end
6952:   THTTPDateFieldEnum', '( hdnNone, hdnCustom, hdnParts, hdnDateTime )
6953:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek : '
6954:   + ' Int; Day : Int; Month : Int; Year : Int; Hour : Int; '
6955:   + 'Min : Int; Sec : Int; TimeZoneGMT : Bool; CustomTimeZone : Ansi'
6956:   + 'String; DateTime : TDateTime; Custom : Ansistr; end
6957:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6958:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum; Custom: Ansistr; end
6959:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6960:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum; Custom: Ansistr; end
6961:   THTTPPageFieldEnum', '( hafNone, hafCustom, hafAge )
6962:   THTTPPageField', 'record Value: THTTPPageFieldEnum; Age : Int64; Custom: Ansistr; end
6963:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6964:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6965:   + ', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )

```

```

6966: THTTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6967: +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6968: +'ProxyRevalidate, hccrfMaxAge, hccrfSMMaxAge )
6969: THTTTPCacheControlField', 'record Value : THTTTPCacheControlFieldEnum; end
6970: THTTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6971: +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6972: THTTTPContentEncoding', 'record Value:THTTTPContentEncodingEnum;Custom:Ansistr;end;
6973: THTTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6974: THTTTPContentEncodingField', 'record Value : THTTTPContentEncoding'
6975: +'FieldEnum; List : array of THTTTPContentEncoding; end
6976: THTTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6977: THTTTPRetryAfterField', 'record Value : THTTTPRetryAfterFieldEnum;'
6978: +' Custom : Ansistr; Date : TDateTime; Seconds : Int64; end
6979: THTTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6980: THTTTPContentRangeField', 'record Value : THTTTPContentRangeFieldE'
6981: +num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom: Ansistr; end
6982: THTTTPSetCookieFieldEnum', '( hskoNone, hskoDecoded, hskoCustom )
6983: THTTTPSetCookieCustomField', 'record Name : Ansistr; Value : Ansistr; end
6984: THTTTPSetCookieCustomFieldArray', 'array of THTTTPSetCookieCustomField
6985: THTTTPSetCookieField', 'record Value : THTTTPSetCookieFieldEnum; D'
6986: +'omain : Ansistr; Path : Ansistr; Expires : THTTTPDateField; MaxAge : '
6987: +'Int64; HttpOnly :Bool; Secure :Bool; CustomFields : THTTTPSetCookie'
6988: +'CustomFieldArray; Custom : Ansistr; end
6989: //PHTTTPSetCookieField', '^THTTTPSetCookieField // will not work
6990: THTTTPSetCookieFieldArray', 'array of THTTTPSetCookieField
6991: THTTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6992: THTTTPCookieFieldEntry', 'record Name : Ansistr; Value : Ansistr; end
6993: //PHTTTPCookieFieldEntry', '^THTTTPCookieFieldEntry // will not work
6994: THTTTPCookieFieldEntryArray', 'array of THTTTPCookieFieldEntry
6995: THTTTPCookieField', 'record Value : THTTTPCookieFieldEnum; Entries'
6996: +' : THTTTPCookieFieldEntryArray; Custom : Ansistr; end
6997: THTTTPCommonHeaders', 'record TransferEncoding : THTTTPTransferEnc'
6998: +'oding; ContentType : THTTTPContentType; ContentLength : THTTTPContentLength;'
6999: +' Connection : THTTTPConnectionField; ProxyConnection : THTTTPConnectionField'
7000: +' ; Date : THTTTPDateField; ContentEncoding : THTTTPContentEncodingField; end
7001: THTTTPCustomHeaders', 'array of THTTTPCustomHeader
7002: //THTTTPFixedHeaders', 'array[THTTTPHeaderNameEnum] of Ansistr
7003: THTTTPFixedHeaders', 'array[0..42] of Ansistr
7004: THTTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
7005: +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
7006: THTTTPMethod', 'record Value : THTTTPMethodEnum; Custom : Ansistr; end
7007: THTTTPRequestStartLine', 'record Method:THTTTPMethod;URI:Ansistr;Version:THTTTPVersion;end
7008: THTTTPRequestHeader', 'record CommonHeaders : THTTTPCommonHeaders;'
7009: +' FixedHeaders : THTTTPFixedHeaders; CustomHeaders : THTTTPCustomHeaders; Coo'
7010: +'kie:THTTTPCookieField; IfModifiedSince:THTTTPDateField;IfUnmodifiedSince:THTTTPDateField;end
7011: //PHTTTPRequestHeader', '^THTTTPRequestHeader // will not work
7012: THTTTPRequest', 'record StartLine : THTTTPRequestStartLine; Header'
7013: +' : THTTTPRequestHeader; HeaderComplete :Bool; HasContent :Bool; end
7014: THTTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
7015: THTTTPResponseStartLine', 'record Version : THTTTPVersion; Code : '
7016: +'Int; Msg : THTTTPResponseStartLineMessage; CustomMsg : Ansistr; end
7017: THTTTPResponseHeader', 'record CommonHeaders : THTTTPCommonHeaders'
7018: +' ; FixedHeaders : THTTTPFixedHeaders; CustomHeaders : THTTTPCustomHeaders; Co'
7019: +'okies : THTTTPSetCookieFieldArray; Expires : THTTTPDateField; LastModified : '
7020: +' THTTTPDateField; Age : THTTTPAgeField; end
7021: //PHTTTPResponseHeader', '^THTTTPResponseHeader // will not work
7022: THTTTPResponse', 'record StartLine : THTTTPResponseStartLine; Head'
7023: +'er : THTTTPResponseHeader; HeaderComplete :Bool; HasContent :Bool; end
7024: Func HTTPMessageHasContent( const H : THTTTPCommonHeaders ) :Bool
7025: Proc InitHTTPRequest( var A : THTTTPRequest)
7026: Proc InitHTTPResponse( var A : THTTTPResponse)
7027: Proc ClearHTTPVersion( var A : THTTTPVersion)
7028: Proc ClearHTTPContentLength( var A : THTTTPContentLength)
7029: Proc ClearHTTPContentType( var A : THTTTPContentType)
7030: Proc ClearHTTPDateField( var A : THTTTPDateField)
7031: Proc ClearHTTPTransferEncoding( var A : THTTTPTransferEncoding)
7032: Proc ClearHTTPConnectionField( var A : THTTTPConnectionField)
7033: Proc ClearHTTPPageField( var A : THTTTPPageField)
7034: Proc ClearHTTPContentEncoding( var A : THTTTPContentEncoding)
7035: Proc ClearHTTPContentEncodingField( var A : THTTTPContentEncodingField)
7036: Proc ClearHTTPContentRangeField( var A : THTTTPContentRangeField)
7037: Proc ClearHTTPSetCookieField( var A : THTTTPSetCookieField)
7038: Proc ClearHTTPCommonHeaders( var A : THTTTPCommonHeaders)
7039: //Proc ClearHTTTPFixedHeaders( var A : THTTTPFixedHeaders)
7040: Proc ClearHTTTPCustomHeaders( var A : THTTTPCustomHeaders)
7041: Proc ClearHTTTPCookieField( var A : THTTTPCookieField)
7042: Proc ClearHTTTPMethod( var A : THTTTPMethod)
7043: Proc ClearHTTTPRequestStartLine( var A : THTTTPRequestStartLine)
7044: Proc ClearHTTTPRequestHeader( var A : THTTTPRequestHeader)
7045: Proc ClearHTTTPRequest( var A : THTTTPRequest)
7046: Proc ClearHTTTPResponseStartLine( var A : THTTTPResponseStartLine)
7047: Proc ClearHTTTPResponseHeader( var A : THTTTPResponseHeader)
7048: Proc ClearHTTTPResponse( var A : THTTTPResponse)
7049: THTTTPStringOption', '( hsoNone )
7050: THTTTPStringOptions', 'set of THTTTPStringOption
7051: FindClass('TOBJECT'),'TAnsistrBuilder
7052: Proc BuildStrHTTPVersion(const A:THTTTPVersion;const B:TAnsistrBuilder; PHTTTPStringOptions;
7053: Proc BuildStrHTTPContentLengthValue(const A:THTTTPContentLength;B:TAnsistrBuilder;P:THTTTPStringOptions)
7054: Proc BuildStrHTTPContentLength(const A : THTTTPContentLength; B:TAnsistrBuilder;PHTTTPStringOptions)

```

```

7055: Proc BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsistrBuilder;const P:THTTPStringOptions)
7056: Proc BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsistrBuilder; const P:THTTPStringOptions)
7057: Proc BuildStrRFCDateTime(const DOW, Da, Mo, Ye, Ho, Mi, Se: Int; const TZ:Ansistr;const B:TAnsistrBuilder;const
P:THTTPStringOptions)
7058: Proc BuildStrHTTPDateFieldValue(const A: THTTPDateField;const B: TAnsistrBuilder; const
P:THTTPStringOptions)
7059: Proc BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsistrBuilder;const P:THTTPStringOptions);
7060: Proc BuildStrHTTPTransferEncodingValue(const A : THTTPTransferEncoding; const B : TAnsistrBuilder; const
P : THTTPStringOptions)
7061: Proc BuildStrHTTPTransferEncoding(const A:THTTPTransferEncoding;const B:TAnsistrBuilder;const P:
THTTPStringOptions)
7062: Proc BuildStrHTTPContentRangeField(const A:THTTPContentRangeField;const B:TAnsistrBuilder;const P:
THTTPStringOptions)
7063: Proc BuildStrHTTPConnectionFieldValue(const A:THTTPConnectionField;const B:TAnsistrBuilder;const P:
THTTPStringOptions)
7064: Proc BuildStrHTTPConnectionField(const A:THTTPConnectionField;const B:TAnsistrBuilder;const
P:THTTPStringOptions)
7065: Proc BuildStrHTTPPageField(const A:THTTPPageField;const B:TAnsistrBuilder;const P:THTTPStringOptions);
7066: Proc BuildStrHTTPContentEncoding(const A: THTTPContentEncoding; const B: TAnsistrBuilder; const P :
THTTPStringOptions)
7067: Proc BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const B:TAnsistrBuilder;const
P:THTTPStringOptions)
7068: Proc BuildStrHTTPProxyConnectionField(const A:THTTPConnectionField;const B: TAnsistrBuilder;const P:
THTTPStringOptions)
7069: Proc BuildStrHTTPCommonHeaders( const A: THTTPCommonHeaders; const B : TAnsistrBuilder; const P :
THTTPStringOptions)
7070: Proc BuildStrHTTFFixedHeaders(const A:THTTFFixedHeaders;const B:TAnsiStrBuilder;const
P:THTTPStringOptions)
7071: Proc BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsistrBuilder; const P :
THTTPStringOptions)
7072: Proc BuildStrHTTPSetCookieFieldValue(const A:THTTPSetCookieField;const B:TAnsistrBuilder;const P:
THTTPStringOptions)
7073: Proc BuildStrHTTPCookieFieldValue(const A:THTTPCookieField;const B:TAnsistrBuilder;const P:
THTTPStringOptions)
7074: Proc BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const P:THTTPStringOptions);
7075: Proc BuildStrHTTPMethod(const A:THTTPMethod;const B:TAnsistrBuilder;const P: THTTPStringOptions)
7076: Proc BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsistrBuilder; const P :
THTTPStringOptions)
7077: Proc BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsistrBuilder;const
P:THTTPStrOptions);
7078: Proc BuildStrHTTPRequest(const A : THTTPRequest; const B: TAnsistrBuilder; const P : THTTPStringOptions)
7079: Proc BuildStrHTTPResponseCookieFieldArray(const A : THTTPSetCookieFieldArray; const B : TAnsistrBuilder;
const P : THTTPStringOptions)
7080: Proc BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
THTTPStrOptions);
7081: Proc BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
P:THTTPStringOptions);
7082: Proc BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsistrBuilder; const P:THTTPStringOptions);
7083: Func HTTPContentTypeValueToStr( const A : THTTPContentType) : Ansistr
7084: Func HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : Ansistr
7085: Func HTTPCookieFieldValueToStr( const A : THTTPCookieField) : Ansistr
7086: Func HTTPMethodToStr( const A : THTTPMethod) : Ansistr
7087: Func HTTPRequestToStr( const A : THTTPRequest) : Ansistr
7088: Func HTTPResponseToStr( const A : THTTPResponse) : Ansistr
7089: Proc PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const Domain:Ansistr;const
Secure:Bool; THTTTParserHeaderParseFunc,'Func (const HeaderName : THTT
+PHeaderNameEnum; const HeaderPtr : __Pointer) :Bool
7090: SRegister_THTTTParser(CL);
7091: FindClass('TOBJECT'),'THTTPContentDecoder
7092: THTTPContentDecoderProc', 'Proc ( const Sender : THTTPContentDecoder)
7093: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsize )
7094: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent, '
7095: + ' crcsContentCRLF, crcsTrailer, crcsFinished )
7096: THTTPContentDecoderLogEvent, Procedure(const Sender:THTTPContentDecoder;const LogMsg:str)
7097: SRegister_THTTPContentDecoder(CL);
7098: THTTPContentReaderMechanism', '( hcrnEvent, hcrnString, hcrnStream, hcrnFile )
7099: FindClass('TOBJECT'),'THTTPContentReader
7100: THTTPContentReaderProc', 'Proc ( const Sender : THTTPContentReader)
7101: THTTPContentReaderLogEvent', 'Procedure(const Sender:THTTPContentReader;const LogMsg:str;const
LogLevel: Int;
7102: SRegister_THTTPContentReader(CL);
7103: THTTPContentWriterMechanism', '( hctmEvent, hctmString, hctmStream, hctmFile )
7104: FindClass('TOBJECT'),'THTTPContentWriter
7105: THTTPContentWriterLogEvent, Procedure(const Sender:THTTPContentWriter;const LogMsg:Ansistr);
7106: SRegister_THTTPContentWriter(CL);
7107: Proc SelfTestcHTTTPUtils
7108: end;
7109:
7110:
7111: (*-----*)
7112: Proc SRegister_cTLSUtils(CL: TPSPascalCompiler);
7113: begin
7114: 'TLSLibraryVersion', 'String '1.00
7115: 'TLSError_None', 'LongInt'( 0);
7116: 'TLSError_InvalidBuffer', 'LongInt'( 1);
7117: 'TLSError_InvalidParameter', 'LongInt'( 2);
7118: 'TLSError_InvalidCertificate', 'LongInt'( 3);
7119: 'TLSError_InvalidState', 'LongInt'( 4);
7120: 'TLSError_DecodeError', 'LongInt'( 5);
7121: 'TLSError_BadProtocol', 'LongInt'( 6);

```

```

7122: Func TLSErrorMessage( const TLError : Int) :Str
7123:   SRegister_ETLError(CL);
7124:   TLSProtocolVersion', 'record major : Byte; minor : Byte; end
7125:   PTLSPProtocolVersion', '^TLSProtocolVersion // will not work
7126: Proc InitSSLProtocolVersion30( var A : TLSProtocolVersion)
7127: Proc InitTLSProtocolVersion10( var A : TLSProtocolVersion)
7128: Proc InitTLSProtocolVersion11( var A : TLSProtocolVersion)
7129: Proc InitTLSProtocolVersion12( var A : TLSProtocolVersion)
7130: Func IsTLSProtocolVersion( const A, B : TLSProtocolVersion):Bool
7131: Func IsSSL2( const A : TLSProtocolVersion) :Boolean
7132: Func IsSSL3( const A : TLSProtocolVersion) :Boolean
7133: Func IsTLS10( const A : TLSProtocolVersion):Boolean
7134: Func IsTLS11( const A : TLSProtocolVersion):Boolean
7135: Func IsTLS12( const A : TLSProtocolVersion):Boolean
7136: Func IsTLS10OrLater( const A : TLSProtocolVersion) :Bool
7137: Func IsTLS11OrLater( const A : TLSProtocolVersion) :Bool
7138: Func IsTLS12OrLater( const A : TLSProtocolVersion) :Bool
7139: Func IsFutureTLSVersion( const A : TLSProtocolVersion) :Bool
7140: Func IsKnownTLSVersion( const A : TLSProtocolVersion) :Bool
7141: Func TLSProtocolVersionToStr( const A : TLSProtocolVersion):Str
7142: Func TLSProtocolVersionName( const A : TLSProtocolVersion) :Str
7143:   PTLSSRandom', '^TLSSRandom // will not work
7144: Proc InitTLSSRandom( var Random : TLSSRandom)
7145: Func TLSSRandomToStr( const Random : TLSSRandom) : AnsiStr
7146: 'TLSSessionIDMaxLen', 'LongInt'( 32);
7147: Proc InitTLSSessionID( var SessionID : TLSSessionID; const A : AnsiStr)
7148: Func EncodeTLSSessionID( var Buff:Str; const Size:Int; const SessionID:TLSSessionID):Int;
7149: Func DecodeTLSSessionID( const Buff:Str; const Size:Int; var SessionID:TLSSessionID):Int;
7150:   TLSSSignatureAndHashAlgorithm', 'record Hash : TLSSHashAlgorithm'
7151:   +'; Signature : TLSSSignatureAlgorithm; end
7152: // PTLSSSignatureAndHashAlgorithm', '^TLSSSignatureAndHashAlgorithm +''' will not work
7153:   TLSSSignatureAndHashAlgorithmArray', 'array of TLSSSignatureAndHashAlgorithm
7154:   TLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
7155:   +DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
7156:   TLSSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
7157:   +HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
7158:   TLSSMACAlgorithmInfo', 'record Name:AnsiStr; DigestSize:Integer; Supported:Bool; end
7159:   PTLSSMACAlgorithmInfo', '^TLSSMACAlgorithmInfo // will not work
7160: 'TLS_MAC_MAXDIGESTSIZE', 'LongInt'( 64);
7161:   TLSPRFAlgorithm', '( tpspaSHA256 )
7162: Func tpsP_MD5( const Secret, Seed : AnsiStr; const Size : Int) : AnsiStr
7163: Func tpsP_SHA1( const Secret, Seed : AnsiStr; const Size : Int) : AnsiStr
7164: Func tpsP_SHA256( const Secret, Seed : AnsiStr; const Size : Int) : AnsiStr
7165: Func tpsP_SHA512( const Secret, Seed : AnsiStr; const Size : Int) : AnsiStr
7166: Func tps10PRF( const Secret, ALabel, Seed : AnsiStr; const Size : Int) : AnsiStr
7167: Func tps12PRF_SHA256( const Secret, ALabel, Seed:AnsiStr; const Size : Int): AnsiStr
7168: Func tps12PRF_SHA512( const Secret, ALabel, Seed:AnsiStr; const Size: Int): AnsiStr
7169: Func TLSPRF( const ProtocolVersion:TLSProtocolVersion; const Secret, ALabel, Seed:AStr; const
Size:Int):AStr;
7170: Func tps10KeyBlock( const MasterSecret, ServerRandom, ClientRandom:AnsiStr; const Size:Int):AnsiStr
7171: Func tps12SHA256KeyBlock( const MasterSecret, ServerRandom, ClientRandom: AnsiStr; const Size:Int):AnsiStr;
7172: Func tps12SHA512KeyBlock( const MasterSecret, ServerRandom, ClientRandom: AnsiStr; const Size:Int):AnsiStr;
7173: Func TLSKeyBlock( const ProtocolVersion:TLSProtocolVersion; const MasterSecret, ServerRandom, ClientRandom:
AnsiStr; const Size:Int):AnsiStr
7174: Func tps10MasterSecret( const PreMasterSecret, ClientRandom, ServerRandom:AnsiStr) : AnsiStr;
7175: Func tps12SHA256MasterSecret( const PreMasterSecret, ClientRandom, ServerRandom:AnsiStr):AnsiStr;
7176: Func tps12SHA512MasterSecret( const PreMasterSecret, ClientRandom, ServerRandom:AnsiStr) : AnsiStr;
7177: Func TLSPMasterSecret( const ProtocolVersion:TLSProtocolVersion; const PreMasterSecret, ClientRandom,
ServerRandom:AnsiStr):AnsiStr
7178:   TLSSKeys', 'record KeyBlock : AnsiStr; ClientMACKey : AnsiStr'
7179:   +'; ServerMACKey : AnsiStr; ClientEncKey : AnsiStr; ServerEncKey : '
7180:   +AnsiStr; ClientIV : AnsiStr; ServerIV : AnsiStr; end
7181: Proc GenerateTLSSKeys( const ProtocolVersion:TLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits:Int; const MasterSecret, ServerRandom, ClientRandom:AnsiStr; var TLSSKeys: TLSSKeys)
7182: Proc GenerateFinalTLSSKeys( const ProtocolVersion:TLSProtocolVersion; const IsExportable:Bool; const
ExpandedKeyBits:Int; const ServerRandom, ClientRandom:AnsiStr; var TLSSKeys:TLSSKeys)
7183: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE', 'LongInt'( 16384 - 1);
7184: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE', 'LongInt'( 16384 + 1024);
7185: Proc SelfTestcTLSSUtils
7186: end;
7187:
7188: (*-----*)
7189: Proc SRegister_Reversi(CL: TPSPascalCompiler);
7190: begin
7191:   sPosData', 'record
corner:boolean; square2x2:boolean; edge:boolean; stable:Integer; internal:Integer; disks:Integer; mx:Integer; my:Integer; end
7192:   // pBoard', '^tBoard // will not work
7193:   Func rCalculateData( cc : byte; cx, cy : Integer) : sPosData
7194:   Func rCheckMove( color : byte; cx, cy : Integer) : Integer
7195:   //Func rDoStep( data : pBoard) : word
7196:   Func winExecAndWait( const sAppPath :Str; wVisible : word) :Bool
7197: end;
7198:
7199: Proc SRegister_IWDBCommon(CL: TPSPascalCompiler);
7200: begin
7201:   Func InEditMode( ADataSet : TDataSet) :Bool
7202:   Func CheckDataSource( ADataSource : TDataSource) :Bool;
7203:   Func CheckDataSource1( ADataSource:TDataSource; const AFieldName:Str; var VField:TField):bool;
7204:   Func GetFieldText( AField:TField) :Str

```

```

7205: end;
7206:
7207: Proc SIRegister_SortGrid(CL: TPSPascalCompiler);
7208: begin
7209:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7210:   TMyPrintRange', '( prAll, prSelected )
7211:   TSortStyle', '(ssAutomatic,ssNormal,ssNumeric,ssNumericExtended,ssDateTime,ssTime,ssCustom )
7212:   TSortDirection', '( sdAscending, sdDescending )
7213:   TSetChecked', 'Proc ( Sender : TObject; ACol, ARow : Int; State :Bool)
7214:   TGetCombobox', 'Proc ( Sender : TObject; ACol, ARow : integ'
7215:   +er; var Strs : TStringList; var Width, Height : Int; var Sorted :Bool)
7216:   TSetCombobox', 'Proc ( Sender : TObject; ACol, ARow : Int; Str :Str)
7217:   TSetEllipsis', 'Proc ( Sender : TObject; ACol, ARow : Int)
7218:   SIRegister_TSortOptions(CL);
7219:   SIRegister_TPrintOptions(CL);
7220:   TSortedListEntry', 'record Str :Str; RowNum : Int; SortOption : TSortOptions; end
7221:   SIRegister_TSortedList(CL);
7222:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7223:   TCellBevel', 'record Style: TCellBevelStyle;UpperLeftColor:TColor;LowerRightColor:TColor;end
7224:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7225:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7226:   +Horz : TAlignment;AlignmentVert:TVertAlignment;Bevel:TCellBevel;HideText:Boolean; end
7227:   SIRegister_TFontSetting(CL);
7228:   SIRegister_TFontList(CL);
7229:   AddTypeS(TFormatDrawCellEvent', 'Proc ( Sender : TObject; Col, Row :
7230: +Int;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7231:   TSetFilterEvent', 'Proc ( ARows : TStringList; var Accept :Bool)
7232:   TSearchEvent', 'Proc ( ARows : TStringList; var Accept :Bool)
7233:   TUpdateGridEvent', 'Proc ( Sender : TObject; ARow : Int)
7234:   TSizeChangedEvent', 'Proc ( Sender : TObject; OldColCount, OldRowCount : Int)
7235:   TBeginSortEvent', 'Proc ( Sender : TObject; var Col : Int)
7236:   TEndSortEvent', 'Proc ( Sender : TObject; Col : Int)
7237:   TGetSortStyleEvent', 'Proc ( Sender : TObject; Col : integer; var SortStyle : TSortStyle)
7238:   TCellValidateEvent', 'Proc ( Sender : TObject; ACol, ARow :
7239:   + Int; const OldValue :Str; var NewValue :Str; var Valid :Bool)
7240:   SIRegister_TSortGrid(CL);
7241:   Func ExtendedCompare( const Str1, Str2 :Str) : Int
7242:   Func NormalCompare( const Str1, Str2 :Str) : Int
7243:   Func DateTimeCompare( const Str1, Str2 :Str) : Int
7244:   Func NumericCompare( const Str1, Str2 :Str) : Int
7245:   Func TimeCompare( const Str1, Str2 :Str) : Int
7246:   //Func Compare( Item1, Item2 : Pointer) : Int
7247: end;
7248:
7249: ***** Proc Register_IB(CL: TPSPascalCompiler);
7250: Proc IBAlloc( var P, OldSize, NewSize : Int)
7251: Proc IBError( ErrMess : TIBClientError; const Args : array of const)
7252: Proc IBDataBaseError
7253: Func StatusVector : PISC_STATUS
7254: Func StatusVectorArray : PStatusVector
7255: Func CheckStatusVector( ErrorCodes : array of ISC_STATUS) :Bool
7256: Func StatusVectorAsText :Str
7257: Proc SetIBDataBaseErrorMessage( Value : TIBDataBaseErrorMessage)
7258: Func GetIBDataBaseErrorMessage : TIBDataBaseErrorMessage
7259:
7260: //*****unit uPSI_BoldUtils;*****
7261: Func CharCount( c : char; const s :Str) : Int
7262: Func BoldNamesEqual( const name1, name2 :Str) :Bool
7263: Proc BoldAppendToStrings(strings:TStrings; const aString:str;const ForceNewLine:Bool)
7264: Func BoldSeparateStringList(strings:TStringList;const Separatr,PreString,PostString:Str)Str
7265: Func BoldSeparatedAppend( const S1, S2 :Str; const Separator :Str) :Str
7266: Func BoldTrim( const S :Str) :Str
7267: Func BoldIsPrefix( const S, Prefix :Str) :Bool
7268: Func BoldStrEqual( P1, P2 : PChar; Len : Int) :Bool
7269: Func BoldStrAnsiEqual( P1, P2 : PChar; Len : Int) :Bool
7270: Func BoldAnsiEqual( const S1, S2 :Str) :Bool
7271: Func BoldStrStringEqual( const S1 :Str; P2 : PChar; Len : Int) :Bool
7272: Func BoldCaseIndependentPos( const Substr, S :Str) : Int
7273: //Proc EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7274: Func CapitalisedToSpaced( Capitalised :Str) :Str
7275: Func SpacedToCapitalised( Spaced :Str) :Str
7276: Func BooleanToString( BoolValue :Bool) :Str
7277: Func StringToBoolean( StrValue :Str) :Bool
7278: Func GetUpperLimitForMultiplicity( const Multiplicity :Str) : Int
7279: Func GetLowerLimitForMultiplicity( const Multiplicity :Str) : Int
7280: Func StringListToVarArray( List : TStringList) : variant
7281: Func IsLocalMachine( const Machinename : WideString) :Bool
7282: Func GetComputerNameStr :Str
7283: Func TimeStampComp( const Time1, Time2 : TTimeStamp) : Int
7284: Func BoldStrToDateFmt(const S:str;const DateFormat:str;const DateSeparatorChar:char):TDateTime
7285: Func BoldDateToStrFmt(const aDate:TDateTime;DateFormat:str;const DateSeparatorChar:char):Str;
7286: Func BoldParseFormattedDateList(const value:str;const formats:TStrings;var Date:TDateTime):Boolean;
7287: Func BoldParseFormattedDate(const value:str;const formats:array of string; var Date:TDateTime):Boolean;
7288: Proc EnsureTrailing( var Str :Str; ch : char)
7289: Func BoldDirectoryExists( const Name :Str) :Bool
7290: Func BoldForceDirectories( Dir :Str) :Bool
7291: Func BoldRootRegistryKey :Str
7292: Func GetModuleFileNameAsString( IncludePath :Bool) :Str
7293: Func BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Int

```



```

7294: Func LogicalAnd( A, B : Int) :Bool
7295: record TByHandleFileInformation dwFileAttributes: DWORD; '
7296:   +ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7297:   +: TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7298:   +eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow:DWORD; end
7299: Func GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7300: Func IsFirstInstance :Bool
7301: Proc ActivateFirst( AString : PChar)
7302: Proc ActivateFirstCommandLine
7303: Func MakeAckPkt(const BlockNumber: Word):Str;
7304: Proc SendError(UDPBase:TIdUDPBase;APeerIP:str;const APort:Int;const ErrNumb:Word;ErrStr:str;
7305: Proc SendError(UDPCliet:TIdUDPCliet;const ErrNumber:Word; ErrorString:str);overload;
7306: Proc SendError(UDPBase:TIdUDPBase;APeerIP:str;const APort:Int; E:Exception);overload;
7307: Proc SendError(UDPCliet: TIdUDPCliet; E: Exception); overload;
7308: Func IdStrToWord(const Value:Str): Word;
7309: Func IdWordToStr(const Value: Word): WordStr;
7310: Func HasInstructionSet( const InstructionSet : TCPUInstructionSet) :Bool
7311: Func CPUFeatures : TCPUFeatures
7312:
7313: Proc SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7314: begin
7315:   AddTypeS('TXRTLBitIndex', 'Int
7316: Func XRTLSwapBits( Data :Card; Bit1Index, Bit2Index : TXRTLBitIndex) :Card
7317: Func XRTLBitTest( Data :Card; BitIndex : TXRTLBitIndex) :Bool
7318: Func XRTLBitSet( Data :Card; BitIndex : TXRTLBitIndex) :Card
7319: Func XRTLBitReset( Data :Card; BitIndex : TXRTLBitIndex) :Card
7320: Func XRTLBitComplement( Data :Card; BitIndex : TXRTLBitIndex) :Card
7321: Func XRTLSwapHiLo16( X : Word) : Word
7322: Func XRTLSwapHiLo32( X :Card) :Card
7323: Func XRTLSwapHiLo64( X : Int64) : Int64
7324: Func XRTLROL32( A, S :Card) :Card
7325: Func XRTLROL32( A, S :Card) :Card
7326: Func XRTLROL16( A : Word; S :Card) : Word
7327: Func XRTLROL16( A : Word; S :Card) : Word
7328: Func XRTLROL8( A : Byte; S :Card) : Byte
7329: Func XRTLROL8( A : Byte; S :Card) : Byte
7330: //Proc XRTLXorBlock( I1, I2, O1 : PByteArray; Len : Int)
7331: //Proc XRTLIncBlock( P : PByteArray; Len : Int)
7332: Proc XRTLUMul64( const A, B : Int; var MulL, MulH : Int)
7333: Func XRTLPopulation( A :Card):Card
7334: end;
7335: Func XRTLURLDecode( const ASrc : WideString) : WideString
7336: Func XRTLURLEncode( const ASrc : WideString) : WideString
7337: Func XRTLURINormalize( const AURI : WideString) : WideString
7338: Proc XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
VPassword:WideString)
7339: Func XRTLExtractLongPathName(APath:Str):Str;
7340:
7341: Proc SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7342: begin
7343:   Int8', 'ShortInt
7344:   AddTypeS('Int16', 'SmallInt
7345:   Int32', 'LongInt
7346:   UInt8', 'Byte
7347:   UInt16', 'Word
7348:   UInt32', 'LongWord
7349:   UInt64', 'Int64
7350:   Word8', 'UInt8
7351:   Word16', 'UInt16
7352:   Word32', 'UInt32
7353:   Word64', 'UInt64
7354:   LargeInt', 'Int64
7355:   NativeInt', 'Int
7356:   AddTypeS('NativeUInt', 'Cardinal
7357:   Const('BitsPerByte','LongInt'( 8);
7358:   Const('BitsPerWord','LongInt'( 16);
7359:   Const('BitsPerLongWord','LongInt'( 32);
7360:   //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7361:   //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7362: Func MinI( const A,B Int) : Int
7363: Func MaxI( const A,B:Int) : Int
7364: Func MinC( const A,B:Card) :Card
7365: Func MaxC( const A,B:Card) :Card
7366: Func SumClipI( const A, I : Int) : Int
7367: Func SumClipC( const A :Card; const I : Int) :Card
7368: Func InByteRange( const A : Int64) :Bool
7369: Func InWordRange( const A : Int64) :Bool
7370: Func InLongWordRange( const A : Int64) : Bool
7371: Func InShortIntRange( const A : Int64) : Bool
7372: Func InSmallIntRange( const A : Int64) : Bool
7373: Func InLongIntRange( const A : Int64) : Bool
7374: AddTypeS('Bool8', 'ByteBool
7375: AddTypeS('Bool16', 'WordBool
7376: AddTypeS('Bool32', 'LongBool
7377: AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7378: AddTypeS('TCompareResultSet', 'set of TCompareResult
7379: Func ReverseCompareResult( const C : TCompareResult) : TCompareResult
7380: Const('MinSingle','Single').setExtended( 1.5E-45);
7381: Const('MaxSingle','Single').setExtended( 3.4E+38);

```

```

7382: Const('MinDouble','Double').setExtended( 5.0E-324);
7383: Const('MaxDouble','Double').setExtended( 1.7E+308);
7384: Const('MinExtended','Extended').setExtended(3.4E+4932);
7385: Const('MaxExtended','Extended').setExtended(1.1E+4932);
7386: Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7387: Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7388: Func MinF( const A, B : Float) : Float
7389: Func MaxF( const A, B : Float) : Float
7390: Func ClipF( const Value : Float; const Low, High : Float) : Float
7391: Func InSingleRange( const A : Float) :Bool
7392: Func InDoubleRange( const A : Float) :Bool
7393: Func InCurrencyRange( const A : Float) :Bool;
7394: Func InCurrencyRange1( const A : Int64) :Bool;
7395: Func FloatExponentBase2( const A : Extended; var Exponent : Int) :Bool
7396: Func FloatExponentBase10( const A : Extended; var Exponent : Int):Bool
7397: Func FloatIsInfinity( const A : Extended) :Bool
7398: Func FloatIsNaN( const A : Extended) :Bool
7399: Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7400: Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7401: Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7402: Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7403: Func FloatZero( const A : Float; const CompareDelta : Float) :Bool
7404: Func FloatOne( const A : Float; const CompareDelta : Float) :Bool
7405: Func FloatsEqual( const A, B : Float; const CompareDelta : Float) :Bool
7406: Func FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7407: Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7408: Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7409: Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7410: Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7411: Func ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) :Bool
7412: Func ApproxCompare( const A, B : Extended; const CompareEpsilon : Double) : TCompareResult
7413: Func cClearBit( const Value, BitIndex : LongWord) : LongWord
7414: Func cSetBit( const Value, BitIndex : LongWord) : LongWord
7415: Func cIsBitSet( const Value, BitIndex : LongWord) :Bool
7416: Func cToggleBit( const Value, BitIndex : LongWord) : LongWord
7417: Func cIsHighBitSet( const Value : LongWord) :Bool
7418: Func SetBitScanForward( const Value : LongWord) : Int;
7419: Func SetBitScanForward1( const Value, BitIndex : LongWord) : Int;
7420: Func SetBitScanReverse( const Value : LongWord) : Int;
7421: Func SetBitScanReverse1( const Value, BitIndex : LongWord) : Int;
7422: Func ClearBitScanForward( const Value : LongWord) : Int;
7423: Func ClearBitScanForward1( const Value, BitIndex : LongWord) : Int;
7424: Func ClearBitScanReverse( const Value : LongWord) : Int;
7425: Func ClearBitScanReverse1( const Value, BitIndex : LongWord) : Int;
7426: Func cReverseBits( const Value : LongWord) : LongWord;
7427: Func cReverseBits1( const Value : LongWord; const BitCount : Int) : LongWord;
7428: Func cSwapEndian( const Value : LongWord) : LongWord
7429: Func cTwosComplement( const Value : LongWord) : LongWord
7430: Func RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7431: Func RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7432: Func RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7433: Func RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7434: Func cBitCount( const Value : LongWord) : LongWord
7435: Func cIsPowerOfTwo( const Value : LongWord) :Bool
7436: Func LowBitMask( const HighBitIndex : LongWord) : LongWord
7437: Func HighBitMask( const LowBitIndex : LongWord) : LongWord
7438: Func RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7439: Func SetBitRange( const Value:LongWord;const LowBitIndex,HighBitIndex: LongWord) : LongWord
7440: Func ClearBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord) : LongWord
7441: Func ToggleBitRange(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):LongWord
7442: Func IsBitRangeSet(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):Bool
7443: Func IsBitRangeClear(const Value:LongWord;const LowBitIndex,HighBitIndex:LongWord):Boolean
7444: // AddTypeS('CharSet', 'set of AnsiChar
7445: AddTypeS('CharSet', 'set of Char' //!!!
7446: AddTypeS('AnsiCharSet', 'TCharSet
7447: AddTypeS('ByteSet', 'set of Byte
7448: AddTypeS('AnsiChar', 'Char
7449: // Func AsCharSet( const C : array of AnsiChar) : CharSet
7450: Func AsByteSet( const C : array of Byte) : ByteSet
7451: Proc ComplementChar( var C : CharSet; const Ch : Char)
7452: Proc ClearCharSet( var C : CharSet)
7453: Proc FillCharSet( var C : CharSet)
7454: Proc FillCharSearchRec; // with 0
7455: Proc ComplementCharSet( var C : CharSet)
7456: Proc AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7457: Proc Union( var DestSet : CharSet; const SourceSet : CharSet)
7458: Proc Difference( var DestSet : CharSet; const SourceSet : CharSet)
7459: Proc Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7460: Proc XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7461: Func IsSubSet( const A, B : CharSet) :Bool
7462: //Func IsEqual( const A, B : CharSet) :Bool
7463: Func IsEqualCharSet( const A, B : CharSet) :Bool
7464: Func IsEqual( const D1, D2 : TDateTime) :Bool;
7465: Func IsEmpty( const C : CharSet) :Bool
7466: Func IsEmptyCharSet( const C : CharSet) :Bool
7467: Func IsComplete( const C : CharSet) :Bool
7468: Func cCharCount( const C : CharSet) : Int
7469: Proc ConvertCaseInsensitive( var C : CharSet)
7470: Func CaseInsensitiveCharSet( const C : CharSet) : CharSet

```

```

7471: Func IntRangeLength( const Low, High : Int ) : Int64
7472: Func IntRangeAdjacent( const Low1, High1, Low2, High2 : Int ) : Bool
7473: Func IntRangeOverlap( const Low1, High1, Low2, High2 : Int ) : Bool
7474: Func IntRangeHasElement( const Low, High, Element : Int ) : Bool
7475: Func IntRangeIncludeElement( var Low, High : Int; const Element : Int ) : Bool
7476: Func IntRangeIncludeElementRange( var Low, High: Int; const LowElement, HighElement: Int ) : Boolean
7477: Func CardRangeLength( const Low, High : Card ) : Int64
7478: Func CardRangeAdjacent( const Low1, High1, Low2, High2 : Card ) : Bool
7479: Func CardRangeOverlap( const Low1, High1, Low2, High2 : Card ) : Bool
7480: Func CardRangeHasElement( const Low, High, Element : Card ) : Bool
7481: Func CardRangeIncludeElement( var Low, High : Card; const Element : Card ) : Bool
7482: Func CardRangeIncludeElementRange( var Low, High: Card; const LowElement, HighElement: Card ) : Bool;
7483: AddTypeS('UnicodeChar', 'WideChar
7484: Func Compare( const I1, I2 : Bool ) : TCompareResult;
7485: Func Compare1( const I1, I2 : Int ) : TCompareResult;
7486: Func Compare2( const I1, I2 : Int64 ) : TCompareResult;
7487: Func Compare3( const I1, I2 : Extended ) : TCompareResult;
7488: Func CompareA( const I1, I2 : Ansistr ) : TCompareResult
7489: Func CompareW( const I1, I2 : WideString ) : TCompareResult
7490: Func cSgn( const A : LongInt ) : Int;
7491: Func cSgn1( const A : Int64 ) : Int;
7492: Func cSgn2( const A : Extended ) : Int;
7493: AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7494: Func AnsiCharToInt( const A : AnsiChar ) : Int
7495: Func WideCharToInt( const A : WideChar ) : Int
7496: Func CharToInt( const A : Char ) : Int
7497: Func IntToAnsiChar( const A : Int ) : AnsiChar
7498: Func IntToWideChar( const A : Int ) : WideChar
7499: Func IntToChar( const A : Int ) : Char
7500: Func IsHexAnsiChar( const Ch : AnsiChar ) : Bool
7501: Func IsHexWideChar( const Ch : WideChar ) : Bool
7502: Func IsHexChar( const Ch : Char ) : Bool
7503: Func HexAnsiCharToInt( const A : AnsiChar ) : Int
7504: Func HexWideCharToInt( const A : WideChar ) : Int
7505: Func HexCharToInt( const A : Char ) : Int
7506: Func IntToUpperHexAnsiChar( const A : Int ) : AnsiChar
7507: Func IntToUpperHexWideChar( const A : Int ) : WideChar
7508: Func IntToUpperHexChar( const A : Int ) : Char
7509: Func IntToLowerHexAnsiChar( const A : Int ) : AnsiChar
7510: Func IntToLowerHexWideChar( const A : Int ) : WideChar
7511: Func IntToLowerHexChar( const A : Int ) : Char
7512: Func IntToStringA( const A : Int64 ) : Ansistr
7513: Func IntToStringW( const A : Int64 ) : WideString
7514: Func IntToString( const A : Int64 ) : Str
7515: Func UIntToStringA( const A : NativeUInt ) : Ansistr
7516: Func UIntToStringW( const A : NativeUInt ) : WideString
7517: Func UIntToString( const A : NativeUInt ) : Str
7518: Func LongWordToStrA( const A : LongWord; const Digits : Int ) : Ansistr
7519: Func LongWordToStrW( const A : LongWord; const Digits : Int ) : WideString
7520: Func LongWordToStrU( const A : LongWord; const Digits : Int ) : UnicodeString
7521: Func LongWordToStr( const A : LongWord; const Digits : Int ) : Str
7522: Func LongWordToHexA( const A: LongWord; const Digits: Int; const UpperCase: Bool ) : Ansistr;
7523: Func LongWordToHexW( const A: LongWord; const Digits: Int; const UpperCase: Bool ) : WideString;
7524: Func LongWordToHex( const A : LongWord; const Digits : Int; const UpperCase: Boolean ) : str
7525: Func LongWordToOctA( const A : LongWord; const Digits : Int ) : Ansistr
7526: Func LongWordToOctW( const A : LongWord; const Digits : Int ) : WideString
7527: Func LongWordToOct( const A : LongWord; const Digits : Int ) : Str
7528: Func LongWordToBinA( const A : LongWord; const Digits : Int ) : Ansistr
7529: Func LongWordToBinW( const A : LongWord; const Digits : Int ) : WideString
7530: Func LongWordToBin( const A : LongWord; const Digits : Int ) : Str
7531: Func TryStringToInt64A( const S : Ansistr; out A : Int64 ) : Bool
7532: Func TryStringToInt64W( const S : WideString; out A : Int64 ) : Bool
7533: Func TryStringToInt64( const S : Str; out A : Int64 ) : Bool
7534: Func StringToInt64DefA( const S : Ansistr; const Default : Int64 ) : Int64
7535: Func StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7536: Func StringToInt64Def( const S : Str; const Default : Int64 ) : Int64
7537: Func StringToInt64A( const S : Ansistr ) : Int64
7538: Func StringToInt64W( const S : WideString ) : Int64
7539: Func StringToInt64( const S : Str ) : Int64
7540: Func TryStringToIntA( const S : Ansistr; out A : Int ) : Bool
7541: Func TryStringToIntW( const S : WideString; out A : Int ) : Bool
7542: Func TryStringToInt( const S : Str; out A : Int ) : Bool
7543: Func StringToIntDefA( const S : Ansistr; const Default : Int ) : Int
7544: Func StringToIntDefW( const S : WideString; const Default : Int ) : Int
7545: Func StringToIntDef( const S : Str; const Default : Int ) : Int
7546: Func StringToIntA( const S : Ansistr ) : Int
7547: Func StringToIntW( const S : WideString ) : Int
7548: Func StringToInt( const S : Str ) : Int
7549: Func TryStringToLongWordA( const S : Ansistr; out A : LongWord ) : Bool
7550: Func TryStringToLongWordW( const S : WideString; out A : LongWord ) : Bool
7551: Func TryStringToLongWord( const S : Str; out A : LongWord ) : Bool
7552: Func StringToLongWordA( const S : Ansistr ) : LongWord
7553: Func StringToLongWordW( const S : WideString ) : LongWord
7554: Func StringToLongWord( const S : Str ) : LongWord
7555: Func HexToUIntA( const S : Ansistr ) : NativeUInt
7556: Func HexToUIntW( const S : WideString ) : NativeUInt
7557: Func HexToUInt( const S : Str ) : NativeUInt
7558: Func TryHexToLongWordA( const S : Ansistr; out A : LongWord ) : Bool
7559: Func TryHexToLongWordW( const S : WideString; out A : LongWord ) : Bool

```

```

7560: Func TryHexToLongWord( const S :Str; out A : LongWord) :Bool
7561: Func HexToLongWordA( const S : Ansistr) : LongWord
7562: Func HexToLongWordW( const S : WideString) : LongWord
7563: Func HexToLongWord( const S :Str) : LongWord
7564: Func TryOctToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7565: Func TryOctToLongWordW( const S : WideString; out A : LongWord) :Bool
7566: Func TryOctToLongWord( const S :Str; out A : LongWord) :Bool
7567: Func OctToLongWordA( const S : Ansistr) : LongWord
7568: Func OctToLongWordW( const S : WideString) : LongWord
7569: Func OctToLongWord( const S :Str) : LongWord
7570: Func TryBinToLongWordA( const S : Ansistr; out A : LongWord) :Bool
7571: Func TryBinToLongWordW( const S : WideString; out A : LongWord) :Bool
7572: Func TryBinToLongWord( const S :Str; out A : LongWord) :Bool
7573: Func BinToLongWordA( const S : Ansistr) : LongWord
7574: Func BinToLongWordW( const S : WideString) : LongWord
7575: Func BinToLongWord( const S :Str) : LongWord
7576: Func FloatToStringA( const A : Extended) : Ansistr
7577: Func FloatToStringW( const A : Extended) : WideString
7578: Func FloatToString( const A : Extended) :Str
7579: Func FloatToStringA( const A : Extended) :Str
7580: Func TryStringToFloatA( const A : Ansistr; out B : Extended) :Bool
7581: Func TryStringToFloatW( const A : WideString; out B : Extended) :Bool
7582: Func TryStringToFloat( const A :Str; out B : Extended) :Bool
7583: Func StringToFloatA( const A : Ansistr) : Extended
7584: Func StringToFloatW( const A : WideString) : Extended
7585: Func StringToFloat( const A :Str) : Extended
7586: Func StringToFloatDefA( const A : Ansistr; const Default : Extended) : Extended
7587: Func StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7588: Func StringToFloatDef( const A :Str; const Default : Extended) : Extended
7589: Func EncodeBase64( const S,Alphabet:Ansistr;const Pad:Bool;const PadMultiple:Int;const
PadChar:AnsiChar):Ansistr
7590: Func DecodeBase64( const S, Alphabet:Ansistr;const PadSet:CharSet):Ansistr
7591: unit uPSI_cFundamentUtils;
7592: Const('b64_MIMEBase64',Str).String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7593: Const('b64_UUEncode','String').String('!#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNQRSTUUVWXYZ[\]^_';
7594: Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvwxyz';
7595: Const('CCHARSET','Stringb64_XXEncode);
7596: Const('CHEXSET','String'0123456789ABCDEF
7597: Const('HEXDIGITS','String'0123456789ABCDEF
7598: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7599: Const('DIGISET','String'0123456789
7600: Const('LETTERSET','String'ABCDEFGHIJKLMNQRSTUUVWXYZ'
7601: Const('DIGISET2','TCharSet').SetSet('0123456789'
7602: Const('LETTERSET2','TCharSet').SetSet('ABCDEFGHIJKLMNQRSTUUVWXYZ'
7603: Const('HEXSET2','TCharSet').SetSet('0123456789ABCDEF;
7604: Const('NUMBERSET','TCharSet').SetSet('0123456789;
7605: Const('NUMBERS','String'0123456789;
7606: Const('LETTERS','String'ABCDEFGHIJKLMNQRSTUUVWXYZ;
7607: Const('NUMBLETTTS','String').SetString('0123456789ABCDEFGHIJKLMNQRSTUUVWXYZ;
7608: Const('NUMBLETTSET','TCharSet').SetSet('0123456789ABCDEFGHIJKLMNQRSTUUVWXYZ;
7609: Const('ALPHANUMSET','TCharSet').SetSet('0123456789ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvwxyz);
7610: tantN('ALPHANUMSET','TCharSet').SetSet('0123456789ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvwxyz);
7611: Const('ALPHANUMSTR','String').SetString('0123456789ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvwxyz);
7612:
7613: Func CharSetToStr( const C : CharSet) : Ansistr
7614: Func StrToCharSet( const S : Ansistr) : CharSet
7615: Func MIMEBase64Decode( const S : Ansistr) : Ansistr
7616: Func MIMEBase64Encode( const S : Ansistr) : Ansistr
7617: Func UUDDecode( const S: Ansistr) : Ansistr
7618: Func XXDecode( const S Ansistr) : Ansistr
7619: Func BytesToHex( const P : array of Byte; const UpperCase :Bool) : Ansistr
7620: Func InterfaceToStrA( const I : IInterface) : Ansistr
7621: Func InterfaceToStrW( const I : IInterface) : WideString
7622: Func InterfaceToStr( const I : IInterface) :Str
7623: Func ObjectClassName( const O : TObject) :Str
7624: Func ClassClassName( const C : TClass) :Str
7625: Func ObjectToStr( const O : TObject) :Str
7626: Func ObjectToString( const O : TObject) :Str
7627: Func CharSetToStr( const C : CharSet) : Ansistr
7628: Func StrToCharSet( const S : Ansistr) : CharSet
7629: Func HashStrA(const S:Ansistr;const Index:Int;const Count:Int;const AsciiCaseSensitive:Bool;const
Slots:LongWord) :LongWord
7630: Func HashStrW(const S:WideString;const Index:Int;const Count:Int;const AsciiCaseSensitive:Bool;const
Slots:LongWord):LongWord
7631: Func HashStr(const S:str;const Index:Int;const Count:Int;const AsciiCaseSensitive:Bool;const Slots:
LongWord):LongWord
7632: Func HashInt( const I : Int; const Slots : LongWord) : LongWord
7633: Func HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7634: Const('Bytes1KB','LongInt'( 1024);
7635: SIRegister_IInterface(CL);
7636: Proc SelfTestCFundamentUtils
7637:
7638: Func CreateSchedule : IJclSchedule
7639: Func NullStamp : TTimeStamp
7640: Func CompareTimeStamps(const Stamp1, Stamp2 : TTimeStamp) : Int64
7641: Func EqualTimeStamps(const Stamp1, Stamp2 : TTimeStamp) :Bool
7642: Func IsNullTimeStamp(const Stamp : TTimeStamp) :Bool
7643:
7644: Proc SIRegister_uwinplot(CL: TPSPascalCompiler);

```

```

7645: begin
7646:   AddTypeS(TFunc, 'function(X : Float) : Float;
7647: Func InitGraphics( Width, Height : Int) :Bool
7648: Proc SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Int; GraphBorder :Bool)
7649: Proc SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7650: Proc SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7651: Proc GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7652: Proc GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7653: Proc SetGraphTitle( Title :Str)
7654: Proc SetOxTitle( Title :Str)
7655: Proc SetOyTitle( Title :Str)
7656: Func GetGraphTitle :Str
7657: Func GetOxTitle :Str
7658: Func GetOyTitle :Str
7659: Proc PlotOxAxis( Canvas : TCanvas)
7660: Proc PlotOyAxis( Canvas : TCanvas)
7661: Proc PlotGrid( Canvas : TCanvas; Grid : TGrid)
7662: Proc WriteGraphTitle( Canvas : TCanvas)
7663: Func SetMaxCurv( NCurv : Byte) :Bool
7664: Proc SetPointParam( CurvIndex, Symbol, Size : Int; Color : TColor)
7665: Proc SetLineParam(CurvIndex: Int; Style : TPenStyle; Width : Int; Color : TColor)
7666: Proc SetCurvLegend( CurvIndex : Int; Legend :Str)
7667: Proc SetCurvStep( CurvIndex, Step : Int)
7668: Func GetMaxCurv : Byte
7669: Proc GetPointParam( CurvIndex : Int; var Symbol, Size : Int; var Color : TColor)
7670: Proc GetLineParam(CurvIndex: Int; var Style:TPenStyle; var Width: Int; var Color: TColor);
7671: Func GetCurvLegend( CurvIndex : Int) :Str
7672: Func GetCurvStep( CurvIndex : Int) : Int
7673: Proc PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Int)
7674: Proc PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Int)
7675: Proc PlotCurveWithErrorBars(Canvas : TCanvas; X, Y, S: TVector; Ns, Lb, Ub, CurvIndex: Int)
7676: Proc PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin, Xmax: Float; Npt, CurvIndex: Int)
7677: Proc WriteLegend( Canvas : TCanvas; NCurv : Int; ShowPoints, ShowLines :Bool)
7678: Proc ConRec( Canvas : TCanvas; Nx, Ny, Nc : Int; X, Y, Z : TVector; F : TMatrix)
7679: Func Xpixel(X:Float) : Int
7680: Func Ypixel(Y:Float) : Int
7681: Func Xuser( X: Int) : Float
7682: Func Yuser( Y: Int) : Float
7683: end;
7684:
7685: Proc FFT( NumSamples : Int; InArray, OutArray : TCompVector)
7686: Proc IFFT( NumSamples : Int; InArray, OutArray : TCompVector)
7687: Proc FFT_Int( NumSamples : Int; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7688: Proc FFT_Int_Cleanup
7689: Proc CalcFrequency(NumSamples, FrequencyIndex: Int; InArray: TCompVector; var FT : Complex)
7690: //unit uPSI_JclStreams;
7691: Func StreamSeek(Stream: TStream; const Offset: Int64; const Origin : TSeekOrigin) : Int64
7692: Func StreamCopy( Source : TStream; Dest : TStream; BufferSize : Int) : Int64
7693: Func CompareStreams( A, B : TStream; BufferSize : Int) :Bool
7694: Func JCompareFiles( const FileA, FileB : TFileName; BufferSize : Int) :Bool
7695:
7696: Proc SIRegister_FmxUtils(CL: TPSPascalCompiler);
7697: begin
7698:   FindClass('TOBJECT'), 'EInvalidDest
7699:   FindClass('TOBJECT'), 'EFCantMove
7700:   Proc fmxCopyFile( const FileName, DestName :Str)
7701:   Proc fmxMoveFile( const FileName, DestName :Str)
7702:   Func fmxGetFileSize( const FileName :Str) : LongInt
7703:   Func fmxFileDateTime( const FileName :Str) : TDateTime
7704:   Func fmxHasAttr( const FileName :Str; Attr : Word) :Bool
7705:   Func fmxExecuteFile( const FileName, Params, DefaultDir:Str; ShowCmd : Int):THandle;
7706: end;
7707:
7708: Proc SIRegister_FindFileIter(CL: TPSPascalCompiler);
7709: begin
7710:   SIRegister_IFindFileIterator(CL);
7711:   Func CreateFindFile(const Path:str; IncludeAttr: Int; out iffi:IFindFileIterator):Bool;
7712: end;
7713:
7714: Proc SIRegister_PCharUtils(CL: TPSPascalCompiler);
7715: begin
7716:   Func SkipWhite( cp : PChar) : PChar
7717:   Func ReadStringDoubleQuotedMaybe( cp : PChar; var AStr :Str) : PChar
7718:   Func ReadStringSingleQuotedMaybe( cp : PChar; var AStr :Str) : PChar
7719:   Func ReadIdent( cp : PChar; var ident :Str) : PChar
7720: end;
7721:
7722: Proc SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7723: begin
7724:   SIRegister_TStringHashMapTraits(CL);
7725:   Func CaseSensitiveTraits : TStringHashMapTraits
7726:   Func CaseInsensitiveTraits : TStringHashMapTraits
7727:   THashNode, 'record Str:str; Ptr:Pointer; Left:PHashNode; Right:PHashNode; end
7728:   //PHashArray, '^THashArray // will not work
7729:   SIRegister_TStringHashMap(CL);
7730:   THashValue, 'Cardinal
7731:   Func StrHash( const s :Str) : THashValue
7732:   Func TextHash( const s :Str) : THashValue
7733:   Func DataHash( var AValue, ASize :Card) : THashValue

```



```

7734: Func Iterate_FreeObjects(AUserData:Pointer;const AStr:str; var AData:Pointer):Boolean
7735: Func Iterate_Dispose(AUserData: Pointer;const AStr:str; var AData: Pointer):Bool
7736: Func Iterate_FreeMem(AUserData: Pointer;const AStr:str; var AData: Pointer):Bool
7737: SIRegister_TCaseSensitiveTraits(CL);
7738: SIRegister_TCaseInsensitiveTraits(CL);
7739: //*****unit uPSI_umath;
7740: Func uExpo( X : Float) : Float
7741: Func uExp2( X : Float) : Float
7742: Func uExp10( X : Float) : Float
7743: Func uLog( X : Float) : Float
7744: Func uLog2( X : Float) : Float
7745: Func uLog10( X : Float) : Float
7746: Func uLogA( X, A : Float) : Float
7747: Func uIntPower( X : Float; N : Int): Float
7748: Func uPower( X, Y : Float) : Float
7749: Func SgnGamma( X : Float) : Int
7750: Func Stirling( X : Float) : Float
7751: Func StirLog( X : Float) : Float
7752: Func Gamma( X : Float) : Float
7753: Func LnGamma( X : Float) : Float
7754: Func DiGamma( X : Float) : Float
7755: Func TriGamma( X : Float) : Float
7756: Func IGamma( X : Float) : Float
7757: Func JGamma( X : Float) : Float
7758: Func InvGamma( X : Float) : Float
7759: Func Erf( X : Float) : Float
7760: Func Erfc( X : Float) : Float
7761: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float;
7762: { Correlation coefficient between samples X and Y }
7763: Func DBeta(A, B, X : Float) : Float;
7764: { Density of Beta distribution with parameters A and B }
7765: Func LambertW( X : Float; UBranch, Offset :Bool) : Float
7766: Func Beta(X, Y : Float) : Float
7767: Func Binomial( N, K : Int) : Float
7768: Func PBinom( N : Int; P : Float; K : Int) : Float
7769: Proc Cholesky( A, L : TMatrix; Lb, Ub : Int)
7770: Proc LU_Decom( A : TMatrix; Lb, Ub : Int)
7771: Proc LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Int; X : TVector)
7772: Func DNorm( X : Float) : Float
7773: Func DGamma(A, B, X : Float) : Float;
7774: { Density of Gamma distribution with parameters A and B }
7775: Func DKhi2(Nu : Int; X : Float) : Float;
7776: { Density of Khi-2 distribution with Nu d.o.f. }
7777: Func DStudent(Nu : Int; X : Float) : Float;
7778: { Density of Student distribution with Nu d.o.f. }
7779: Func DSnedecor(Nu1, Nu2 : Int; X : Float) : Float;
7780: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7781: Func IBeta(A, B, X : Float) : Float;
7782: { Incomplete Beta function }
7783: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float;
7784:
7785: Proc SIRegister_unlfit(CL: TPSPascalCompiler);
7786: begin Proc SetOptAlgo( Algo : TOptAlgo)
7787: Proc SetOptAlgo(Algo : TOptAlgo);
7788: { -----
7789:   Sets the optimization algorithm according to Algo, which must be
7790:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ }
7791: Func GetOptAlgo : TOptAlgo
7792: Proc SetMaxParam( N : Byte)
7793: Func GetMaxParam : Byte
7794: Proc SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7795: Proc GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7796: Func NullParam( B : TVector; Lb, Ub : Int) :Bool
7797: Proc NLFit(RegFunc:TRegFunc;DerivProc:TDerivProc;X,Y:TVector;Lb
Ub:Int;MaxIter:Int;Tol:Float;B:TVector;FirstPar,LastPar:Int;V:TMatrix)
7798: Proc WNLFit(RegFunc:TRegFunc;DerivProc:TDerivProc;X,Y,S:TVector;Lb,Ub:Int;MaxIter:Int;Tol
:Float;B:TVector;FirstPar,LastPar:Int;V:TMatrix)
7799: Proc SetMCFile( FileName :Str)
7800: Proc SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Int;B:TVector;FirstPar,LastPar:Int;V:TMatrix;
7801: Proc WSimFit(RegFunc:TRegFunc;X,Y,S:TVector;Lb,Ub:Int;B:TVector;FirstPar,LastPar:Int;V:TMatrix);
7802: end;
7803:
7804: {-----*)
7805: Proc SIRegister_usimplex(CL: TPSPascalCompiler);
7806: begin
7807: Proc SaveSimplex( FileName :Str)
7808: Proc Simplex(Func:TFuncNVar;X:TVector;Lb,Ub:Int; MaxIter:Int;Tol:Float;var F_min:Float);
7809: end;
7810: {-----*)
7811: Proc SIRegister_uregtest(CL: TPSPascalCompiler);
7812: begin
7813: Proc RegTest(Y,Ycalc:TVector;LbY,UbY:Int;V:TMatrix;LbV,UbV:Int;var Test:TRegTest)
7814: Proc WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Int;V:TMatrix;LbV,UbV:Int;var Test:TRegTest);
7815: end;
7816:
7817: Proc SIRegister_ustrings(CL: TPSPascalCompiler);
7818: begin
7819: Func LTrim( S:Str) :Str
7820: Func RTrim( S:Str) :Str

```

```

7821: Func uTrim( S:Str) :Str
7822: Func StrChar(N : Byte; C : Char) :Str
7823: Func RFill( S:Str; L : Byte) :Str
7824: Func LFill( S:Str; L : Byte) :Str
7825: Func CFill( S:Str; L : Byte) :Str
7826: Func Replace(S :Str; C1, C2 : Char) :Str
7827: Func Extract(S :Str; var Index : Byte; Delim : Char) :Str
7828: Proc Parse( S :Str; Delim : Char; Field : TStrVector; var N : Byte)
7829: Proc SetFormat( NumLength, MaxDec : Int; FloatPoint, NSZero :Bool)
7830: Func FloatStr( X : Float) :Str
7831: Func IntStr( N : LongInt) :Str
7832: Func uCompStr( Z : Complex) :Str
7833: end;
7834:
7835: Proc SIRegister_uhyper(CL: TPSPascalCompiler);
7836: begin
7837:   Func uSinh( X : Float) : Float
7838:   Func uCosh( X : Float) : Float
7839:   Func uTanh( X : Float) : Float
7840:   Func uArcSinh( X : Float) : Float
7841:   Func uArcCosh( X : Float) : Float
7842:   Func ArcTanh( X : Float) : Float
7843:   Proc SinhCosh( X : Float; var SinhX, CoshX : Float)
7844:   end;
7845:
7846: Proc SIRegister_urandom(CL: TPSPascalCompiler);
7847: begin
7848:   type RNG_Type =
7849:     (RNG_MWC,      { Multiply-With-Carry }
7850:     RNG_MT,        { Mersenne Twister }
7851:     RNG_UVAG);     { Universal Virtual Array Generator }
7852:   Proc SetRNG( RNG : RNG_Type)
7853:   Proc InitGen( Seed : RNG_IntType)
7854:   Proc SRand( Seed : RNG_IntType)
7855:   Func IRanGen : RNG_IntType
7856:   Func IRanGen31 : RNG_IntType
7857:   Func RanGen1 : Float
7858:   Func RanGen2 : Float
7859:   Func RanGen3 : Float
7860:   Func RanGen53 : Float
7861:   end;
7862:
7863: // Optimization by Simulated Annealing
7864: Proc SIRegister_usimann(CL: TPSPascalCompiler);
7865: begin
7866:   Proc InitSAParams( NT, NS, NCycles : Int; RT : Float)
7867:   Proc SA_CreateLogFile( FileName :Str)
7868:   Proc SimAnn(Func: TFuncNVar; X,Xmin,Xmax: TVector; Lb, Ub : Int; var F_min : Float);
7869:   end;
7870:
7871: Proc SIRegister_uranuvag(CL: TPSPascalCompiler);
7872: begin
7873:   Proc InitUVAGbyString( KeyPhrase :Str)
7874:   Proc InitUVAG( Seed : RNG_IntType)
7875:   Func IRanUVAG : RNG_IntType
7876:   end;
7877:
7878: Proc SIRegister_uenalg(CL: TPSPascalCompiler);
7879: begin
7880:   Proc InitGAParams( NP, NG : Int; SR, MR, HR : Float)
7881:   Proc GA_CreateLogFile( LogFileName :Str)
7882:   Proc GenAlg(Func: TFuncNVar; X,Xmin,Xmax : TVector; Lb, Ub : Int; var F_min : Float);
7883:   end;
7884:
7885:   TVector', 'array of Float
7886: Proc SIRegister_uqsort(CL: TPSPascalCompiler);
7887: begin
7888:   Proc QSort( X : TVector; Lb, Ub : Int)
7889:   Proc DQSort( X : TVector; Lb, Ub : Int)
7890:   end;
7891:
7892: Proc SIRegister_uinterv(CL: TPSPascalCompiler);
7893: begin
7894:   Proc Interval( X1, X2 : Float; MinDiv, MaxDiv : Int; var Min, Max, Step : Float)
7895:   Proc AutoScale(X: TVector; Lb, Ub : Int; Scale : TScale; var XMin, XMax, XStep:Float)
7896:   end;
7897:
7898: Proc SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7899: begin
7900:   FT Result', 'Int
7901:   //TDWordptr', '^DWord // will not work
7902:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWord'
7903:   d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PChar
7904:   r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP
7905:   ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B
7906:   yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By
7907:   te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte
7908:   ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S
7909:   erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'

```

```

7910: Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By
7911: te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B
7912: yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7913: Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I
7914: nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv
7915: ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7916: : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RisVCP : B
7917: yte; end
7918: end;
7919:
7920: //***** PaintFX*****
7921: Proc SRegister_TJvPaintFX(CL: TPSPascalCompiler);
7922: begin
7923: //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7924: with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7925: Proc Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Int)
7926: Proc Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Int)
7927: Proc Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7928: Proc Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount: Single)
7929: Proc ExtractColor( const Dst : TBitmap; AColor : TColor)
7930: Proc ExcludeColor( const Dst : TBitmap; AColor : TColor)
7931: Proc Turn( Src, Dst : TBitmap)
7932: Proc TurnRight( Src, Dst : TBitmap)
7933: Proc HeightMap( const Dst : TBitmap; Amount : Int)
7934: Proc TexturizeTile( const Dst : TBitmap; Amount : Int)
7935: Proc TexturizeOverlap( const Dst : TBitmap; Amount : Int)
7936: Proc RippleRandom( const Dst : TBitmap; Amount : Int)
7937: Proc RippleTooth( const Dst : TBitmap; Amount : Int)
7938: Proc RippleTriangle( const Dst : TBitmap; Amount : Int)
7939: Proc Triangles( const Dst : TBitmap; Amount : Int)
7940: Proc DrawMandelJulia(const Dst:TBitmap;x0,y0,x1,y1:Single;Niter:Int;Mandel:Bool)
7941: Proc FilterXBlue( const Dst : TBitmap; Min, Max : Int)
7942: Proc FilterXGreen( const Dst : TBitmap; Min, Max : Int)
7943: Proc FilterXRed( const Dst : TBitmap; Min, Max : Int)
7944: Proc FilterBlue( const Dst : TBitmap; Min, Max : Int)
7945: Proc FilterGreen( const Dst : TBitmap; Min, Max : Int)
7946: Proc FilterRed( const Dst : TBitmap; Min, Max : Int)
7947: Proc Emboss( var Bmp : TBitmap)
7948: Proc Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7949: Proc Shake( Src, Dst : TBitmap; Factor : Single)
7950: Proc ShakeDown( Src, Dst : TBitmap; Factor : Single)
7951: Proc KeepBlue( const Dst : TBitmap; Factor : Single)
7952: Proc KeepGreen( const Dst : TBitmap; Factor : Single)
7953: Proc KeepRed( const Dst : TBitmap; Factor : Single)
7954: Proc Mandelbrot( const Dst : TBitmap; Factor : Int)
7955: Proc MaskMandelbrot( const Dst : TBitmap; Factor : Int)
7956: Proc FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7957: Proc QuartoOpaque( Src, Dst : TBitmap)
7958: Proc SemiOpaque( Src, Dst : TBitmap)
7959: Proc ShadowDownLeft( const Dst : TBitmap)
7960: Proc ShadowDownRight( const Dst : TBitmap)
7961: Proc ShadowUpLeft( const Dst : TBitmap)
7962: Proc ShadowUpRight( const Dst : TBitmap)
7963: Proc Darkness( const Dst : TBitmap; Amount : Int)
7964: Proc Trace( const Dst : TBitmap; Intensity : Int)
7965: Proc FlipRight( const Dst : TBitmap)
7966: Proc FlipDown( const Dst : TBitmap)
7967: Proc SpotLight( const Dst : TBitmap; Amount : Int; Spot : TRect)
7968: Proc SplitLight( const Dst : TBitmap; Amount : Int)
7969: Proc MakeSeamlessClip( var Dst : TBitmap; Seam : Int)
7970: Proc Wave( const Dst : TBitmap; Amount, Inference, Style : Int)
7971: Proc Mosaic( const Bm : TBitmap; Size : Int)
7972: Proc SmoothRotate( var Src, Dst : TBitmap; CX, CY : Int; Angle : Single)
7973: Proc SmoothResize( var Src, Dst : TBitmap)
7974: Proc Twist( var Bmp, Dst : TBitmap; Amount : Int)
7975: Proc SplitBlur( const Dst : TBitmap; Amount : Int)
7976: Proc GaussianBlur( const Dst : TBitmap; Amount : Int)
7977: Proc Smooth( const Dst : TBitmap; Weight : Int)
7978: Proc GrayScale( const Dst : TBitmap)
7979: Proc AddColorNoise( const Dst : TBitmap; Amount : Int)
7980: Proc AddMonoNoise( const Dst : TBitmap; Amount : Int)
7981: Proc Contrast( const Dst : TBitmap; Amount : Int)
7982: Proc Lightness( const Dst : TBitmap; Amount : Int)
7983: Proc Saturation( const Dst : TBitmap; Amount : Int)
7984: Proc Spray( const Dst : TBitmap; Amount : Int)
7985: Proc AntiAlias( const Dst : TBitmap)
7986: Proc AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Int)
7987: Proc SmoothPoint( const Dst : TBitmap; XK, YK : Int)
7988: Proc FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7989: Proc Marble(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7990: Proc Marble2(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7991: Proc Marble3(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7992: Proc Marble4(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7993: Proc Marble5(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7994: Proc Marble6(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7995: Proc Marble7(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7996: Proc Marble8(const Src:TBitmap; var Dst:TBitmap;Scale:Single;Turbulence:Int)
7997: Proc SqueezeHor( Src, Dst : TBitmap; Amount : Int; Style : TLightBrush)
7998: Proc SplitRound( Src, Dst : TBitmap; Amount : Int; Style : TLightBrush)

```

```

7999: Proc Tile( Src, Dst : TBitmap; Amount : Int)
8000: Proc Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
8001: Proc Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Int)
8002: Proc Invert( Src : TBitmap)
8003: Proc MirrorRight( Src : TBitmap)
8004: Proc MirrorDown( Src : TBitmap)
8005: end;
8006: end;
8007:
8008: (*-----*)
8009: Proc SIRegister_JvPaintFX(CL: TPSPascalCompiler);
8010: begin
8011:   AddTypeS('TLightBrush','( lbBrightness, lbContrast, lbSaturation, lbFishe'
8012:     + 'ye, lbrotate, lbtwist, lbriple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
8013:     + 'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
8014:   SIRegister_TJvPaintFX(CL);
8015:   Func SplineFilter( Value : Single) : Single
8016:   Func BellFilter( Value : Single) : Single
8017:   Func TriangleFilter( Value : Single) : Single
8018:   Func BoxFilter( Value : Single) : Single
8019:   Func HermiteFilter( Value : Single) : Single
8020:   Func Lanczos3Filter( Value : Single) : Single
8021:   Func MitchellFilter( Value : Single) : Single
8022: end;
8023:
8024: (*-----*)
8025: Proc SIRegister_Chart(CL: TPSPascalCompiler);
8026: begin
8027:   'TeeMsg_DefaultFunctionName','String 'TeeFunction
8028:   TeeMsg_DefaultSeriesName','String 'Series
8029:   TeeMsg_DefaultToolName','String 'ChartTool
8030:   ChartComponentPalette','String 'TeeChart
8031:   TeeMaxLegendColumns','LongInt'( 2);
8032:   TeeDefaultLegendSymbolWidth','LongInt'( 20);
8033:   TeeTitleFootDistance,'LongInt'( 5);
8034:   SIRegister_TCustomChartWall(CL);
8035:   SIRegister_TChartWall(CL);
8036:   SIRegister_TChartLegendGradient(CL);
8037:   TLegendStyle','( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
8038:   TLegendAlignment','( laLeft, laRight, laTop, laBottom )
8039:   FindClass('TOBJECT'),'LegendException
8040:   TOnGetLegendText','Proc ( Sender : TCustomAxisPanel; Legen'
8041:     + 'dStyle : TLegendStyle; Index : Int; var LegendText :Str)
8042:   FindClass('TOBJECT'),'TCustomChartLegend
8043:   TLegendSymbolSize','( lcsPercent, lcsPixels )
8044:   TLegendSymbolPosition','( splLeft, splRight )
8045:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Int;R:TRect);
8046:   TSymbolCalcHeight','Func : Int
8047:   SIRegister_TLegendSymbol(CL);
8048:   SIRegister_TTeeCustomShapePosition(CL);
8049:   TCheckBoxesStyle','( cbsCheck, cbsRadio )
8050:   SIRegister_TLegendTitle(CL);
8051:   SIRegister_TLegendItem(CL);
8052:   SIRegister_TLegendItems(CL);
8053:   TLegendCalcSize','Proc ( Sender : TCustomChartLegend; var ASize : Int)
8054:   FindClass('TOBJECT'),'TCustomChart
8055:   SIRegister_TCustomChartLegend(CL);
8056:   SIRegister_TChartLegend(CL);
8057:   SIRegister_TChartTitle(CL);
8058:   SIRegister_TChartFootTitle(CL);
8059:   TChartClick','Procedure(ender:TCustomChart;Button:TMouseButton;Shift:TShiftState;X,Y:Int)
8060:   TChartClickAxis','Proc ( Sender : TCustomChart; Axis : TCh'
8061:     + 'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Int)
8062:   TChartClickSeries','Proc ( Sender : TCustomChart; Series : '
8063:     + 'TChartSeries; ValueIndex : Int; Button : TMouseButton; Shift:TShiftState;X,Y:Int)
8064:   TChartClickTitle','Proc ( Sender : TCustomChart; ATitle : '
8065:     + 'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Int)
8066:   TOnGetLegendPos','Proc (Sender: TCustomChart; Index: Int; var X,Y,XColor:Int)
8067:   TOnGetLegendRect','Proc ( Sender : TCustomChart; var Rect : TRect)
8068:   TAxisSavedScales','record Auto:Boolean;AutoMin:Bool;AutoMax:Bool;Min:Double;Max:Double;end
8069:   TAllAxisSavedScales','array of TAxisSavedScales
8070:   SIRegister_TChartBackWall(CL);
8071:   SIRegister_TChartRightWall(CL);
8072:   SIRegister_TChartBottomWall(CL);
8073:   SIRegister_TChartLeftWall(CL);
8074:   SIRegister_TChartWalls(CL);
8075:   TChartAllowScrollEvent,'Procedure(Sender:TChartAxis;var Amin,AMax:Double;var AllowScroll:Bool);
8076:   SIRegister_TCustomChart(CL);
8077:   SIRegister_TChart(CL);
8078:   SIRegister_TTeeSeriesTypes(CL);
8079:   SIRegister_TTeeToolTypes(CL);
8080:   SIRegister_TTeeDragObject(CL);
8081:   SIRegister_TColorPalettes(CL);
8082:   Proc RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
   AGalleryPage:PString;ANumGallerySeries:Int;
8083:   Proc RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
8084:   Proc RegisterTeeFunction(AFuncClass:TFunctionClass;ADescription,
   AGalleryPage:PString;ANumGallerySeries:Int;
8085:   Proc RegisterTeeBasicFunction(AFunctionClass: TTeeFunctionClass; ADescription : PString)

```

```

8086: Proc RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass; AFunctionClass: TteeFunctionClass;
      ADescription, AGalleryPage: PString; ANumGallerySeries: Int; ASubIndex : Int)
8087: Proc UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
8088: Proc UnRegisterTeeFunctions( const AFunctionList : array of TteeFunctionClass)
8089: Proc AssignSeries( var OldSeries, NewSeries : TChartSeries)
8090: Func CreateNewTeeFunction(ASeries: TChartSeries; AClass: TteeFunctionClass) : TteeFunction
8091: Func CreateNewSeries(AOwner: TComponent; AChar: TCustomAxisPanel; AClass: TChartSeriesClass; AFunctionClass :
      TteeFunctionClass): TChartSeries
8092: Func CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
8093: Func CloneChartSeries1(ASeries: TChartSeries; AChart: TCustomAxisPanel): TChartSeries;
8094: Func CloneChartSeries2(ASeries: TChartSeries; AOwner: TComponent; AChart: TCustomAxisPanel): TChartSeries;;
8095: Func CloneChartTool( ATool : TteeCustomTool; AOwner : TComponent) : TteeCustomTool
8096: Func ChangeSeriesType(var ASeries: TChartSeries; NewType: TChartSeriesClass) : TChartSeries
8097: Proc ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
8098: Func GetNewSeriesName( AOwner : TComponent) : TComponentName
8099: Proc RegisterTeeTools( const ATools : array of TteeCustomToolClass)
8100: Proc UnRegisterTeeTools( const ATools : array of TteeCustomToolClass)
8101: Func GetGallerySeriesName( ASeries : TChartSeries) : Str
8102: Proc PaintSeriesLegend(ASeries: TChartSeries; ACanvas: TCanvas; const R: TRect; ReferenceChart: TCustomChart);
8103: SIRegister_TChartTheme(CL); //TChartThemeClass', 'class of TChartTheme
8104: //TCanvasClass', 'class of TCanvas3D
8105: Func SeriesNameOrIndex( ASeries : TCustomChartSeries) : Str
8106: Func SeriesTitleOrName( ASeries : TCustomChartSeries) : Str
8107: Proc FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Bool)
8108: Proc ShowMessageUser( const S : Str)
8109: Func HasNoMandatoryValues( ASeries : TChartSeries) : Bool
8110: Func HasLabels( ASeries : TChartSeries): Bool
8111: Func HasColors( ASeries : TChartSeries): Bool
8112: Func SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
8113: Proc TeeDrawBitmapEditor(Canvas: TCanvas; Element: TCustomChartElement; Left, Top: Int)
8114: end;
8115:
8116: Proc SIRegister_TeeProcs(CL: TPSPascalCompiler);
8117: begin
8118: //TeeFormBorderStyle', 'bsNone);
8119: SIRegister_TMetafile(CL);
8120: 'TeeDefVerticalMargin', 'LongInt'( 4);
8121: 'TeeDefHorizMargin', 'LongInt'( 3);
8122: 'crTeeHand', 'LongInt'( TCursor ( 2020 ));
8123: 'TeeMsg_TeeHand', 'String 'crTeeHand
8124: 'TeeNormalPrintDetail', 'LongInt'( 0);
8125: 'TeeHighPrintDetail', 'LongInt'( - 100);
8126: 'TeeDefault_PrintMargin', 'LongInt'( 15);
8127: 'MaxDefaultColors', 'LongInt'( 19);
8128: 'TeeTabDelimiter', Char #9;
8129: TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
8130: + 'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
8131: + 'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
8132: + 'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
8133: + 'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
8134: + 'ourMonths, dtSixMonths, dtOneYear, dtNone )
8135: SIRegister_TCustomPanelNoCaption(CL);
8136: FindClass('TObject'), 'TCustomTeePanel
8137: SIRegister_TZoomPanning(CL);
8138: SIRegister_TTeeEvent(CL);
8139: //SIRegister_TTeeEventListeners(CL);
8140: TTeeMouseEventKind', '( meDown, meUp, meMove )
8141: SIRegister_TTeeMouseEvent(CL);
8142: SIRegister_TCustomTeePanel(CL);
8143: //TChartGradient', 'TteeGradient
8144: //TChartGradientClass', 'class of TChartGradient
8145: TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
8146: SIRegister_TTeeZoomPen(CL);
8147: SIRegister_TTeeZoomBrush(CL);
8148: TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
8149: SIRegister_TTeeZoom(CL);
8150: FindClass('TObject'), 'TCustomTeePanelExtended
8151: TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
8152: SIRegister_TBackImage(CL);
8153: SIRegister_TCustomTeePanelExtended(CL);
8154: //TChartBrushClass', 'class of TChartBrush
8155: SIRegister_TTeeCustomShapeBrushPen(CL);
8156: TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
8157: TTextFormat', '( ttfNormal, ttfHtml )
8158: SIRegister_TTeeCustomShape(CL);
8159: SIRegister_TTeeShape(CL);
8160: SIRegister_TTeeExportData(CL);
8161: Func TeeStr( const Num : Int) : Str
8162: Func DateTimeDefaultFormat( const AStep : Double) : Str
8163: Func TEEDaysInMonth( Year, Month : Word) : Word
8164: Func FindDateTimeStep( const StepValue : Double) : TDateTimeStep
8165: Func NextDateTimeStep( const AStep : Double) : Double
8166: Func PointInLine( const P : TPoint; const px, py, qx, qy : Int): Bool;
8167: Func PointInLine1( const P, FromPoint, ToPoint : TPoint) : Bool;
8168: Func PointInLine2( const P, FromPoint, ToPoint: TPoint; const TolerancePixels: Int): Bool;
8169: Func PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels: Int): Bool;
8170: Func PointInLineTolerance( const P: TPoint; const px, py, qx, qy, TolerancePixels: Int): Bool;
8171: Func PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Bool
8172: Func PointInTriangle( const P, P0, P1, P2 : TPoint) : Bool;

```



```

8173: Func PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Int) :Bool;
8174: Func PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Int) :Bool
8175: Func PointInEllipse( const P : TPoint; const Rect : TRect) :Bool;
8176: Func PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Int) :Bool;
8177: Func DelphiToLocalFormat( const Format :Str) :Str
8178: Func LocalToDelphiFormat( const Format :Str) :Str
8179: Proc TEENableControls(Enable:Bool; const ControlArray : array of TControl)
8180: Func TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8181: Proc TeeDateTimeIncrement(IsDateTime:Bool;Increment:Bool;var Value:Double;const AnIncrement:Double;
tmpWhichDateTime:TDateTimeStep)
8182: TTeeSortCompare', 'Func ( a, b : Int) : Int
8183: TTeeSortSwap', 'Proc ( a, b : Int)
8184: Proc TeeSort(StartIndex,EndIndex:Int;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8185: Func TeeGetUniqueName( AOwner : TComponent; const AStartName :Str) :Str
8186: Func TeeExtractField( St :Str; Index : Int) :Str;
8187: Func TeeExtractField1( St :Str; Index : Int; const Separator :Str) :Str;
8188: Func TeeNumFields( St :Str) : Int;
8189: Func TeeNumFields1( const St, Separator :Str) : Int;
8190: Proc TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8191: Proc TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 :Str)
8192: // TColorArray', 'array of TColor
8193: Func GetDefaultColor( const Index : Int) : TColor
8194: Proc SetDefaultColorPalette;
8195: Proc SetDefaultColorPalett1( const Palette : array of TColor);
8196: 'TeeCheckBoxSize','LongInt'( 11);
8197: Proc TeeDrawCheckBox(x,y:Int;Canvas:TCanvas;Checked:Bool;ABackColor:TColor;CheckBox:Bool);
8198: Func TEEStrToFloatDef( const S :Str; const Default : Extended) : Extended
8199: Func TryStrToFloat( const S :Str; var Value : Double) :Bool
8200: Func CrossingLines(const X1,Y1,X2,Y2,X3,Y3,X4,Y4: Double; out x, y : Double) :Bool
8201: Proc TeeTranslateControl( AControl : TControl);
8202: Proc TeeTranslateControl1(AControl: TControl;const ExcludeChilds: array of TControl);
8203: Func ReplaceChar(const AString:str;const Search: Char; const Replace : Char) :Str
8204: //Proc RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8205: Func TeeAntiAlias( Panel : TCustomTeePanel; ChartRect :Bool) : TBitmap
8206: //Proc DrawBevel(Canvas:TeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Int;Round:Int);
8207: //Func ScreenRatio( ACanvas : TCanvas3D) : Double
8208: Func TeeReadBoolOption( const AKey :Str; DefaultValue :Bool) :Bool
8209: Proc TeeSaveBoolOption( const AKey :Str; Value :Bool)
8210: Func TeeReadIntOption( const AKey :Str; DefaultValue : Int) : Int
8211: Proc TeeSaveIntOption( const AKey :Str; Value : Int)
8212: Func TeeReadStringOption( const AKey, DefaultValue :Str) :Str
8213: Proc TeeSaveStringOption( const AKey, Value :Str)
8214: Func TeeDefaultXMLEncoding :Str
8215: Proc ConvertTextToXML(Stream: TStream; XMLHeader :Bool)
8216: TeeWindowHandle', 'Int
8217: Proc TeeGotoURL( Handle : TeeWindowHandle; const URL :Str)
8218: Proc HtmlTextOut( ACanvas : TCanvas; x, y : Int; Text :Str)
8219: Func HtmlTextExtent( ACanvas : TCanvas; const Text :Str) : TSize
8220: end;
8221:
8222: fix and add teEngine: 4.7.6.20
8223: TChartAxisTitle=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8224: TChartSeriesList=class {$IFDEF CLR}sealed{$ENDIF} (TCustomSeriesList)
8225: TSeriesGroup=class {$IFDEF CLR}sealed{$ENDIF} (TCollectionItem)
8226: Destructor Destroy; override;
8227: TSeriesGroups=class {$IFDEF CLR}sealed{$ENDIF} (TOwnedCollection)
8228: TChartCustomAxes=class {$IFDEF CLR}sealed{$ENDIF} (TOwnedCollection)
8229: TChartChangePage=class {$IFDEF CLR}sealed{$ENDIF} (TTeeEvent);
8230: TSeriesMarksPositions=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8231: TMarksItem=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8232: property Bevel;
8233: property BevelWidth;
8234: property Color default ChartMarkColor;
8235: property Font;
8236: property Gradient;
8237: property Picture;
8238: property Shadow;
8239: property ShapeStyle;
8240: property Text:TStrings read GetText write SetText;
8241: property Transparency;
8242: property Transparent;
8243: TMarksItems=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8244: TSeriesMarksGradient=class {$IFDEF CLR}sealed{$ENDIF} (TChartGradient)
8245: property Direction default gdRightLeft;
8246: property EndColor default clWhite;
8247: property StartColor default clSilver;
8248: TSeriesMarksSymbol=class {$IFDEF CLR}sealed{$ENDIF} (TTeeCustomShape)
8249: TChartValueLists=class {$IFDEF CLR}sealed{$ENDIF} (TList)
8250: Procedure Clear; override;
8251: TEEProcs
8252: TTeeMouseEvent=class {$IFDEF CLR}sealed{$ENDIF} (TTeeEvent)
8253: with CL.AddClassN(CL.FindClass('TOBJECT'),'TTeeMouseEvent') do begin
8254: CNN CreateAscentImages - unit neuraldatasetsv;
8255:
8256:
8257: using mXBDEUtils
8258: *****
8259: Proc SetAlias( aAlias, aDirectory :str)
8260: Proc CheckRegistryEntry(Reg:TRegistry;const Path,Value:Str;const Default,Desired:Variant;Size:Byte);

```

```

8261: Func GetFileVersionNumber( const FileName :Str) : TVersionNo
8262: Proc SetBDE( aPath, aNode, aValue :Str)
8263: Func RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Int): Int; stdcall;
8264: Func GetSystemDirectory( lpBuffer :Str; uSize : UINT) : UINT
8265: Func GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8266: Func GetTempPath( nBufferLength : DWORD; lpBuffer :Str) : DWORD
8267: Func GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer :Str) : DWORD
8268: Func GetTempFileName(lpPathName,lpPrefixString:str;uUnique:UINT;lpTempFileName:str):UINT;
8269:
8270: Proc SIRegister_cDateTime(CL: TPSPascalCompiler);
8271: begin
8272: AddClassN(FindClass('TOBJECT'),'EDateTime
8273: Func DatePart( const D : TDateTime) : Int
8274: Func TimePart( const D : TDateTime) : Double
8275: Func Century( const D : TDateTime) : Word
8276: Func Year( const D : TDateTime) : Word
8277: Func Month( const D : TDateTime) : Word
8278: Func Day( const D : TDateTime) : Word
8279: Func Hour( const D : TDateTime) : Word
8280: Func Minute( const D : TDateTime) : Word
8281: Func Second( const D : TDateTime) : Word
8282: Func Millisecond( const D : TDateTime) : Word
8283: ('OneDay','Extended').setExtended( 1.0);
8284: ('OneHour','Extended').SetExtended( OneDay / 24);
8285: ('OneMinute','Extended').SetExtended( OneHour / 60);
8286: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8287: ('OneMillisecond','Extended').SetExtended( OneSecond/ 1000);
8288: ('OneWeek','Extended').SetExtended( OneDay * 7);
8289: ('HoursPerDay','Extended').SetExtended( 24);
8290: ('MinutesPerHour','Extended').SetExtended( 60);
8291: ('SecondsPerMinute','Extended').SetExtended( 60);
8292: Proc SetYear( var D : TDateTime; const Year : Word)
8293: Proc SetMonth( var D : TDateTime; const Month : Word)
8294: Proc SetDay( var D : TDateTime; const Day : Word)
8295: Proc SetHour( var D : TDateTime; const Hour : Word)
8296: Proc SetMinute( var D : TDateTime; const Minute : Word)
8297: Proc SetSecond( var D : TDateTime; const Second : Word)
8298: Proc SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8299: Func IsEqual( const D1, D2 : TDateTime) :Bool;
8300: Func IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8301: Func IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8302: Func IsAM( const D : TDateTime) :Bool
8303: Func IsPM( const D : TDateTime) :Bool
8304: Func IsMidnight( const D : TDateTime) :Bool
8305: Func IsNoon( const D : TDateTime) :Bool
8306: Func IsSunday( const D : TDateTime) :Bool
8307: Func IsMonday( const D : TDateTime) :Bool
8308: Func IsTuesday( const D : TDateTime) :Bool
8309: Func IsWednesday( const D : TDateTime) :Bool
8310: Func IsThursday( const D : TDateTime) :Bool
8311: Func IsFriday( const D : TDateTime) :Bool
8312: Func IsSaturday( const D : TDateTime) :Bool
8313: Func IsWeekend( const D : TDateTime) :Bool
8314: Func Noon( const D : TDateTime) : TDateTime
8315: Func Midnight( const D : TDateTime) : TDateTime
8316: Func FirstDayOfMonth( const D : TDateTime) : TDateTime
8317: Func LastDayOfMonth( const D : TDateTime) : TDateTime
8318: Func NextWorkday( const D : TDateTime) : TDateTime
8319: Func PreviousWorkday( const D : TDateTime) : TDateTime
8320: Func FirstDayOfYear( const D : TDateTime) : TDateTime
8321: Func LastDayOfYear( const D : TDateTime) : TDateTime
8322: Func EasterSunday( const Year : Word) : TDateTime
8323: Func GoodFriday( const Year : Word) : TDateTime
8324: Func AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8325: Func AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8326: Func AddMinutes( const D : TDateTime; const N : Int) : TDateTime
8327: Func AddHours( const D : TDateTime; const N : Int) : TDateTime
8328: Func AddDays( const D : TDateTime; const N : Int) : TDateTime
8329: Func AddWeeks( const D : TDateTime; const N : Int) : TDateTime
8330: Func AddMonths( const D : TDateTime; const N : Int) : TDateTime
8331: Func AddYears( const D : TDateTime; const N : Int) : TDateTime
8332: Func DayOfYear( const Ye, Mo, Da : Word) : Int
8333: Func DayOfYear( const D : TDateTime) : Int
8334: Func DaysInMonth( const Ye, Mo : Word) : Int
8335: Func DaysInMonth( const D : TDateTime) : Int
8336: Func DaysInYear( const Ye : Word) : Int
8337: Func DaysInYearDate( const D : TDateTime) : Int
8338: Func WeekNumber( const D : TDateTime) : Int
8339: Func ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8340: Proc ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8341: Func DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8342: Func DiffSeconds( const D1, D2 : TDateTime) : Int
8343: Func DiffMinutes( const D1, D2 : TDateTime) : Int
8344: Func DiffHours( const D1, D2 : TDateTime) : Int
8345: Func DiffDays( const D1, D2 : TDateTime) : Int
8346: Func DiffWeeks( const D1, D2 : TDateTime) : Int
8347: Func DiffMonths( const D1, D2 : TDateTime) : Int
8348: Func DiffYears( const D1, D2 : TDateTime) : Int
8349: Func GMTBias : Int

```

```

8350: Func GMTTimeToLocalTime( const D : TDateTime ) : TDateTime
8351: Func LocalTimeToGMTTime( const D : TDateTime ) : TDateTime
8352: Func NowAsGMTTime : TDateTime
8353: Func DateTimeToISO8601String( const D : TDateTime ) : Ansistr
8354: Func ISO8601StringToTime( const D : Ansistr ) : TDateTime
8355: Func ISO8601StringAsDateTime( const D : Ansistr ) : TDateTime
8356: Func DateTimeToANSI( const D : TDateTime ) : Int
8357: Func ANSIToDateTime( const Julian : Int ) : TDateTime
8358: Func DateTimeToISOInt( const D : TDateTime ) : Int
8359: Func DateTimeToISOString( const D : TDateTime ) : Ansistr
8360: Func ISOIntToDateTime( const ISOInt : Int ) : TDateTime
8361: Func TwoDigitRadix2000YearToYear( const Y : Int ) : Int
8362: Func DateTimeAsElapsedTime( const D: TDateTime; const IncludeMilliseconds: Boolean ) : Ansistr
8363: Func UnixTimeToDateTime( const UnixTime : LongWord ) : TDateTime
8364: Func DateTimeToUnixTime( const D : TDateTime ) : LongWord
8365: Func EnglishShortDayOfWeekStrA( const DayOfWeek : Int ) : Ansistr
8366: Func EnglishShortDayOfWeekStrU( const DayOfWeek : Int ) : UnicodeString
8367: Func EnglishLongDayOfWeekStrA( const DayOfWeek : Int ) : Ansistr
8368: Func EnglishLongDayOfWeekStrU( const DayOfWeek : Int ) : UnicodeString
8369: Func EnglishShortMonthStrA( const Month : Int ) : Ansistr
8370: Func EnglishShortMonthStrU( const Month : Int ) : UnicodeString
8371: Func EnglishLongMonthStrA( const Month : Int ) : Ansistr
8372: Func EnglishLongMonthStrU( const Month : Int ) : UnicodeString
8373: Func EnglishShortDayOfWeekA( const S : Ansistr ) : Int
8374: Func EnglishShortDayOfWeekU( const S : UnicodeString ) : Int
8375: Func EnglishLongDayOfWeekA( const S : Ansistr ) : Int
8376: Func EnglishLongDayOfWeekU( const S : UnicodeString ) : Int
8377: Func EnglishShortMonthA( const S : Ansistr ) : Int
8378: Func EnglishShortMonthU( const S : UnicodeString ) : Int
8379: Func EnglishLongMonthA( const S : Ansistr ) : Int
8380: Func EnglishLongMonthU( const S : UnicodeString ) : Int
8381: Func RFC850DayOfWeekA( const S : Ansistr ) : Int
8382: Func RFC850DayOfWeekU( const S : UnicodeString ) : Int
8383: Func RFC1123DayOfWeekA( const S : Ansistr ) : Int
8384: Func RFC1123DayOfWeekU( const S : UnicodeString ) : Int
8385: Func RFCMonthA( const S : Ansistr ) : Word
8386: Func RFCMonthU( const S : UnicodeString ) : Word
8387: Func GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds: Boolean ) : Ansistr
8388: Func GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds: Boolean ) : UnicodeString
8389: Func GMTDateTimeToRFC1123DateTimeA( const D: TDateTime; const IncludeDayOfWeek: Bool ) : Ansistr;
8390: Func GMTDateTimeToRFC1123DateTimeU( const D: TDateTime; const IncludeDayOfWeek: Bool ) : UnicodStr;
8391: Func DateTimeToRFCDateTimeA( const D : TDateTime ) : Ansistr
8392: Func DateTimeToRFCDateTimeU( const D : TDateTime ) : UnicodeString
8393: Func NowAsRFCDateTimeA : Ansistr
8394: Func NowAsRFCDateTimeU : UnicodeString
8395: Func RFCDateTimeToGMTDateTime( const S : Ansistr ) : TDateTime
8396: Func RFCDateTimeToDateTime( const S : Ansistr ) : TDateTime
8397: Func RFTTimeZoneToGMTBias( const Zone : Ansistr ) : Int
8398: Func TimePeriodStr( const D : TDateTime ) : Ansistr
8399: Proc SelfTestCDate
8400: end;
8401: //*****CFileUtils
8402: Func PathHasDriveLetterA( const Path : Ansistr ) : Bool
8403: Func PathHasDriveLetter( const Path : Str ) : Bool
8404: Func PathIsDriveLetterA( const Path : Ansistr ) : Bool
8405: Func PathIsDriveLetter( const Path : Str ) : Bool
8406: Func PathIsDriveRootA( const Path : Ansistr ) : Bool
8407: Func PathIsDriveRoot( const Path : Str ) : Bool
8408: Func PathIsRootA( const Path : Ansistr ) : Bool
8409: Func PathIsRoot( const Path : Str ) : Bool
8410: Func PathIsUNCPathA( const Path : Ansistr ) : Bool
8411: Func PathIsUNCPath( const Path : Str ) : Bool
8412: Func PathIsAbsoluteA( const Path : Ansistr ) : Bool
8413: Func PathIsAbsolute( const Path : Str ) : Bool
8414: Func PathIsDirectoryA( const Path : Ansistr ) : Bool
8415: Func PathIsDirectory( const Path : Str ) : Bool
8416: Func PathInclSuffixA( const Path : Ansistr; const PathSep : Char ) : Ansistr
8417: Func PathInclSuffix( const Path : Str; const PathSep : Char ) : Str
8418: Func PathExclSuffixA( const Path : Ansistr; const PathSep : Char ) : Ansistr
8419: Func PathExclSuffix( const Path : Str; const PathSep : Char ) : Str
8420: Proc PathEnsureSuffixA( var Path : Ansistr; const PathSep : Char )
8421: Proc PathEnsureSuffix( var Path : Str; const PathSep : Char )
8422: Proc PathEnsureNoSuffixA( var Path : Ansistr; const PathSep : Char )
8423: Proc PathEnsureNoSuffix( var Path : Str; const PathSep : Char )
8424: //Func PathCanonicalA( const Path : Ansistr; const PathSep : Char ) : Ansistr
8425: Func PathCanonical( const Path : Str; const PathSep : Char ) : Str
8426: Func PathExpandA( const Path: Ansistr; const BasePath: AnsiStr; const PathSep: Char ) : AnsiStr
8427: Func PathExpand( const Path: str; const BasePath : Str; const PathSep : Char ) : Str
8428: Func PathLeftElementA( const Path : Ansistr; const PathSep : Char ) : Ansistr
8429: Func PathLeftElement( const Path : Str; const PathSep : Char ) : Str
8430: Proc PathSplitLeftElementA( const Path: AString; var LeftElement, RightPath: AString; const PathSep: Char );
8431: Proc PathSplitLeftElement( const Path: str; var LeftElement, RightPath: Str; const PathSep: Char );
8432: Proc DecodeFilePathA( const FilePath: AnsiStr; var Path, FileName: AnsiStr; const PathSep: Char;
8433: Proc DecodeFilePath( const FilePath: str; var Path, FileName: str; const PathSep: Char )
8434: Func FileNameValidA( const FileName : Ansistr ) : Ansistr
8435: Func FileNameValid( const FileName : Str ) : Str
8436: Func FilePathA( const FileName, Path: Ansistr; const BasePath: AnsiStr; const PathSep: Char ) : Ansistr;
8437: Func FilePath( const FileName, Path: str; const BasePath: Str; const PathSep: Char ) : Str
8438: Func DirectoryExpandA( const Path: Ansistr; const BasePath: Ansistr; const PathSep: Char ) : Ansistr

```

```

8439: Func DirectoryExpand(const Path:str;const BasePath:str;const PathSep:Char):str
8440: Func UnixPathToWinPath(const Path : Ansistr) : Ansistr
8441: Func WinPathToUnixPath(const Path : Ansistr) : Ansistr
8442: Proc CCopyFile(const FileName, DestName :Str)
8443: Proc CMoveFile(const FileName, DestName :Str)
8444: Func CDeleteFiles(const FileMask :Str) :Bool
8445: Func FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64;const FilePos:TFileSeekPos):Int64;
8446: Proc FileCloseEx(const FileHandle : TFileHandle)
8447: Func FileExistsA(const FileName : Ansistr) :Bool
8448: Func CFileExists(const FileName :Str) :Bool
8449: Func CFileGetSize(const FileName :Str) : Int64
8450: Func FileGetDateTime(const FileName :Str) : TDateTime
8451: Func FileGetDateTime2(const FileName :Str) : TDateTime
8452: Func FileIsReadOnly(const FileName :Str) :Bool
8453: Proc FileDeleteEx(const FileName :Str)
8454: Proc FileRenameEx(const OldFileName, NewFileName :Str)
8455: Func ReadFileStrA(const FileName:Ansistr; const FileSharing : TFileSharing; const FileCreationMode :
TFileCreationMode; const FileOpenWait : PFileOpenWait) : Ansistr
8456: Func DirectoryEntryExists(const Name :Str) :Bool
8457: Func DirectoryEntrySize(const Name :Str) : Int64
8458: Func CDirectoryExists(const DirectoryName :Str) :Bool
8459: Func DirectoryGetDateTime(const DirectoryName :Str) : TDateTime
8460: Proc CDirectoryCreate(const DirectoryName :Str)
8461: Func GetFirstFileNameMatching(const FileMask :Str) :Str
8462: Func DirEntryGetAttr(const FileName : Ansistr) : Int
8463: Func DirEntryIsDirectory(const FileName : Ansistr) :Bool
8464: Func FileHasAttr(const FileName :Str; const Attr : Word) :Bool
8465: AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8466: + 'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8467: Func DriveIsValid(const Drive : Char) :Bool
8468: Func DriveGetType(const Path : Ansistr) : TLogicalDriveType
8469: Func DriveFreeSpace(const Path : Ansistr) : Int64
8470:
8471: Proc SIRegister_cTimers(CL: TPSPascalCompiler);
8472: begin
8473: AddClassN(FindClass('TOBJECT'),'ETimers
8474: Const('TickFrequency','LongInt'( 1000);Func GetTick : LongWord
8475: Func TickDelta(const D1, D2 : LongWord) : Int
8476: Func TickDeltaW(const D1, D2 : LongWord) : LongWord
8477: AddTypeS('THPTimer', 'Int64
8478: Proc StartTimer(var Timer : THPTimer)
8479: Proc StopTimer(var Timer : THPTimer)
8480: Proc ResumeTimer(var StoppedTimer : THPTimer)
8481: Proc InitStoppedTimer(var Timer : THPTimer)
8482: Proc InitElapsedTimer(var Timer : THPTimer; const Milliseconds : Int)
8483: Func MillisecondsElapsed(const Timer: THPTimer; const TimerRunning :Bool) : Int
8484: Func MicrosecondsElapsed(const Timer: THPTimer; const TimerRunning :Bool) : Int64
8485: Proc WaitMicroseconds(const MicroSeconds : Int)
8486: Func GetHighPrecisionFrequency : Int64
8487: Func GetHighPrecisionTimerOverhead : Int64
8488: Proc AdjustTimerForOverhead(var StoppedTimer:THPTimer; const Overhead:Int64)
8489: Proc SelfTestCTimer
8490: end;
8491:
8492: Proc SIRegister_cRandom(CL: TPSPascalCompiler);
8493: begin
8494: Func RandomSeed : LongWord
8495: Proc AddEntropy(const Value : LongWord)
8496: Func RandomUniform : LongWord;
8497: Func RandomUniform1(const N : Int) : Int;
8498: Func RandomBoolean :Bool
8499: Func RandomByte : Byte
8500: Func RandomByteNonZero : Byte
8501: Func RandomWord : Word
8502: Func RandomInt64 : Int64;
8503: Func RandomInt641(const N : Int64) : Int64;
8504: Func RandomHex(const Digits : Int) :Str
8505: Func RandomFloat : Extended
8506: Func RandomAlphaStr(const Length : Int) : Ansistr
8507: Func RandomPseudoWord(const Length : Int) : Ansistr
8508: Func RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):Ansistr;
8509: Func mwcRandomLongWord : LongWord
8510: Func urnRandomLongWord : LongWord
8511: Func moaRandomFloat : Extended
8512: Func mwcRandomFloat : Extended
8513: Func RandomNormalF : Extended
8514: Proc SelfTestCRandom
8515: end;
8516:
8517: Proc SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8518: begin
8519: // PIntArray, '^TIntArray // will not work
8520: Addtypes('TConvertTabsProc','function(const Line:Ansistr; TabWidth: Int):Ansistr
8521: TConvertTabsProcEx,function(const Line:Ansistr;TabWidth:Int;var HasTabs:boolean):Ansistr
8522: Func synMax(x, y : Int) : Int
8523: Func synMin(x, y : Int) : Int
8524: Func synMinMax(x, mi, ma : Int) : Int
8525: Proc synSwapInt(var l, r : Int)
8526: Func synMaxPoint(const P1, P2 : TPoint) : TPoint

```

```

8527: Func synMinPoint( const P1, P2 : TPoint ) : TPoint
8528: //Func synGetIntArray( Count :Card; InitialValue : Int ) : PIntArray
8529: Proc synInternalFillRect( dc : HDC; const rcPaint : TRect)
8530: Func synGetBestConvertTabsProc( TabWidth : Int ) : TConvertTabsProc
8531: Func synConvertTabs( const Line : Ansistr; TabWidth : Int ) : Ansistr
8532: Func synGetBestConvertTabsProcEx( TabWidth : Int ) : TConvertTabsProcEx
8533: Func synConvertTabsEx(const Line:Ansistr;TabWidth:Int;var HasTabs:bool):Ansistr;
8534: Func synGetExpandedLength( const aStr :Str; aTabWidth : Int ) : Int
8535: Func synCharIndex2CaretPos( Index, TabWidth : Int; const Line :Str ) : Int
8536: Func synCaretPos2CharIndex(Positin,TabWidth:int,const Line:str;var InsideTabChar:bool):int;
8537: Func synStrScanForCharInSet(const Line:str;Start:Int;AChars:TSynIdentChars):int;
8538: Func synStrRScanForCharInSet(const Line:str;Start:Int;AChars:TSynIdentChars):Int;
8539: TStringType', (stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8540: + 'akana, stWideNumAlpha, stWideSymbol, stWideKatakana, stHiragana, stIdeograph, stControl, stKashida)
8541: ('C3_NONSPACING', 'LongInt'( 1);
8542: 'C3_DIACRITIC', 'LongInt'( 2);
8543: 'C3_VOWELMARK', 'LongInt'( 4);
8544: ('C3_SYMBOL', 'LongInt'( 8);
8545: ('C3_KATAKANA', LongWord( $0010);
8546: ('C3_HIRAGANA', LongWord( $0020);
8547: ('C3_HALFWIDTH', LongWord( $0040);
8548: ('C3_FULLWIDTH', LongWord( $0080);
8549: ('C3_IDEOGRAPH', LongWord( $0100);
8550: ('C3_KASHIDA', LongWord( $0200);
8551: ('C3_LEXICAL', LongWord( $0400);
8552: ('C3_ALPHA', LongWord( $8000);
8553: ('C3_NOTAPPLICABLE', 'LongInt'( 0);
8554: Func synStrScanForMultiByteChar( const Line :Str; Start : Int ) : Int
8555: Func synStrRScanForMultiByteChar( const Line :Str; Start : Int ) : Int
8556: Func synIsStringType( Value : Word ) : TStringType
8557: Func synGetEOL( Line : PChar ) : PChar
8558: Func synEncodeString( s :Str ) :Str
8559: Func synDecodeString( s :Str ) :Str
8560: Proc synFreeAndNil( var Obj: TObject)
8561: Proc synAssert( Expr :Bool)
8562: Func synLastDelimiter( const Delimiters, S :Str ) : Int
8563: TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8564: TReplaceFlags', 'set of TReplaceFlag )
8565: Func synStringReplace(const S, OldPattern, NewPattern:str;Flags:TReplaceFlags):str
8566: Func synGetRValue( RGBValue : TColor ) : byte
8567: Func synGetGValue( RGBValue : TColor ) : byte
8568: Func synGetBValue( RGBValue : TColor ) : byte
8569: Func synRGB( r, g, b : Byte ) :Card
8570: //THighlighterAttriProc', 'Func ( Highlighter : TSynCustomHigh'
8571: //+ 'lighter;Attri:TSynHighlighterAttributes;UniqueAttriName:str;Params array of Pointer):Bool;
8572: //Func synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates :Bool;
HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) :Bool
8573: Func synCalcFCS( const ABuf, ABufSize :Card ) : Word
8574: Proc synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,AEndColor:TColor;ASteps:Int;const
ARect:TRect; const AHorizontal:Bool) end;
8575: Func GET_APPCOMMAND_LPARAM( lParam : LPARAM ) : WORD
8576: Func GET_DEVICE_LPARAM( lParam : LPARAM ) : WORD
8577: Func GET_KEYSTATE_LPARAM( lParam : LPARAM ) : WORD
8578:
8579: Proc SIRegister_synutil(CL: TPSPascalCompiler);
8580: begin
8581: Func STimeZoneBias : Int
8582: Func TimeZone :Str
8583: Func Rfc822DateTime( t : TDateTime ) :Str
8584: Func CDateTime( t : TDateTime ) :Str
8585: Func SimpleDateTime( t : TDateTime ) :Str
8586: Func AnsiCDateTime( t : TDateTime ) :Str
8587: Func GetMonthNumber( Value :Str ) : Int
8588: Func GetTimeFromStr( Value :Str) : TDateTime
8589: Func GetDateMDYFromStr( Value :Str) : TDateTime
8590: Func DecodeRfcDateTime( Value :Str ) : TDateTime
8591: Func GetUTTime : TDateTime
8592: Func SetUTTime( Newdt : TDateTime ) :Bool
8593: Func SGetTick : LongWord
8594: Func STickDelta( TickOld, TickNew : LongWord ) : LongWord
8595: Func CodeInt( Value : Word ) : Ansistr
8596: Func DecodeInt( const Value : Ansistr; Index : Int ) : Word
8597: Func CodeLongInt( Value : LongInt ) : Ansistr
8598: Func DecodeLongInt( const Value : Ansistr; Index : Int) : LongInt
8599: Func DumpStr( const Buffer : Ansistr ) :Str
8600: Func DumpExStr( const Buffer : Ansistr ) :Str
8601: Proc Dump( const Buffer : Ansistr; DumpFile :Str)
8602: Proc DumpEx( const Buffer : Ansistr; DumpFile :Str)
8603: Func TrimSPLeft( const S :Str ) :Str
8604: Func TrimSPRight( const S :Str ) :Str
8605: Func TrimSP( const S :Str ) :Str
8606: Func SeparateLeft( const Value, Delimiter :Str ) :Str
8607: Func SeparateRight( const Value, Delimiter :Str ) :Str
8608: Func SGetParameter( const Value, Parameter :Str ) :Str
8609: Proc ParseParametersEx(Value,Delimiter:str;const Parameters : TString)
8610: Proc ParseParameters( Value :Str; const Parameters : TString)
8611: Func IndexByBegin( Value :Str; const List : TString) : Int
8612: Func GetEmailAddr( const Value :Str ) :Str
8613: Func GetEmailDesc( Value :Str ) :Str

```



```

8614: Func CStrToHex( const Value : Ansistr ) :Str
8615: Func CIntToBin( Value : Int; Digits : Byte) :Str
8616: Func CBinToInt( const Value :Str) : Int
8617: Func ParseURL(URL:Str;var Prot,User,Pass,Host,Port,Path,Para:str):str
8618: Func CReplaceString( Value, Search, Replace : Ansistr) : Ansistr
8619: Func CRPosEx( const Sub, Value :Str; From : Int) : Int
8620: Func CRPos( const Sub, Value :Str) : Int
8621: Func FetchBin( var Value :Str; const Delimiter :Str) :Str
8622: Func CFetch( var Value :Str; const Delimiter :Str) :Str
8623: Func FetchEx(var Value:Str;const Delimiter,Quotation:str):Str
8624: Func IsBinaryString( const Value : Ansistr) :Bool
8625: Func PosCRLF( const Value : Ansistr; var Terminator : Ansistr) : Int
8626: Proc StringsTrim( const value : TStrings)
8627: Func PosFrom( const SubStr, Value :Str; From : Int) : Int
8628: Func IncPoint( const p : __pointer; Value : Int) : __pointer
8629: Func GetBetween( const PairBegin, PairEnd, Value :Str) :Str
8630: Func CCountOfChar( const Value :Str; aChr : char) : Int
8631: Func UnquoteStr( const Value :Str; Quote : Char) :Str
8632: Func QuoteStr( const Value :Str; Quote : Char) :Str
8633: Proc HeadersToList( const Value : TStrings)
8634: Proc ListToHeaders( const Value : TStrings)
8635: Func SwapBytes( Value : Int) : Int
8636: Func ReadStrFromStream( const Stream : TStream; len : Int) : Ansistr
8637: Proc WriteStrToStream( const Stream : TStream; Value : Ansistr)
8638: Func GetTempFile( const Dir, prefix : Ansistr) : Ansistr
8639: Func CPadString( const Value:Ansistr;len:Int;Pad:AnsiChar): Ansistr
8640: Func CXorString( Indatal, Indata2 : Ansistr) : Ansistr
8641: Func NormalizeHeader( Value : TStrings; var Index : Int) :Str
8642: end;
8643:
8644: Proc SIRegister_StCRC(CL: TPSPascalCompiler);
8645: begin
8646: ('CrcBufSize','LongInt'( 2048);
8647: Func Adler32Prim( var Data, DataSize :Card; CurCrc : LongInt): LongInt
8648: Func Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8649: Func Adler32OfFile( FileName : Ansistr) : LongInt
8650: Func Crc16Prim( var Data, DataSize, CurCrc :Card) :Card
8651: Func Crc16OfStream( Stream : TStream; CurCrc :Card) :Card
8652: Func Crc16OfFile( FileName : Ansistr) :Card
8653: Func Crc32Prim( var Data, DataSize :Card; CurCrc : LongInt) : LongInt
8654: Func Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8655: Func Crc32OfFile( FileName : Ansistr) : LongInt
8656: Func InternetSumPrim( var Data, DataSize, CurCrc :Card) :Card
8657: Func InternetSumOfStream( Stream : TStream; CurCrc :Card) :Card
8658: Func InternetSumOfFile( FileName : Ansistr) :Card
8659: Func Kermit16Prim( var Data, DataSize, CurCrc :Card) :Card
8660: Func Kermit16OfStream( Stream : TStream; CurCrc :Card) :Card
8661: Func Kermit16OfFile( FileName : Ansistr) :Card
8662: end;
8663:
8664: Proc SIRegister_ComObj(cl: TPSPascalCompiler);
8665: begin
8666: Func CreateOleObject(const ClassName:Str): IDispatch;
8667: Func GetActiveOleObject(const ClassName:Str): IDispatch;
8668: Func ProgIDToClassID(const ProgID:Str): TGUID;
8669: Func ClassIDToProgID(const ClassID: TGUID):Str;
8670: Func CreateClassID:Str;
8671: Func CreateGUIDString:Str;
8672: Func CreateGUIDID:Str;
8673: Proc OleError(ErrorCode: longint)
8674: Proc OleCheck(Result: HRESULT);
8675: end;
8676:
8677: Func xCreateOleObject( const ClassName :Str) : Variant //or IDispatch
8678: Func xGetActiveOleObject( const ClassName :Str) : Variant
8679: //Func DllGetClassObject(const CLSID:TCLSID;const IID TIID;var Obj):HRESULT
8680: Func DllCanUnloadNow : HRESULT
8681: Func DllRegisterServer : HRESULT
8682: Func DllUnregisterServer : HRESULT
8683: Func VarFromInterface( Unknown : IUnknown) : Variant
8684: Func VarToInterface( const V : Variant) : IDispatch
8685: Func VarToAutoObject( const V : Variant) : TAutoObject
8686: //Proc DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8687: //Proc DispInvokeError( Status : HRESULT; const ExcepInfo : TExcepInfo)
8688: Proc OleError( ErrorCoder : HRESULT)
8689: Proc OleCheck( Result : HRESULT)
8690: Func StringToClassID( const S :Str) : TCLSID
8691: Func ClassIDToString( const ClassID : TCLSID) :Str
8692: Func xProgIDToClassID( const ProgID :Str) : TCLSID
8693: Func xClassIDToProgID( const ClassID : TCLSID) :Str
8694: Func xWideCompareStr( const S1, S2 : WideString) : Int
8695: Func xWideSameStr( const S1, S2 : WideString) :Bool
8696: Func xGUIDToString( const ClassID : TGUID) :Str
8697: Func xStringToGUID( const S :Str) : TGUID
8698: Func xGetModuleName( Module : HMODULE) :Str
8699: Func xAcquireExceptionObject : TObject
8700: Func xIfThen(AValue :Bool; const ATrue: Int; const AFalse: Int): Int
8701: Func xUtf8Encode( const WS : WideString) : UTF8String
8702: Func xUtf8Decode( const S : UTF8String) : WideString

```

```

8703: Func xExcludeTrailingPathDelimiter( const S :Str) :Str
8704: Func xIncludeTrailingPathDelimiter( const S :Str) :Str
8705: Func XRTLHandleCOMException : HRESULT
8706: Proc XRTLCheckArgument( Flag :Bool)
8707: //Proc XRTLCheckOutArgument( out Arg)
8708: Proc XRTLInterfaceConnect(const Source:IUnknown,const IID:TIID,const Sink:IUnknown,var
Connection:Longint);
8709: Proc XRTLInterfaceDisconnect(const Source:IUnknown,const IID:TIID,var Connection:Longint)
8710: Func XRTLRegisterActiveObject(const Unk:IUnknown,ClassID:TCLSID,Flags:DWORD,var
RegisterCookie:Int):HRESULT
8711: Func XRTLUnRegisterActiveObject( var RegisterCookie : Int) : HRESULT
8712: //Func XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HRESULT
8713: Proc XRTLEnumActiveObjects( Strings : TStrings)
8714: Func XRTLDefaultCategoryManager: IUnknown;
8715: Func XRTLIsCategoryEmpty(CatID:TGUID,const CategoryManager:IUnknown=nil):Bool;
8716: // ICatRegister helper functions
8717: Func XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
LocaleID: TLCID = LOCALE_USER_DEFAULT;
8718: const CategoryManager: IUnknown = nil): HRESULT;
8719: Func XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
LocaleID: TLCID = LOCALE_USER_DEFAULT;
8720: const CategoryManager: IUnknown = nil): HRESULT;
8721: Func XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8722: const CategoryManager: IUnknown = nil): HRESULT;
8723: Func XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8724: const CategoryManager: IUnknown = nil): HRESULT;
8725: // ICatInformation helper functions
8726: Func XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
LocaleID: TLCID = LOCALE_USER_DEFAULT;
8727: const CategoryManager: IUnknown = nil): HRESULT;
8728: Func XRTLGetCategoryList( Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8729: const CategoryManager: IUnknown = nil): HRESULT;
8730: Func XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8731: const CategoryManager: IUnknown = nil): HRESULT;
8732: Func XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8733: const CategoryManager: IUnknown = nil): HRESULT;
8734: Func XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8735: const ADelete:Bool = True): WideString;
8736: Func XRTLRLPos(const ASub, AIn: WideString; AStart: Int = -1): Int;
8737: Func XRTLGetVariantAsString( const Value : Variant) :Str
8738: Func XRTLDateTimeToTimeZoneTime(DT:TDateTime,TimeZone:TXRTLTimeZone):TDateTime
8739: Func XRTLGetTimeZones : TXRTLTimeZones
8740: Func XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8741: Func DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8742: Func GMTNow : TDateTime
8743: Func GMTToLocalTime( GMT : TDateTime) : TDateTime
8744: Func LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8745: Proc XRTLNotImplemented
8746: Proc XRTLRaiseError( E : Exception)
8747: Proc XRTLRaise( E : Exception);
8748: Proc XRLaise( E : Exception);
8749: Proc XRTLInvalidOperation(ClassName:str;OperationName:str;Description:Str)
8750: Proc SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8751: begin
8752: SIRegister_IXRTLValue(CL);
8753: SIRegister_TXRTLValue(CL);
8754: //AddTypes('PXRTLValueArray', '^TXRTLValueArray // will not work
8755: AddTypes('TXRTLValueArray', 'array of IXRTLValue
8756: Func XRTLValue( const AValue :Card) : IXRTLValue;
8757: Func XRTLSetValue(const IValue:IXRTLValue; const AValue:Card) :Card;
8758: Func XRTLGetAsCardinal( const IValue : IXRTLValue) :Card
8759: Func XRTLGetAsCardinalDef(const IValue:IXRTLValue,const DefValue:Card) : Card
8760: Func XRTLValue1( const AValue : Int) : IXRTLValue;
8761: Func XRTLSetValue1( const IValue : IXRTLValue; const AValue : Int) : Int;
8762: Func XRTLGetAsInt( const IValue : IXRTLValue) : Int
8763: Func XRTLGetAsIntDef( const IValue : IXRTLValue; const DefValue : Int) : Int
8764: Func XRTLValue2( const AValue : Int64) : IXRTLValue;
8765: Func XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8766: Func XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8767: Func XRTLGetAsInt64Def(const IValue:IXRTLValue,const DefValue:Int64) : Int64
8768: Func XRTLValue3( const AValue : Single) : IXRTLValue;
8769: Func XRTLSetValue3( const IValue : IXRTLValue; const AValue:Single) : Single;
8770: Func XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8771: Func XRTLGetAsSingleDef(const IValue:IXRTLValue,const DefValue:Single):Single
8772: Func XRTLValue4( const AValue : Double) : IXRTLValue;
8773: Func XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8774: Func XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8775: Func XRTLGetAsDoubleDef(const IValue:IXRTLValue,const DefValue:Double):Double
8776: Func XRTLValue5( const AValue : Extended) : IXRTLValue;
8777: Func XRTLSetValue5(const IValue:IXRTLValue,const AValue:Extended): Extended;
8778: Func XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8779: Func XRTLGetAsExtendedDef(const IValue:IXRTLValue,const DefValue:Extended): Extended
8780: Func XRTLValue6( const AValue : IInterface) : IXRTLValue;
8781: Func XRTLSetValue6(const IValue:IXRTLValue,const AValue:IInterface):IInterface;
8782: Func XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8783: //Func XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8784: Func XRTLGetAsInterfaceDef(const IValue:IXRTLValue,const DefValue:IInterface):IInterface;
8785: Func XRTLValue7( const AValue : WideString) : IXRTLValue;

```

```

8790: Func XRTLSetValue7(const IValue:IXRTLValue;const AValue:WideString):WideString;
8791: Func XRTLGetAsWideString(const IValue : IXRTLValue) : WideString
8792: Func XRTLGetAsWideStringDef(const IValue:IXRTLValue;const DefValue:WideString): WideString
8793: Func XRTLValue8(const AValue:TObject; const AOwnValue:Boolean): IXRTLValue;
8794: Func XRTLSetValue8(const IValue:IXRTLValue;const AValue:TObject) : TObject;
8795: Func XRTLGetAsObject(const IValue:IXRTLValue;const ADetachOwnership:Bool): TObject;
8796: Func XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
ADetachOwnership:Boolean):TObject;
8797: //Func XRTLValue9(const AValue : __Pointer) : IXRTLValue;
8798: //Func XRTLSetValue9(const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8799: //Func XRTLGetAsPointer(const IValue : IXRTLValue) : __Pointer
8800: //Func XRTLGetAsPointerDef(const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8801: Func XRTLValueV(const AValue : Variant) : IXRTLValue;
8802: Func XRTLSetValueV(const IValue:IXRTLValue; const AValue : Variant): Variant;
8803: Func XRTLGetAsVariant(const IValue : IXRTLValue) : Variant
8804: Func XRTLGetAsVariantDef(const IValue:IXRTLValue;const DefValue:Variant): Variant
8805: Func XRTLValueI0(const AValue : Currency) : IXRTLValue;
8806: Func XRTLSetValueI0(const IValue:IXRTLValue; const AValue:Currency):Currency;
8807: Func XRTLGetAsCurrency(const IValue : IXRTLValue) : Currency
8808: Func XRTLGetAsCurrencyDef(const IValue:IXRTLValue;const DefValue:Currency): Currency
8809: Func XRTLValueI1(const AValue : Comp) : IXRTLValue;
8810: Func XRTLSetValueI1(const IValue : IXRTLValue; const AValue : Comp) : Comp;
8811: Func XRTLGetAsComp(const IValue : IXRTLValue) : Comp
8812: Func XRTLGetAsCompDef(const IValue : IXRTLValue; const DefValue : Comp) : Comp
8813: Func XRTLValueI2(const AValue : TClass) : IXRTLValue;
8814: Func XRTLSetValueI2(const IValue:IXRTLValue; const AValue : TClass) : TClass;
8815: Func XRTLGetAsClass(const IValue : IXRTLValue) : TClass
8816: Func XRTLGetAsClassDef(const IValue:IXRTLValue;const DefValue:TClass):TClass
8817: Func XRTLValueI3(const AValue : TGUID) : IXRTLValue;
8818: Func XRTLSetValueI3(const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8819: Func XRTLGetAsGUID(const IValue : IXRTLValue) : TGUID
8820: Func XRTLGetAsGUIDDef(const IValue:IXRTLValue; const DefValue : TGUID) : TGUID
8821: Func XRTLValueI4(const AValue :Bool) : IXRTLValue;
8822: Func XRTLSetValueI4(const IValue:IXRTLValue;const AValue:Boolean):Boolean;
8823: Func XRTLGetAsBoolean(const IValue : IXRTLValue) :Bool
8824: Func XRTLGetAsBooleanDef(const IValue:IXRTLValue;const DefValue:Bool):Bool
8825: end;
8826:
8827: //*****unit uPSI_GR32;*****
8828:
8829: Func Color32( WinColor : TColor) : TColor32;
8830: Func Color32I( R, G, B : Byte; A : Byte) : TColor32;
8831: Func Color32I2( Index : Byte; var Palette : TPalette32) : TColor32;
8832: Func Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8833: Func WinColor( Color32 : TColor32) : TColor
8834: Func ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8835: Proc Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8836: Proc Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8837: Func Color32Components( R, G, B, A :Bool) : TColor32Components
8838: Func RedComponent( Color32 : TColor32) : Int
8839: Func GreenComponent( Color32 : TColor32) : Int
8840: Func BlueComponent( Color32 : TColor32) : Int
8841: Func AlphaComponent( Color32 : TColor32) : Int
8842: Func Intensity( Color32 : TColor32) : Int
8843: Func SetAlpha( Color32 : TColor32; NewAlpha : Int) : TColor32
8844: Func HSLtoRGB( H, S, L : Single) : TColor32;
8845: Proc RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8846: Func HSLtoRGBI( H, S, L : Int) : TColor32;
8847: Proc RGBtoHSLI( RGB : TColor32; out H, S, L : Byte);
8848: Func WinPalette( const P : TPalette32) : HPALETTE
8849: Func FloatPoint( X, Y : Single) : TFloatPoint;
8850: Func FloatPointI( const P : TPoint) : TFloatPoint;
8851: Func FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8852: Func FixedPoint( X, Y : Int) : TFixedPoint;
8853: Func FixedPointI( X, Y : Single): TFixedPoint;
8854: Func FixedPoint2( const P : TPoint) : TFixedPoint;
8855: Func FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8856: AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8857: Func MakeRect( const L, T, R, B : Int) : TRect;
8858: Func MakeRectI( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8859: Func MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8860: Func GFixedRect( const L, T, R, B : TFixed) : TRect;
8861: Func FixedRectI( const ARect : TRect) : TRect;
8862: Func FixedRect2( const FR : TFloatRect) : TRect;
8863: Func GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8864: Func FloatRectI( const ARect : TRect) : TFloatRect;
8865: Func FloatRect2( const FXR : TRect) : TFloatRect;
8866: Func GIntersectRect( out Dst : TRect; const R1, R2 : TRect) :Bool;
8867: Func IntersectRectI(out Dst:TFloatRect;const FR1,FR2:TFloatRect):Boolean;
8868: Func GUnionRect( out Rect : TRect; const R1, R2 : TRect) :Bool;
8869: Func UnionRectI( out Rect : TFloatRect; const R1, R2 : TFloatRect):Boolean;
8870: Func GEqualRect( const R1, R2 : TRect) :Bool;
8871: Func EqualRectI( const R1, R2 : TFloatRect) :Bool;
8872: Proc GInflateRect( var R : TRect; Dx, Dy : Int);
8873: Proc InflateRectI( var FR : TFloatRect; Dx, Dy : TFloat);
8874: Proc GOffsetRect( var R : TRect; Dx, Dy : Int);
8875: Proc OffsetRectI( var FR : TFloatRect; Dx, Dy : TFloat);
8876: Func IsRectEmpty( const R : TRect) :Bool;
8877: Func IsRectEmptyI( const FR : TFloatRect) :Bool;

```

```

8878: Func GpInRect( const R : TRect; const P : TPoint) :Bool;
8879: Func PtInRect1( const R : TFloatRect; const P : TPoint) :Bool;
8880: Func PtInRect2( const R : TRect; const P : TFloatPoint) :Bool;
8881: Func PtInRect3( const R : TFloatRect; const P : TFloatPoint) :Bool;
8882: Func EqualRectSize( const R1, R2 : TRect) :Bool;
8883: Func EqualRectSize1( const R1, R2 : TFloatRect) :Bool;
8884: Func MessageBeep( uType : UINT) : BOOL
8885: Func ShowCursor( bShow : BOOL) : Int
8886: Func SetCursorPos( X, Y : Int) : BOOL
8887: Func SetCursor( hCursor : HICON) : HCURSOR
8888: Func GetCursorPos( var lpPoint : TPoint) : BOOL
8889: //Func ClipCursor( lpRect : PRect) : BOOL
8890: Func GetClipCursor( var lpRect : TRect) : BOOL
8891: Func GetCursor : HCURSOR
8892: Func CreateCaret(hWnd: HWND;hBitmap:HBITMAP;nWidth, nHeight:Int):BOOL
8893: Func GetCaretBlinkTime : UINT
8894: Func SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8895: Func DestroyCaret : BOOL
8896: Func HideCaret( hWnd : HWND) : BOOL
8897: Func ShowCaret( hWnd : HWND) : BOOL
8898: Func SetCaretPos( X, Y : Int) : BOOL
8899: Func GetCaretPos( var lpPoint : TPoint) : BOOL
8900: Func ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8901: Func ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8902: Func MapWindowPoints( hWndFrom,hWndTo:HWND; var lpPoints, cPoints: UINT):Int
8903: Func WindowFromPoint( Point : TPoint) : HWND
8904: Func ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8905:
8906: Proc SIRegister_GR32_Math(CL: TPSPascalCompiler);
8907: begin
8908:   Func FixedFloor( A : TFixed) : Int
8909:   Func FixedCeil( A : TFixed) : Int
8910:   Func FixedMul( A, B : TFixed) : TFixed
8911:   Func FixedDiv( A, B : TFixed) : TFixed
8912:   Func OneOver( Value : TFixed) : TFixed
8913:   Func FixedRound( A : TFixed) : Int
8914:   Func FixedSqr( Value : TFixed) : TFixed
8915:   Func FixedSqrtLP( Value : TFixed) : TFixed
8916:   Func FixedSqrtHP( Value : TFixed) : TFixed
8917:   Func FixedCombine( W, X, Y : TFixed) : TFixed
8918:   Proc GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8919:   Proc GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8920:   Func GRHypot( const X, Y : TFloat) : TFloat;
8921:   Func Hypot1( const X, Y : Int) : Int;
8922:   Func FastSqrt( const Value : TFloat) : TFloat
8923:   Func FastSqrtBab1( const Value : TFloat) : TFloat
8924:   Func FastSqrtBab2( const Value : TFloat) : TFloat
8925:   Func FastInvSqrt( const Value : Single) : Single;
8926:   Func MulDiv( Multiplicand, Multiplier, Divisor : Int) : Int
8927:   Func GRIsPowerOf2( Value : Int) :Bool
8928:   Func PrevPowerOf2( Value : Int) : Int
8929:   Func NextPowerOf2( Value : Int) : Int
8930:   Func Average( A, B : Int) : Int
8931:   Func GRSign( Value : Int) : Int
8932:   Func FloatMod( x, y : Double) : Double
8933: end;
8934:
8935: Proc SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8936: begin
8937:   Func Clamp( const Value : Int) : Int;
8938:   Proc GRFillWord( var X, Count :Card; Value : Longword)
8939:   Func StackAlloc( Size : Int) : Pointer
8940:   Proc StackFree( P : Pointer)
8941:   Proc Swap( var A, B : Pointer);
8942:   Proc Swap1( var A, B : Int);
8943:   Proc Swap2( var A, B : TFixed);
8944:   Proc Swap3( var A, B : TColor32);
8945:   Proc TestSwap( var A, B : Int);
8946:   Proc TestSwap1( var A, B : TFixed);
8947:   Func TestClip( var A, B : Int; const Size : Int) :Bool;
8948:   Func TestClip1( var A, B : Int; const Start, Stop : Int) :Bool;
8949:   Func GRConstrain( const Value, Lo, Hi : Int) : Int;
8950:   Func Constrain1( const Value, Lo, Hi : Single) : Single;
8951:   Func SwapConstrain( const Value:Int; Constrain1,Constrain2:Int) : Int
8952:   Func GRMin( const A, B, C : Int) : Int;
8953:   Func GRMax( const A, B, C : Int) : Int;
8954:   Func Clamp( Value, Max : Int) : Int;
8955:   Func Clamp1( Value, Min, Max : Int) : Int;
8956:   Func Wrap( Value, Max : Int) : Int;
8957:   Func Wrap1( Value, Min, Max : Int) : Int;
8958:   Func Wrap3( Value, Max : Single) : Single;;
8959:   Func WrapPow2( Value, Max : Int) : Int;
8960:   Func WrapPow21( Value, Min, Max : Int) : Int;
8961:   Func Mirror( Value, Max : Int) : Int;
8962:   Func Mirror1( Value, Min, Max : Int) : Int;
8963:   Func MirrorPow2( Value, Max : Int) : Int;
8964:   Func MirrorPow21( Value, Min, Max : Int) : Int;
8965:   Func GetOptimalWrap( Max : Int) : TWrapProc;
8966:   Func GetOptimalWrap1( Min, Max : Int) : TWrapProcEx;

```

```

8967: Func GetOptimalMirror( Max : Int) : TWrapProc;
8968: Func GetOptimalMirror1( Min, Max : Int) : TWrapProcEx;
8969: Func GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8970: Func GetWrapProc1( WrapMode : TWrapMode; Max : Int) : TWrapProc;
8971: Func GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8972: Func GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Int):TWrapProcEx;
8973: Func Div255( Value :Card) :Card
8974: Func SAR_4( Value : Int) : Int
8975: Func SAR_8( Value : Int) : Int
8976: Func SAR_9( Value : Int) : Int
8977: Func SAR_11( Value : Int) : Int
8978: Func SAR_12( Value : Int) : Int
8979: Func SAR_13( Value : Int) : Int
8980: Func SAR_14( Value : Int) : Int
8981: Func SAR_15( Value : Int) : Int
8982: Func SAR_16( Value : Int) : Int
8983: Func ColorSwap( WinColor : TColor) : TColor32
8984: end;
8985:
8986: Proc SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8987: begin
8988:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8989:   Proc CopyComponents(Dst,Src:TCustomBitmap32;Components:TColor32Components);
8990:   Proc CopyComponents1(Dst:TCustomBmap32;DstX,
   DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8991:   Proc AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8992:   Proc ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha :Bool)
8993:   Proc IntensityToAlpha( Dst, Src : TCustomBitmap32)
8994:   Proc Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8995:   Proc InvertRGB( Dst, Src : TCustomBitmap32)
8996:   Proc ApplyLUT(Dst,Src:TCustomBitmap32;const LUT:TLUT8;PreserveAlpha:Boolean)
8997:   Proc ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8998:   Func CreateBitmask( Components : TColor32Components) : TColor32
8999:   Proc ApplyBitmask(Dst:TCustomBitmap32;DstX,DstY:Int;Src:TCustomBitmap32;SrcRect:TRect;
   Bitmask:TColor32;LogicalOperator:TLogicalOperator);
9000:   Proc ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
9001:   Proc CheckParams( Dst, Src : TCustomBitmap32; ResizeDst :Bool)
9002: end;
9003:
9004: Proc SIRegister_JclNTFS(CL: TPSPascalCompiler);
9005: begin
9006:   AddClassN(FindClass('TObject'),'EJclNtfsError
9007:   AddTypeS('TFileCompressionState','(fcNoCompression,fcDefaultCompression,fcLZNT1Compression )
9008:   Func NtfsGetCompression const FileName:str; var State : Short):Bool;
9009:   Func NtfsGetCompression1( const FileName :Str) : TFileCompressionState;
9010:   Func NtfsSetCompression const FileName:str; const State:Short) :Bool
9011:   Proc NtfsSetFileCompression(const FileName:Str; const State: TFileCompressionState)
9012:   Proc NtfsSetDirectoryTreeCompression(const Directory:Str; const State : TFileCompressionState)
9013:   Proc NtfsSetDefaultFileCompression(const Directory:Str; const State:TFileCompressionState)
9014:   Proc NtfsSetPathCompression(const Path:str;const State:TFileCompressionState;Recursive:Bool;
9015:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Int; Data : PFileAlloca'
9016:   //+ 'tedRangeBuffer; MoreData :Bool; end
9017:   Func NtfsSetSparse( const FileName :Str) :Bool
9018:   Func NtfsZeroDataByHandle(const Handle:THandle;const First,Last:Int64): Bool
9019:   Func NtfsZeroDataByName(const FileName:str;const First,Last:Int64): Bool
9020:   //Func NtfsQueryAllocRanges(const FileName:str;Offset,Count:Int64;var Ranges:TNtfsAllocRanges):Boolean;
9021:   //Func NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges; Index:Int):TFileAllocatedRangeBuffer
9022:   Func NtfsSparseStreamsSupported( const Volume :Str) :Bool
9023:   Func NtfsGetSparse( const FileName :Str) :Bool
9024:   Func NtfsDeleteReparsePoint(const FileName:str; ReparseTag:DWORD):Boolean
9025:   Func NtfsSetReparsePoint(const FileName:str;var ReparseData,Size:Longword): Bool
9026:   //Func NtfsGetReparsePoint(const FileName:str;var ReparseData:TReparseGuidDataBuffer):Bool
9027:   Func NtfsGetReparseTag( const Path :Str; var Tag : DWORD) :Bool
9028:   Func NtfsReparsePointsSupported( const Volume :Str) :Bool
9029:   Func NtfsFileHasReparsePoint( const Path :Str) :Bool
9030:   Func NtfsIsFolderMountPoint( const Path :Str) :Bool
9031:   Func NtfsMountDeviceAsDrive( const Device :Str; Drive : Char):Boolean
9032:   Func NtfsMountVolume(const Volume:Char; const MountPoint:str):Bool
9033:   AddTypeS('TOpLock','( olExclusive, olReadOnly, olBatch, olFilter )
9034:   Func NtfsOpLockAckClosePending(Handle:THandle;Overlapped:TOverlapped):boolean
9035:   Func NtfsOpLockBreakAckNo2(Handle:THandle;Overlapped:TOverlapped):Bool
9036:   Func NtfsOpLockBreakAcknowledge(Handle:THandle;Overlapped:TOverlapped):Boolean
9037:   Func NtfsOpLockBreakNotify(Handle:THandle;Overlapped:TOverlapped) :Bool
9038:   Func NtfsRequestOpLock(Handle:THandle;Kind:TOpLock;Overlapped:TOverlapped): Bool
9039:   Func NtfsCreateJunctionPoint( const Source, Destination :Str) :Bool
9040:   Func NtfsDeleteJunctionPoint( const Source :Str) :Bool
9041:   Func NtfsGetJunctionPointDestination(const Source:Str;var Destination:str):Bool
9042:   AddTypeS('TStreamId','( siInvalid, siStandard, siExtendedAttribute, siSec'
   + 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
9043:   AddTypeS('TStreamIds', 'set of TStreamId
9044:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
   + ': __Pointer; StreamIds : TStreamIds; end
9045:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
   + 'tributes: DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
9046:   Func NtfsFindFirstStream(const FileName:str;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
9047:   Func NtfsFindNextStream( var Data : TFindStreamData) :Bool
9048:   Func NtfsFindStreamClose( var Data : TFindStreamData) :Bool
9049:   Func NtfsCreateHardLink(const LinkFileName,ExistingFileName:str):Boolean
9050:   AddTypeS('TNtfsHardLinkInfo','record LinkCount:Card;FileIndex:Int64; end

```



```

9054: Func NtfsGetHardLinkInfo( const FileName :Str; var Info : TNtfsHardLinkInfo ) :Bool
9055: Func NtfsFindHardLinks(const Path:str;const FileIndexHigh,FileIndexLow:Card;const List:TStrings):Bool;
9056: Func NtfsDeleteHardLinks( const FileName :Str) :Bool
9057: Func JclAppInstances : TJclAppInstances;
9058: Func JclAppInstances1( const UniqueAppIdGuidStr :Str ) : TJclAppInstances;
9059: Func ReadMessageCheck( var Message:TMessage;const IgnoredOriginatorWnd:HWND) : TJclAppInstDataKind
9060: Proc ReadMessageData(const Message:TMessage;var Data:___Pointer;var Size:Int)
9061: Proc ReadMessageString( const Message : TMessage; var S :Str)
9062: Proc ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
9063:
9064: (*-----*)
9065: Proc SIRegister_JclGraphics(CL: TPSPascalCompiler);
9066: begin
9067:   FindClass('TObject'),'EJclGraphicsError
9068:   TDynDynIntArrayArray', 'array of TDynIntArray
9069:   TDynPointArray', 'array of TPoint
9070:   TDynDynPointArrayArray', 'array of TDynPointArray
9071:   TPointF', 'record X : Single; Y : Single; end
9072:   TDynPointArrayF', 'array of TPointF
9073:   TDrawMode2', '( dmOpaque, dmBlend )
9074:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
9075:   TConversionKind', '(ckRed,ckGreen,ckBlue,ckAlpha ckUniformRGB, ckWeightedRGB )
9076:   TResamplingFilter', '(rfBox,rfTriangle,rfHermite,rfBell,rfSpline,rfLanczos3,rfMitchell )
9077:   TMatrix3d', 'record array[0..2,0..2] of extended end
9078:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
9079:   TScanLine', 'array of Int
9080:   TScanLines', 'array of TScanLine
9081:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
9082:   TGradientDirection', '( gdVertical, gdHorizontal )
9083:   TPolyFillMode', '( fmAlternate, fmWinding )
9084:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
9085:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
9086:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
9087:   SIRegister_TJclDesktopCanvas(CL);
9088:   FindClass('TObject'),'TJclRegion
9089:   SIRegister_TJclRegionInfo(CL);
9090:   SIRegister_TJclRegion(CL);
9091:   SIRegister_TJclThreadPersistent(CL);
9092:   SIRegister_TJclCustomMap(CL);
9093:   SIRegister_TJclBitmap32(CL);
9094:   SIRegister_TJclByteMap(CL);
9095:   SIRegister_TJclTransformation(CL);
9096:   SIRegister_TJclLinearTransformation(CL);
9097:   Proc Stretch(NewWidth,
NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
9098:   Proc Stretch1(NewWidth,NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
9099:   Proc DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Int)
9100:   Func GetAntialiasedBitmap( const Bitmap : TBitmap ) : TBitmap
9101:   Proc BitmapToJPeg( const FileName :Str)
9102:   Proc JPegToBitmap( const FileName :Str)
9103:   Func ExtractIconCount( const FileName :Str ) : Int
9104:   Func BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Int ) : HICON
9105:   Func IconToBitmapJ( Icon : HICON ) : HBITMAP
9106:   Proc BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
9107:   Proc
StretchTransfer(Dst:TJclBitmap32;DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter;
CombineOp:TDrawMode)
9108:   Proc Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
9109:   Proc SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
9110:   Func FillGradient(DC:HDC;ARect:TRect;ColorCnt:Int;StartColor,
EndColor:TColor;ADirection:TGradientDirection):Bool;
9111:   Func CreateRegionFromBitmap(Bitmap:TBitmap;RegionColor:TColor;RegionBitmapMode:TJclRegionBitmapMode): HRGN
9112:   Proc ScreenShot(bm:TBitmap; Left, Top, Width, Height : Int; Window : HWND);
9113:   Proc ScreenShot1( bm : TBitmap);
9114:   Proc PolyLineTS(Bitmap TJclBitmap32;const Points:TDynPointArray;Color: TColor32)
9115:   Proc PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
9116:   Proc PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
9117:   Proc PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
9118:   Proc PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
9119:   Proc PolygonFS(Bitmap:TJclBitmap32;const Points:TDynPointArrayF;Color: TColor32)
9120:   Proc PolyPolygonTS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
9121:   Proc PolyPolygonAS(Bitm:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
9122:   Proc PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Colr:TColr32);
9123:   Proc AlphaToGrayscale( Dst, Src : TJclBitmap32)
9124:   Proc IntensityToAlpha( Dst, Src : TJclBitmap32)
9125:   Proc Invert( Dst, Src : TJclBitmap32)
9126:   Proc InvertRGB( Dst, Src : TJclBitmap32)
9127:   Proc ColorToGrayscale( Dst, Src : TJclBitmap32)
9128:   Proc ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
9129:   Proc SetGamma( Gamma : Single)
9130: end;
9131:
9132: (*-----*)
9133: Proc SIRegister_JclSynch(CL: TPSPascalCompiler);
9134: begin
9135:   Func LockedAdd( var Target : Int; Value : Int) : Int
9136:   Func LockedCompareExchange( var Target : Int; Exch, Comp : Int) : Int;
9137:   Func LockedCompareExchangel(var Target: ___Pointer;Exch,Comp:___Pointer): Pointer;
9138:   Func LockedDec( var Target : Int) : Int

```

```

9139: Func LockedExchange( var Target : Int; Value : Int) : Int
9140: Func LockedExchangeAdd( var Target : Int; Value : Int) : Int
9141: Func LockedExchangeDec( var Target : Int) : Int
9142: Func LockedExchangeInc( var Target : Int) : Int
9143: Func LockedExchangeSub( var Target : Int; Value : Int) : Int
9144: Func LockedInc( var Target : Int) : Int
9145: Func LockedSub( var Target : Int; Value : Int) : Int
9146: TjclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
9147: SIRegister_TJclDispatcherObject(CL);
9148: Func WaitForMultipleObjects(const Objects:array of TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardin):Card;
9149: Func WaitAlertableForMultipleObjects(const Objects:array of
TJclDispatcherObject;WaitAll:Bool;TimeOut:Card):Card
9150: SIRegister_TJclCriticalSection(CL);
9151: SIRegister_TJclCriticalSectionEx(CL);
9152: SIRegister_TJclEvent(CL);
9153: SIRegister_TJclWaitableTimer(CL);
9154: SIRegister_TJclSemaphore(CL);
9155: SIRegister_TJclMutex(CL);
9156: POptexSharedInfo', '^TOptexSharedInfo // will not work
9157: TOptexSharedInfo,record SpinCount:Int;LockCount:Int;ThreadId:Longword;RecursionCount:Int;end
9158: SIRegister_TJclOptex(CL);
9159: TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
9160: TMrewThreadInfo', 'record ThreadId:Longword;RecursionCount:Int;Reader:Bool; end
9161: TMrewThreadInfoArray', 'array of TMrewThreadInfo
9162: SIRegister_TJclMultiReadExclusiveWrite(CL);
9163: PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
9164: TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
+ 'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
9165: PMeteredSection', 'TMeteredSection // will not work
9166: TMeteredSection', 'record Event:THandle; FileMap:THandle; SharedInfo:PMetSectSharedInfo; end
9167: SIRegister_TJclMeteredSection(CL);
9168: TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
9169: TMutexInfo', 'record SignalState:Longint; Owned:Boolean; Abandoned:Bool; end
9170: TSemaphoreCounts', 'record CurrentCount: Longint; MaximumCount: Longint; end
9171: TTimerInfo', 'record Remaining : TLargeInt; Signaled : LongBool; end
9172: Func QueryCriticalSection(CS TJclCriticalSection;var Info:TRTLCriticalSection):Boolean
9173: Func QueryEvent( Handle : THandle; var Info : TEventInfo) :Bool
9174: Func QueryMutex( Handle : THandle; var Info : TMutexInfo) :Bool
9175: Func QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) :Bool
9176: Func QueryTimer( Handle : THandle; var Info : TTimerInfo) :Bool
9177: FindClass('TOBJECT'),'EJclWin32HandleObjectError
9178: FindClass('TOBJECT'),'EJclDispatcherObjectError
9179: FindClass('TOBJECT'),'EJclCriticalSectionError
9180: FindClass('TOBJECT'),'EJclEventError
9181: FindClass('TOBJECT'),'EJclWaitableTimerError
9182: FindClass('TOBJECT'),'EJclSemaphoreError
9183: FindClass('TOBJECT'),'EJclMutexError
9184: FindClass('TOBJECT'),'EJclMeteredSectionError
9185: end;
9186:
9187:
9188: //*****unit uPSI_mORMotReport;
9189: Proc SetCurrentPrinterAsDefault
9190: Func CurrentPrinterName :Str
9191: Func mCurrentPrinterPaperSize :Str
9192: Proc UseDefaultPrinter
9193:
9194: Proc SIRegisterTSTREAM(CI: TPSPascalCompiler);
9195: begin
9196: with FindClass('TOBJECT'), 'TStream') do begin
9197: IsAbstract := True;
9198: //RegisterMethod('Func Read( var Buffer, Count : Longint) : Longint
9199: // RegisterMethod('Func Write( const Buffer, Count : Longint) : Longint
9200: Func Read(Buffer:str;Count:LongInt):LongInt
9201: Func Write(Buffer:str;Count:LongInt):LongInt
9202: Func ReadString(Buffer:str;Count:LongInt):LongInt //FileStream
9203: Func WriteString(Buffer:str;Count:LongInt):LongInt
9204: Func ReadInt(Buffer:Int;Count:LongInt):LongInt
9205: Func WriteInt(Buffer:Int;Count:LongInt):LongInt
9206: Func ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt;
9207: Func WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt;
9208:
9209: Proc ReadAB(Buffer: TByteArray;Count:LongInt)
9210: Proc WriteAB(Buffer: TByteArray;Count:LongInt)
9211: Proc ReadABD(Buffer: TByteDynArray;Count:LongInt)
9212: Proc WriteABD(Buffer: TByteDynArray;Count:LongInt)
9213: Proc ReadAC(Buffer: TCharArray;Count:LongInt)
9214: Proc WriteAC(Buffer: TCharArray;Count:LongInt)
9215: Proc ReadACD(Buffer: TCharDynArray;Count:LongInt)
9216: Proc WriteACD(Buffer: TCharDynArray;Count:LongInt)
9217:
9218: Func Seek(Offset:LongInt;Origin:Word):LongInt
9219: Proc ReadBuffer(Buffer:str;Count:LongInt)
9220: Proc WriteBuffer(Buffer:str;Count:LongInt)
9221: Proc ReadBufferInt(Buffer:Int;Count:LongInt);
9222: Proc WriteBufferInt(Buffer:Int;Count:LongInt);
9223: Proc ReadBufferFloat(Buffer:Double;Count:LongInt);
9224: Proc WriteBufferFloat(Buffer:Double;Count:LongInt);
9225:
9226: Proc ReadBufferAB(Buffer: TByteArray;Count:LongInt)

```

```

9227: Proc WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9228: Proc ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9229: Proc WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9230: Proc ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9231: Proc WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9232: Proc ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9233: Proc WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9234: Proc ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9235: Proc WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9236: Proc ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9237: Proc WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9238: Proc ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9239: Proc WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9240:
9241: Proc ReadBufferP(Buffer: PChar;Count:LongInt)
9242: Proc WriteBufferP(Buffer: PChar;Count:LongInt)
9243: Proc ReadBufferO(Buffer: TObject;Count:LongInt);
9244: Proc WriteBufferO(Buffer: TObject;Count:LongInt);
9245: //READBUFFERAC
9246: Func InstanceSize: Longint
9247: Proc FixupResourceHeader( FixupInfo : Int)
9248: Proc ReadResHeader
9249: {$IFDEF DELPHI4UP}
9250: Func CopyFrom(Source:TStream;Count:Int64):LongInt
9251: {$ELSE}
9252: Func CopyFrom(Source:TStream;Count:Int):LongInt
9253: {$ENDIF}
9254: RegisterProperty('Position', 'LongInt', iptrw);
9255: RegisterProperty('Size', 'LongInt', iptrw);
9256: end;
9257: end;
9258:
9259: { *****
9260:   Unit DMATH - Interface for DMATH.DLL
9261:   ***** }
9262: // see more docs/dmath_manual.pdf
9263:
9264: Func InitEval : Int
9265: Proc SetVariable( VarName : Char; Value : Float)
9266: Proc SetFunction( FuncName :Str; Wrapper : TWrapper)
9267: Func Eval( ExpressionString :Str) : Float
9268:
9269: unit dmath; //types are in built, others are external in DLL
9270: interface
9271: {$IFDEF DELPHI}
9272: uses
9273:   StdCtrls, Graphics;
9274: {$ENDIF}
9275: { -----
9276:   Types and constants
9277:   ----- }
9278: {$i types.inc}
9279: { -----
9280:   Error handling
9281:   ----- }
9282: Proc SetErrCode(ErrCode : Int); external 'dmath';
9283: { Sets the error code }
9284: Func DefaultVal(ErrCode : Int; DefVal : Float) : Float; external 'dmath';
9285: { Sets error code and default Func value }
9286: Func MathErr : Int; external 'dmath';
9287: { Returns the error code }
9288: { -----
9289:   Dynamic arrays
9290:   ----- }
9291: Proc SetAutoInit(AutoInit :Bool); external 'dmath';
9292: { Sets the auto-initialization of arrays }
9293: Proc DimVector(var V : TVector; Ub : Int); external 'dmath';
9294: { Creates floating point vector V[0..Ub] }
9295: Proc DimIntVector(var V : TIntVector; Ub : Int); external 'dmath';
9296: { Creates Int vector V[0..Ub] }
9297: Proc DimCompVector(var V : TCompVector; Ub : Int);external 'dmath';
9298: { Creates complex vector V[0..Ub] }
9299: Proc DimBoolVector(var V : TBoolVector; Ub : Int);external 'dmath';
9300: { Creates boolean vector V[0..Ub] }
9301: Proc DimStrVector(var V : TStrVector; Ub : Int); external 'dmath';
9302: { Creates string vector V[0..Ub] }
9303: Proc DimMatrix(var A : TMatrix; Ub1, Ub2 : Int); external 'dmath';
9304: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9305: Proc DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Int); external 'dmath';
9306: { Creates Int matrix A[0..Ub1, 0..Ub2] }
9307: Proc DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Int);external 'dmath';
9308: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9309: Proc DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Int);external 'dmath';
9310: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9311: Proc DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Int); external 'dmath';
9312: { Creates string matrix A[0..Ub1, 0..Ub2] }
9313: { -----
9314:   Minimum, maximum, sign and exchange
9315:   ----- }

```

```

9316: Func FMin(X, Y : Float) : Float; external 'dmath';
9317: { Minimum of 2 reals }
9318: Func FMax(X, Y : Float) : Float; external 'dmath';
9319: { Maximum of 2 reals }
9320: Func IMin(X, Y : Int) : Int; external 'dmath';
9321: { Minimum of 2 Ints }
9322: Func IMax(X, Y : Int) : Int; external 'dmath';
9323: { Maximum of 2 Ints }
9324: Func Sgn(X : Float) : Int; external 'dmath';
9325: { Sign (returns 1 if X = 0) }
9326: Func Sgn0(X : Float) : Int; external 'dmath';
9327: { Sign (returns 0 if X = 0) }
9328: Func DSgn(A, B : Float) : Float; external 'dmath';
9329: { Sgn(B) * |A| }
9330: Proc FSwap(var X, Y : Float); external 'dmath';
9331: { Exchange 2 reals }
9332: Proc ISwap(var X, Y : Int); external 'dmath';
9333: { Exchange 2 Ints }
9334: { -----
9335:   Rounding functions
9336:   ----- }
9337: Func RoundN(X : Float; N : Int) : Float; external 'dmath';
9338: { Rounds X to N decimal places }
9339: Func Ceil(X : Float) : Int; external 'dmath';
9340: { Ceiling Func }
9341: Func Floor(X : Float) : Int; external 'dmath';
9342: { Floor Func }
9343: { -----
9344:   Logarithms, exponentials and power
9345:   ----- }
9346: Func Expo(X : Float) : Float; external 'dmath';
9347: { Exponential }
9348: Func Exp2(X : Float) : Float; external 'dmath';
9349: { 2^X }
9350: Func Exp10(X : Float) : Float; external 'dmath';
9351: { 10^X }
9352: Func Log(X : Float) : Float; external 'dmath';
9353: { Natural log }
9354: Func Log2(X : Float) : Float; external 'dmath';
9355: { Log, base 2 }
9356: Func Log10(X : Float) : Float; external 'dmath';
9357: { Decimal log }
9358: Func LogA(X, A : Float) : Float; external 'dmath';
9359: { Log, base A }
9360: Func IntPower(X : Float; N : Int) : Float; external 'dmath';
9361: { X^N }
9362: Func Power(X, Y : Float) : Float; external 'dmath';
9363: { X^Y, X >= 0 }
9364: { -----
9365:   Trigonometric functions
9366:   ----- }
9367: Func Pythag(X, Y : Float) : Float; external 'dmath';
9368: { Sqrt(X^2 + Y^2) }
9369: Func FixAngle(Theta : Float) : Float; external 'dmath';
9370: { Set Theta in -Pi..Pi }
9371: Func Tan(X : Float) : Float; external 'dmath';
9372: { Tangent }
9373: Func ArcSin(X : Float) : Float; external 'dmath';
9374: { Arc sinus }
9375: Func ArcCos(X : Float) : Float; external 'dmath';
9376: { Arc cosinus }
9377: Func ArcTan2(Y, X : Float) : Float; external 'dmath';
9378: { Angle (Ox, OM) with M(X,Y) }
9379: { -----
9380:   Hyperbolic functions
9381:   ----- }
9382: Func Sinh(X : Float) : Float; external 'dmath';
9383: { Hyperbolic sine }
9384: Func Cosh(X : Float) : Float; external 'dmath';
9385: { Hyperbolic cosine }
9386: Func Tanh(X : Float) : Float; external 'dmath';
9387: { Hyperbolic tangent }
9388: Func ArcSinh(X : Float) : Float; external 'dmath';
9389: { Inverse hyperbolic sine }
9390: Func ArcCosh(X : Float) : Float; external 'dmath';
9391: { Inverse hyperbolic cosine }
9392: Func ArcTanh(X : Float) : Float; external 'dmath';
9393: { Inverse hyperbolic tangent }
9394: Proc SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9395: { Sinh & Cosh }
9396: { -----
9397:   Gamma Func and related functions
9398:   ----- }
9399: Func Fact(N : Int) : Float; external 'dmath';
9400: { Factorial }
9401: Func SgnGamma(X : Float) : Int; external 'dmath';
9402: { Sign of Gamma Func }
9403: Func Gamma(X : Float) : Float; external 'dmath';
9404: { Gamma Func }

```

```

9405: Func LnGamma(X : Float) : Float; external 'dmath';
9406: { Logarithm of Gamma Func }
9407: Func Stirling(X : Float) : Float; external 'dmath';
9408: { Stirling's formula for the Gamma Func }
9409: Func StirLog(X : Float) : Float; external 'dmath';
9410: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9411: Func DiGamma(X : Float) : Float; external 'dmath';
9412: { Digamma Func }
9413: Func TriGamma(X : Float) : Float; external 'dmath';
9414: { Trigamma Func }
9415: Func IGamma(A, X : Float) : Float; external 'dmath';
9416: { Incomplete Gamma function}
9417: Func JGamma(A, X : Float) : Float; external 'dmath';
9418: { Complement of incomplete Gamma Func }
9419: Func InvGamma(A, P : Float) : Float; external 'dmath';
9420: { Inverse of incomplete Gamma Func }
9421: Func Erf(X : Float) : Float; external 'dmath';
9422: { Error Func }
9423: Func Erfc(X : Float) : Float; external 'dmath';
9424: { Complement of error Func }
9425: { -----
9426:   Beta Func and related functions
9427: ----- }
9428: Func Beta(X, Y : Float) : Float; external 'dmath';
9429: { Beta Func }
9430: Func IBeta(A, B, X : Float) : Float; external 'dmath';
9431: { Incomplete Beta Func }
9432: Func InvBeta(A, B, Y : Float) : Float; external 'dmath';
9433: { Inverse of incomplete Beta Func }
9434: { -----
9435:   Lambert's function
9436: ----- }
9437: Func LambertW(X : Float; UBranch, Offset :Bool) : Float; external 'dmath';
9438: -----
9439:   Binomial distribution
9440: ----- }
9441: Func Binomial(N, K : Int) : Float; external 'dmath';
9442: { Binomial coefficient C(N,K) }
9443: Func PBinom(N : Int; P : Float; K : Int) : Float; external 'dmath';
9444: { Probability of binomial distribution }
9445: Func FBinom(N : Int; P : Float; K : Int) : Float; external 'dmath';
9446: { Cumulative probability for binomial distrib. }
9447: { -----
9448:   Poisson distribution
9449: ----- }
9450: Func PPoisson(Mu : Float; K : Int) : Float; external 'dmath';
9451: { Probability of Poisson distribution }
9452: Func FPoisson(Mu : Float; K : Int) : Float; external 'dmath';
9453: { Cumulative probability for Poisson distrib. }
9454: { -----
9455:   Exponential distribution
9456: ----- }
9457: Func DExpo(A, X : Float) : Float; external 'dmath';
9458: { Density of exponential distribution with parameter A }
9459: Func FExpo(A, X : Float) : Float; external 'dmath';
9460: { Cumulative probability Func for exponential dist. with parameter A }
9461: { -----
9462:   Standard normal distribution
9463: ----- }
9464: Func DNorm(X : Float) : Float; external 'dmath';
9465: { Density of standard normal distribution }
9466: Func FNorm(X : Float) : Float; external 'dmath';
9467: { Cumulative probability for standard normal distrib. }
9468: Func PNorm(X : Float) : Float; external 'dmath';
9469: { Prob(|U| > X) for standard normal distrib. }
9470: Func InvNorm(P : Float) : Float; external 'dmath';
9471: { Inverse of standard normal distribution }
9472: { -----
9473:   Student's distribution
9474: ----- }
9475: Func DStudent(Nu : Int; X : Float) : Float; external 'dmath';
9476: { Density of Student distribution with Nu d.o.f. }
9477: Func FStudent(Nu : Int; X : Float) : Float; external 'dmath';
9478: { Cumulative probability for Student distrib. with Nu d.o.f. }
9479: Func PStudent(Nu : Int; X : Float) : Float; external 'dmath';
9480: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9481: Func InvStudent(Nu : Int; P : Float) : Float; external 'dmath';
9482: { Inverse of Student's t-distribution Func }
9483: { -----
9484:   Khi-2 distribution
9485: ----- }
9486: Func DKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9487: { Density of Khi-2 distribution with Nu d.o.f. }
9488: Func FKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9489: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9490: Func PKhi2(Nu : Int; X : Float) : Float; external 'dmath';
9491: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9492: Func InvKhi2(Nu : Int; P : Float) : Float; external 'dmath';
9493: { Inverse of Khi-2 distribution Func }

```



```

9494: { -----
9495:   Fisher-Snedecor distribution
9496: ----- }
9497: Func DSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9498: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9499: Func FSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9500: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9501: Func PSnedecor(Nu1, Nu2 : Int; X : Float) : Float; external 'dmath';
9502: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9503: Func InvSnedecor(Nu1, Nu2 : Int; P : Float) : Float; external 'dmath';
9504: { Inverse of Snedecor's F-distribution Func }
9505: { -----
9506:   Beta distribution
9507: ----- }
9508: Func DBeta(A, B, X : Float) : Float; external 'dmath';
9509: { Density of Beta distribution with parameters A and B }
9510: Func FBeta(A, B, X : Float) : Float; external 'dmath';
9511: { Cumulative probability for Beta distrib. with param. A and B }
9512: { -----
9513:   Gamma distribution
9514: ----- }
9515: Func DGamma(A, B, X : Float) : Float; external 'dmath';
9516: { Density of Gamma distribution with parameters A and B }
9517: Func FGamma(A, B, X : Float) : Float; external 'dmath';
9518: { Cumulative probability for Gamma distrib. with param. A and B }
9519: { -----
9520:   Expression evaluation
9521: ----- }
9522: Func InitEval : Int; external 'dmath';
9523: { Initializes built-in functions and returns their number }
9524: Func Eval(ExpressionString :Str) : Float; external 'dmath';
9525: { Evaluates an expression at run-time }
9526: Proc SetVariable(VarName : Char; Value : Float); external 'dmath';
9527: { Assigns a value to a variable }
9528: Proc SetFunction(FuncName :Str; Wrapper : TWrapper); external 'dmath';
9529: { Adds a Func to the parser }
9530: { -----
9531:   Matrices and linear equations
9532: ----- }
9533: Proc GaussJordan(A : TMatrix;
9534:   Lb, Ub1, Ub2 : Int;
9535:   var Det : Float); external 'dmath';
9536: { Transforms a matrix according to the Gauss-Jordan method }
9537: Proc LinEq(A : TMatrix;
9538:   B : TVector;
9539:   Lb, Ub : Int;
9540:   var Det : Float); external 'dmath';
9541: { Solves a linear system according to the Gauss-Jordan method }
9542: Proc Cholesky(A, L : TMatrix; Lb, Ub : Int); external 'dmath';
9543: { Cholesky factorization of a positive definite symmetric matrix }
9544: Proc LU_Decom(A : TMatrix; Lb, Ub : Int); external 'dmath';
9545: { LU decomposition }
9546: Proc LU_Solve(A : TMatrix;
9547:   B : TVector;
9548:   Lb, Ub : Int;
9549:   X : TVector); external 'dmath';
9550: { Solution of linear system from LU decomposition }
9551: Proc QR_Decom(A : TMatrix;
9552:   Lb, Ub1, Ub2 : Int;
9553:   R : TMatrix); external 'dmath';
9554: { QR decomposition }
9555: Proc QR_Solve(Q, R : TMatrix;
9556:   B : TVector;
9557:   Lb, Ub1, Ub2 : Int;
9558:   X : TVector); external 'dmath';
9559: { Solution of linear system from QR decomposition }
9560: Proc SV_Decom(A : TMatrix;
9561:   Lb, Ub1, Ub2 : Int;
9562:   S : TVector;
9563:   V : TMatrix); external 'dmath';
9564: { Singular value decomposition }
9565: Proc SV_SetZero(S : TVector;
9566:   Lb, Ub : Int;
9567:   Tol : Float); external 'dmath';
9568: { Set lowest singular values to zero }
9569: Proc SV_Solve(U : TMatrix;
9570:   S : TVector;
9571:   V : TMatrix;
9572:   B : TVector;
9573:   Lb, Ub1, Ub2 : Int;
9574:   X : TVector); external 'dmath';
9575: { Solution of linear system from SVD }
9576: Proc SV_Approx(U : TMatrix;
9577:   S : TVector;
9578:   V : TMatrix;
9579:   Lb, Ub1, Ub2 : Int;
9580:   A : TMatrix); external 'dmath';
9581: { Matrix approximation from SVD }
9582: Proc EigenVals(A : TMatrix;

```

```

9583:         Lb, Ub : Int;
9584:         Lambda : TCompVector); external 'dmath';
9585: { Eigenvalues of a general square matrix }
9586: Proc EigenVect(A : TMatrix;
9587:         Lb, Ub : Int;
9588:         Lambda : TCompVector;
9589:         V : TMatrix); external 'dmath';
9590: { Eigenvalues and eigenvectors of a general square matrix }
9591: Proc EigenSym(A : TMatrix;
9592:         Lb, Ub : Int;
9593:         Lambda : TVector;
9594:         V : TMatrix); external 'dmath';
9595: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9596: Proc Jacobi(A : TMatrix;
9597:         Lb, Ub, MaxIter : Int;
9598:         Tol : Float;
9599:         Lambda : TVector;
9600:         V : TMatrix); external 'dmath';
9601: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9602: { -----
9603:   Optimization
9604:   ----- }
9605: Proc MinBrack(Func : TFunc;
9606:         var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9607: { Brackets a minimum of a Func }
9608: Proc GoldSearch(Func : TFunc;
9609:         A, B : Float;
9610:         MaxIter : Int;
9611:         Tol : Float;
9612:         var Xmin, Ymin : Float); external 'dmath';
9613: { Minimization of a Func of one variable (golden search) }
9614: Proc LinMin(Func : TFuncNVar;
9615:         X, DeltaX : TVector;
9616:         Lb, Ub : Int;
9617:         var R : Float;
9618:         MaxIter : Int;
9619:         Tol : Float;
9620:         var F_min : Float); external 'dmath';
9621: { Minimization of a Func of several variables along a line }
9622: Proc Newton(Func : TFuncNVar;
9623:         HessGrad : THessGrad;
9624:         X : TVector;
9625:         Lb, Ub : Int;
9626:         MaxIter : Int;
9627:         Tol : Float;
9628:         var F_min : Float;
9629:         G : TVector;
9630:         H_inv : TMatrix;
9631:         var Det : Float); external 'dmath';
9632: { Minimization of a Func of several variables (Newton's method) }
9633: Proc SaveNewton(FileName : Str); external 'dmath';
9634: { Save Newton iterations in a file }
9635: Proc Marquardt(Func : TFuncNVar;
9636:         HessGrad : THessGrad;
9637:         X : TVector;
9638:         Lb, Ub : Int;
9639:         MaxIter : Int;
9640:         Tol : Float;
9641:         var F_min : Float;
9642:         G : TVector;
9643:         H_inv : TMatrix;
9644:         var Det : Float); external 'dmath';
9645: { Minimization of a Func of several variables (Marquardt's method) }
9646: Proc SaveMarquardt(FileName : Str); external 'dmath';
9647: { Save Marquardt iterations in a file }
9648: Proc BFGS(Func : TFuncNVar;
9649:         Gradient : TGradient;
9650:         X : TVector;
9651:         Lb, Ub : Int;
9652:         MaxIter : Int;
9653:         Tol : Float;
9654:         var F_min : Float;
9655:         G : TVector;
9656:         H_inv : TMatrix); external 'dmath';
9657: { Minimization of a Func of several variables (BFGS method) }
9658: Proc SaveBFGS(FileName : Str); external 'dmath';
9659: { Save BFGS iterations in a file }
9660: Proc Simplex(Func : TFuncNVar;
9661:         X : TVector;
9662:         Lb, Ub : Int;
9663:         MaxIter : Int;
9664:         Tol : Float;
9665:         var F_min : Float); external 'dmath';
9666: { Minimization of a Func of several variables (Simplex) }
9667: Proc SaveSimplex(FileName : Str); external 'dmath';
9668: { Save Simplex iterations in a file }
9669: { -----
9670:   Nonlinear equations
9671:   ----- }

```

```

9672: Proc RootBrack(Func          : TFunc;
9673:                 var X, Y, FX, FY : Float); external 'dmath';
9674: { Brackets a root of Func Func between X and Y }
9675: Proc Bisect(Func          : TFunc;
9676:             var X, Y : Float;
9677:             MaxIter  : Int;
9678:             Tol      : Float;
9679:             var F     : Float); external 'dmath';
9680: { Bisection method }
9681: Proc Secant(Func          : TFunc;
9682:             var X, Y : Float;
9683:             MaxIter  : Int;
9684:             Tol      : Float;
9685:             var F     : Float); external 'dmath';
9686: { Secant method }
9687: Proc NewtEq(Func, Deriv : TFunc;
9688:             var X      : Float;
9689:             MaxIter    : Int;
9690:             Tol        : Float;
9691:             var F       : Float); external 'dmath';
9692: { Newton-Raphson method for a single nonlinear equation }
9693: Proc NewtEqs(Equations : TEquations;
9694:             Jacobian   : TJacobian;
9695:             X, F       : TVector;
9696:             Lb, Ub     : Int;
9697:             MaxIter    : Int;
9698:             Tol        : Float); external 'dmath';
9699: { Newton-Raphson method for a system of nonlinear equations }
9700: Proc Broyden(Equations : TEquations;
9701:             X, F       : TVector;
9702:             Lb, Ub     : Int;
9703:             MaxIter    : Int;
9704:             Tol        : Float); external 'dmath';
9705: { Broyden's method for a system of nonlinear equations }
9706: { -----
9707:   Polynomials and rational fractions
9708:   ----- }
9709: Func Poly(X      : Float;
9710:          Coef : TVector;
9711:          Deg  : Int) : Float; external 'dmath';
9712: { Evaluates a polynomial }
9713: Func RFrac(X      : Float;
9714:          Coef : TVector;
9715:          Deg1, Deg2 : Int) : Float; external 'dmath';
9716: { Evaluates a rational fraction }
9717: Func RootPol1(A, B : Float;
9718:             var X : Float) : Int; external 'dmath';
9719: { Solves the linear equation A + B * X = 0 }
9720: Func RootPol2(Coef : TVector;
9721:             Z      : TCompVector) : Int; external 'dmath';
9722: { Solves a quadratic equation }
9723: Func RootPol3(Coef : TVector;
9724:             Z      : TCompVector) : Int; external 'dmath';
9725: { Solves a cubic equation }
9726: Func RootPol4(Coef : TVector;
9727:             Z      : TCompVector) : Int; external 'dmath';
9728: { Solves a quartic equation }
9729: Func RootPol(Coef : TVector;
9730:             Deg  : Int;
9731:             Z      : TCompVector) : Int; external 'dmath';
9732: { Solves a polynomial equation }
9733: Func SetRealRoots(Deg : Int;
9734:             Z      : TCompVector;
9735:             Tol : Float) : Int; external 'dmath';
9736: { Set the imaginary part of a root to zero }
9737: Proc SortRoots(Deg : Int;
9738:             Z      : TCompVector); external 'dmath';
9739: { Sorts the roots of a polynomial }
9740: { -----
9741:   Numerical integration and differential equations
9742:   ----- }
9743: Func TrapInt(X, Y : TVector; N : Int) : Float; external 'dmath';
9744: { Integration by trapezoidal rule }
9745: Func GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9746: { Integral from A to B }
9747: Func GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9748: { Integral from 0 to B }
9749: Func Convolve(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9750: { Convolution product at time T }
9751: Proc ConvTrap(Func1, Func2: TFunc; T, Y: TVector; N: Int); external 'dmath';
9752: { Convolution by trapezoidal rule }
9753: Proc RKF45(F          : TDiffEqs;
9754:           Negn      : Int;
9755:           Y, Yp     : TVector;
9756:           var T      : Float;
9757:           Tout, RelErr, AbsErr : Float;
9758:           var Flag    : Int); external 'dmath';
9759: { Integration of a system of differential equations }
9760: { -----

```

```

9761:   Fast Fourier Transform
9762:   ----- }
9763: Proc FFT(NumSamples      : Int;
9764:         InArray, OutArray : TCompVector); external 'dmath';
9765: { Fast Fourier Transform }
9766: Proc IFFT(NumSamples      : Int;
9767:         InArray, OutArray : TCompVector); external 'dmath';
9768: { Inverse Fast Fourier Transform }
9769: Proc FFT_Int(NumSamples      : Int;
9770:             RealIn, ImagIn   : TIntVector;
9771:             OutArray         : TCompVector); external 'dmath';
9772: { Fast Fourier Transform for Int data }
9773: Proc FFT_Int_Cleanup; external 'dmath';
9774: { Clear memory after a call to FFT_Int }
9775: Proc CalcFrequency(NumSamples,
9776:                  FrequencyIndex : Int;
9777:                  InArray         : TCompVector;
9778:                  var FFT         : Complex); external 'dmath';
9779: { Direct computation of Fourier transform }
9780: { ----- }
9781:   Random numbers
9782:   ----- }
9783: Proc SetRNG(RNG : RNG_Type); external 'dmath';
9784: { Select generator }
9785: Proc InitGen(Seed : RNG_IntType); external 'dmath';
9786: { Initialize generator }
9787: Func IRanGen : RNG_IntType; external 'dmath';
9788: { 32-bit random Int in [-2^31 .. 2^31 - 1] }
9789: Func IRanGen31 : RNG_IntType; external 'dmath';
9790: { 31-bit random Int in [0 .. 2^31 - 1] }
9791: Func RanGen1 : Float; external 'dmath';
9792: { 32-bit random real in [0,1] }
9793: Func RanGen2 : Float; external 'dmath';
9794: { 32-bit random real in [0,1] }
9795: Func RanGen3 : Float; external 'dmath';
9796: { 32-bit random real in (0,1) }
9797: Func RanGen53 : Float; external 'dmath';
9798: { 53-bit random real in [0,1] }
9799: Proc InitMWC(Seed : RNG_IntType); external 'dmath';
9800: { Initializes the 'Multiply with carry' random number generator }
9801: Func IRanMWC : RNG_IntType; external 'dmath';
9802: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9803: Proc InitMT(Seed : RNG_IntType); external 'dmath';
9804: { Initializes Mersenne Twister generator with a seed }
9805: Proc InitMTbyArray(InitKey : array of RNG_LongType;
9806:                  KeyLength : Word); external 'dmath';
9807: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9808: Func IRanMT : RNG_IntType; external 'dmath';
9809: { Random Int from MT generator }
9810: Proc InitUVAGbyString(KeyPhrase : Str); external 'dmath';
9811: { Initializes the UVAG generator with a string }
9812: Proc InitUVAG(Seed : RNG_IntType); external 'dmath';
9813: { Initializes the UVAG generator with an Int }
9814: Func IRanUVAG : RNG_IntType; external 'dmath';
9815: { Random Int from UVAG generator }
9816: Func RanGaussStd : Float; external 'dmath';
9817: { Random number from standard normal distribution }
9818: Func RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9819: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9820: Proc RanMult(M : TVector; L : TMatrix;
9821:             Lb, Ub : Int;
9822:             X       : TVector); external 'dmath';
9823: { Random vector from multinormal distribution (correlated) }
9824: Proc RanMultIndep(M, S : TVector;
9825:                 Lb, Ub : Int;
9826:                 X       : TVector); external 'dmath';
9827: { Random vector from multinormal distribution (uncorrelated) }
9828: Proc InitMHPParams(NCycles, MaxSim, SavedSim : Int); external 'dmath';
9829: { Initializes Metropolis-Hastings parameters }
9830: Proc GetMHPParams(var NCycles, MaxSim, SavedSim: Int); external 'dmath';
9831: { Returns Metropolis-Hastings parameters }
9832: Proc Hastings(Func : TFuncNVar;
9833:             T       : Float;
9834:             X       : TVector;
9835:             V       : TMatrix;
9836:             Lb, Ub  : Int;
9837:             Xmat    : TMatrix;
9838:             X_min   : TVector;
9839:             var F_min : Float); external 'dmath';
9840: { Simulation of a probability density Func by Metropolis-Hastings }
9841: Proc InitSAParams(NT, NS, NCycles : Int; RT : Float); external 'dmath';
9842: { Initializes Simulated Annealing parameters }
9843: Proc SA_CreateLogFile(FileName : Str); external 'dmath';
9844: { Initializes log file }
9845: Proc SimAnn(Func : TFuncNVar;
9846:            X, Xmin, Xmax : TVector;
9847:            Lb, Ub       : Int;
9848:            var F_min    : Float); external 'dmath';
9849: { Minimization of a Func of several var. by simulated annealing }

```

```

9850: Proc InitGAParams(NP, NG : Int; SR, MR, HR : Float); external 'dmath';
9851: { Initializes Genetic Algorithm parameters }
9852: Proc GA_CreateLogFile(FileName :Str); external 'dmath';
9853: { Initializes log file }
9854: Proc GenAlg(Func          : TFuncNVar;
9855:             X, Xmin, Xmax : TVector;
9856:             Lb, Ub       : Int;
9857:             var F_min     : Float); external 'dmath';
9858: { Minimization of a Func of several var. by genetic algorithm }
9859: { -----
9860:   Statistics
9861: ----- }
9862: Func Mean(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
9863: { Mean of sample X }
9864: Func Min(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
9865: { Minimum of sample X }
9866: Func Max(X : TVector; Lb, Ub : Int) : Float; external 'dmath';
9867: { Maximum of sample X }
9868: Func Median(X : TVector; Lb, Ub : Int; Sorted :Bool) : Float; external 'dmath';
9869: { Median of sample X }
9870: Func StDev(X : TVector; Lb, Ub : Int; M : Float) : Float; external 'dmath';
9871: { Standard deviation estimated from sample X }
9872: Func StDevP(X : TVector; Lb, Ub : Int; M : Float) : Float; external 'dmath';
9873: { Standard deviation of population }
9874: Func Correl(X, Y : TVector; Lb, Ub : Int) : Float; external 'dmath';
9875: { Correlation coefficient }
9876: Func Skewness(X : TVector; Lb, Ub : Int; M,Sigma: Float): Float; external 'dmath';
9877: { Skewness of sample X }
9878: Func Kurtosis(X : TVector; Lb, Ub : Int; M,Sigma: Float): Float; external 'dmath';
9879: { Kurtosis of sample X }
9880: Proc QSort(X : TVector; Lb, Ub : Int); external 'dmath';
9881: { Quick sort (ascending order) }
9882: Proc DQSort(X : TVector; Lb, Ub : Int); external 'dmath';
9883: { Quick sort (descending order) }
9884: Proc Interval(X1, X2          : Float;
9885:              MinDiv, MaxDiv   : Int;
9886:              var Min, Max, Step : Float); external 'dmath';
9887: { Determines an interval for a set of values }
9888: Proc AutoScale(X : TVector; Lb, Ub : Int; Scale : TScale;
9889:               var XMin, XMax, XStep : Float); external 'dmath';
9890: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9891: Proc StudIndep(N1, N2          : Int;
9892:               M1, M2, S1, S2 : Float;
9893:               var T          : Float;
9894:               var DoF        : Int); external 'dmath';
9895: { Student t-test for independent samples }
9896: Proc StudPaired(X, Y          : TVector;
9897:               Lb, Ub       : Int;
9898:               var T        : Float;
9899:               var DoF      : Int); external 'dmath';
9900: { Student t-test for paired samples }
9901: Proc AnOVal (Ns          : Int;
9902:             N            : TIntVector;
9903:             M, S         : TVector;
9904:             var V_f, V_r, F : Float;
9905:             var DoF_f, DoF_r : Int); external 'dmath';
9906: { One-way analysis of variance }
9907: Proc AnOVA2 (NA, NB, Nobs : Int;
9908:            M, S          : TMatrix;
9909:            V, F          : TVector;
9910:            DoF           : TIntVector); external 'dmath';
9911: { Two-way analysis of variance }
9912: Proc Snedecor(N1, N2          : Int;
9913:             S1, S2          : Float;
9914:             var F           : Float;
9915:             var DoF1, DoF2 : Int); external 'dmath';
9916: { Snedecor's F-test (comparison of two variances) }
9917: Proc Bartlett(Ns          : Int;
9918:             N            : TIntVector;
9919:             S            : TVector;
9920:             var Khi2      : Float;
9921:             var DoF       : Int); external 'dmath';
9922: { Bartlett's test (comparison of several variances) }
9923: Proc Mann_Whitney(N1, N2          : Int;
9924:                 X1, X2          : TVector;
9925:                 var U, Eps      : Float); external 'dmath';
9926: { Mann-Whitney test }
9927: Proc Wilcoxon(X, Y          : TVector;
9928:             Lb, Ub       : Int;
9929:             var Ndiff    : Int;
9930:             var T, Eps   : Float); external 'dmath';
9931: { Wilcoxon test }
9932: Proc Kruskal_Wallis(Ns          : Int;
9933:                   N            : TIntVector;
9934:                   X            : TMatrix;
9935:                   var H        : Float;
9936:                   var DoF      : Int); external 'dmath';
9937: { Kruskal-Wallis test }
9938: Proc Khi2_Conform(N_cls      : Int;

```



```

9939:             N_estim : Int;
9940:             Obs      : TIntVector;
9941:             Calc      : TVector;
9942:             var Khi2  : Float;
9943:             var DoF   : Int); external 'dmath';
9944: { Khi-2 test for conformity }
9945: Proc Khi2_Indep(N_lin : Int;
9946:               N_col   : Int;
9947:               Obs      : TIntMatrix;
9948:               var Khi2 : Float;
9949:               var DoF   : Int); external 'dmath';
9950: { Khi-2 test for independence }
9951: Proc Woolf_Conform(N_cls : Int;
9952:                  N_estim : Int;
9953:                  Obs      : TIntVector;
9954:                  Calc      : TVector;
9955:                  var G      : Float;
9956:                  var DoF    : Int); external 'dmath';
9957: { Woolf's test for conformity }
9958: Proc Woolf_Indep(N_lin : Int;
9959:                 N_col   : Int;
9960:                 Obs      : TIntMatrix;
9961:                 var G      : Float;
9962:                 var DoF    : Int); external 'dmath';
9963: { Woolf's test for independence }
9964: Proc DimStatClassVector(var C : TStatClassVector;
9965:                        Ub      : Int); external 'dmath';
9966: { Allocates an array of statistical classes: C[0..Ub] }
9967: Proc Distrib(X : TVector;
9968:             Lb, Ub : Int;
9969:             A, B, H : Float;
9970:             C : TStatClassVector); external 'dmath';
9971: { Distributes an array X[Lb..Ub] into statistical classes }
9972: { ----- }
9973: Linear / polynomial regression
9974: { ----- }
9975: Proc LinFit(X, Y : TVector;
9976:            Lb, Ub : Int;
9977:            B : TVector;
9978:            V : TMatrix); external 'dmath';
9979: { Linear regression :  $Y = B(0) + B(1) * X$  }
9980: Proc WLinFit(X, Y, S : TVector;
9981:             Lb, Ub : Int;
9982:             B : TVector;
9983:             V : TMatrix); external 'dmath';
9984: { Weighted linear regression :  $Y = B(0) + B(1) * X$  }
9985: Proc SVDLinFit(X, Y : TVector;
9986:              Lb, Ub : Int;
9987:              SVDTol : Float;
9988:              B : TVector;
9989:              V : TMatrix); external 'dmath';
9990: { Unweighted linear regression by singular value decomposition }
9991: Proc WSVDLinFit(X, Y, S : TVector;
9992:               Lb, Ub : Int;
9993:               SVDTol : Float;
9994:               B : TVector;
9995:               V : TMatrix); external 'dmath';
9996: { Weighted linear regression by singular value decomposition }
9997: Proc MulFit(X : TMatrix;
9998:            Y : TVector;
9999:            Lb, Ub, Nvar : Int;
10000:           ConstTerm : Bool;
10001:           B : TVector;
10002:           V : TMatrix); external 'dmath';
10003: { Multiple linear regression by Gauss-Jordan method }
10004: Proc WMulFit(X : TMatrix;
10005:             Y, S : TVector;
10006:             Lb, Ub, Nvar : Int;
10007:             ConstTerm : Bool;
10008:             B : TVector;
10009:             V : TMatrix); external 'dmath';
10010: { Weighted multiple linear regression by Gauss-Jordan method }
10011: Proc SVDFit(X : TMatrix;
10012:            Y : TVector;
10013:            Lb, Ub, Nvar : Int;
10014:            ConstTerm : Bool;
10015:            SVDTol : Float;
10016:            B : TVector;
10017:            V : TMatrix); external 'dmath';
10018: { Multiple linear regression by singular value decomposition }
10019: Proc WSVDFit(X : TMatrix;
10020:             Y, S : TVector;
10021:             Lb, Ub, Nvar : Int;
10022:             ConstTerm : Bool;
10023:             SVDTol : Float;
10024:             B : TVector;
10025:             V : TMatrix); external 'dmath';
10026: { Weighted multiple linear regression by singular value decomposition }
10027: Proc PolFit(X, Y : TVector;

```

```

10028:          Lb, Ub, Deg : Int;
10029:          B           : TVector;
10030:          V           : TMatrix); external 'dmath';
10031: { Polynomial regression by Gauss-Jordan method }
10032: Proc WPolFit(X, Y, S : TVector;
10033:          Lb, Ub, Deg : Int;
10034:          B           : TVector;
10035:          V           : TMatrix); external 'dmath';
10036: { Weighted polynomial regression by Gauss-Jordan method }
10037: Proc SVDPolFit(X, Y : TVector;
10038:          Lb, Ub, Deg : Int;
10039:          SVDTol      : Float;
10040:          B           : TVector;
10041:          V           : TMatrix); external 'dmath';
10042: { Unweighted polynomial regression by singular value decomposition }
10043: Proc WSVDPolFit(X, Y, S : TVector;
10044:          Lb, Ub, Deg : Int;
10045:          SVDTol      : Float;
10046:          B           : TVector;
10047:          V           : TMatrix); external 'dmath';
10048: { Weighted polynomial regression by singular value decomposition }
10049: Proc RegTest(Y, Ycalc : TVector;
10050:          LbY, UbY : Int;
10051:          V         : TMatrix;
10052:          LbV, UbV : Int;
10053:          var Test : TRegTest); external 'dmath';
10054: { Test of unweighted regression }
10055: Proc WRegTest(Y, Ycalc, S : TVector;
10056:          LbY, UbY : Int;
10057:          V         : TMatrix;
10058:          LbV, UbV : Int;
10059:          var Test : TRegTest); external 'dmath';
10060: { Test of weighted regression }
10061: { -----
10062:   Nonlinear regression
10063:   ----- }
10064: Proc SetOptAlgo(Algo : TOptAlgo); external 'dmath';
10065: { Sets the optimization algorithm for nonlinear regression }
10066: Func GetOptAlgo : TOptAlgo; external 'dmath';
10067: { Returns the optimization algorithm }
10068: Proc SetMaxParam(N : Byte); external 'dmath';
10069: { Sets the maximum number of regression parameters for nonlinear regression }
10070: Func GetMaxParam : Byte; external 'dmath';
10071: { Returns the maximum number of regression parameters for nonlinear regression }
10072: Proc SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
10073: { Sets the bounds on the I-th regression parameter }
10074: Proc GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
10075: { Returns the bounds on the I-th regression parameter }
10076: Proc NLFit(RegFunc : TRegFunc;
10077:          DerivProc : TDerivProc;
10078:          X, Y      : TVector;
10079:          Lb, Ub    : Int;
10080:          MaxIter   : Int;
10081:          Tol       : Float;
10082:          B         : TVector;
10083:          FirstPar,
10084:          LastPar   : Int;
10085:          V         : TMatrix); external 'dmath';
10086: { Unweighted nonlinear regression }
10087: Proc WNLFit(RegFunc : TRegFunc;
10088:          DerivProc : TDerivProc;
10089:          X, Y, S    : TVector;
10090:          Lb, Ub     : Int;
10091:          MaxIter    : Int;
10092:          Tol        : Float;
10093:          B          : TVector;
10094:          FirstPar, LastPar : Int;
10095:          V          : TMatrix); external 'dmath';
10096: { Weighted nonlinear regression }
10097: Proc SetMCFile(FileName :Str); external 'dmath';
10098: { Set file for saving MCMC simulations }
10099: Proc SimFit(RegFunc : TRegFunc;
10100:          X, Y      : TVector;
10101:          Lb, Ub    : Int;
10102:          B         : TVector;
10103:          FirstPar, LastPar : Int;
10104:          V         : TMatrix); external 'dmath';
10105: { Simulation of unweighted nonlinear regression by MCMC }
10106: Proc WSimFit(RegFunc : TRegFunc;
10107:          X, Y, S    : TVector;
10108:          Lb, Ub     : Int;
10109:          B          : TVector;
10110:          FirstPar, LastPar : Int;
10111:          V          : TMatrix); external 'dmath';
10112: { Simulation of weighted nonlinear regression by MCMC }
10113: { -----
10114:   Nonlinear regression models
10115:   ----- }
10116: Proc FracFit(X, Y : TVector;

```

```

10117:          Lb, Ub      : Int;
10118:          Deg1, Deg2  : Int;
10119:          ConstTerm   : Bool;
10120:          MaxIter      : Int;
10121:          Tol          : Float;
10122:          B            : TVector;
10123:          V            : TMatrix; external 'dmath';
10124: { Unweighted fit of rational fraction }
10125: Proc WFracFit(X, Y, S : TVector;
10126:          Lb, Ub      : Int;
10127:          Deg1, Deg2  : Int;
10128:          ConstTerm   : Bool;
10129:          MaxIter      : Int;
10130:          Tol          : Float;
10131:          B            : TVector;
10132:          V            : TMatrix); external 'dmath';
10133: { Weighted fit of rational fraction }
10134:
10135: Func FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10136: { Returns the value of the rational fraction at point X }
10137: Proc ExpFit(X, Y : TVector;
10138:          Lb, Ub, Nexpt : Int;
10139:          ConstTerm     : Bool;
10140:          MaxIter        : Int;
10141:          Tol            : Float;
10142:          B              : TVector;
10143:          V              : TMatrix); external 'dmath';
10144: { Unweighted fit of sum of exponentials }
10145: Proc WExpFit(X, Y, S : TVector;
10146:          Lb, Ub, Nexpt : Int;
10147:          ConstTerm     : Bool;
10148:          MaxIter        : Int;
10149:          Tol            : Float;
10150:          B              : TVector;
10151:          V              : TMatrix); external 'dmath';
10152: { Weighted fit of sum of exponentials }
10153: Func ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10154: { Returns the value of the regression Func at point X }
10155: Proc IncExpFit(X, Y : TVector;
10156:          Lb, Ub      : Int;
10157:          ConstTerm   : Bool;
10158:          MaxIter      : Int;
10159:          Tol          : Float;
10160:          B            : TVector;
10161:          V            : TMatrix); external 'dmath';
10162: { Unweighted fit of model of increasing exponential }
10163: Proc WIncExpFit(X, Y, S : TVector;
10164:          Lb, Ub      : Int;
10165:          ConstTerm   : Bool;
10166:          MaxIter      : Int;
10167:          Tol          : Float;
10168:          B            : TVector;
10169:          V            : TMatrix); external 'dmath';
10170: { Weighted fit of increasing exponential }
10171: Func IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10172: { Returns the value of the regression Func at point X }
10173: Proc ExpLinFit(X, Y : TVector;
10174:          Lb, Ub      : Int;
10175:          MaxIter      : Int;
10176:          Tol          : Float;
10177:          B            : TVector;
10178:          V            : TMatrix); external 'dmath';
10179: { Unweighted fit of the "exponential + linear" model }
10180: Proc WExpLinFit(X, Y, S : TVector;
10181:          Lb, Ub      : Int;
10182:          MaxIter      : Int;
10183:          Tol          : Float;
10184:          B            : TVector;
10185:          V            : TMatrix); external 'dmath';
10186: { Weighted fit of the "exponential + linear" model }
10187: Func ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10188: { Returns the value of the regression Func at point X }
10189: Proc MichFit(X, Y : TVector;
10190:          Lb, Ub      : Int;
10191:          MaxIter      : Int;
10192:          Tol          : Float;
10193:          B            : TVector;
10194:          V            : TMatrix); external 'dmath';
10195: { Unweighted fit of Michaelis equation }
10196: Proc WMichFit(X, Y, S : TVector;
10197:          Lb, Ub      : Int;
10198:          MaxIter      : Int;
10199:          Tol          : Float;
10200:          B            : TVector;
10201:          V            : TMatrix); external 'dmath';
10202: { Weighted fit of Michaelis equation }
10203: Func MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10204: { Returns the value of the Michaelis equation at point X }
10205: Proc MintFit(X, Y : TVector;

```

```

10206:          Lb, Ub : Int;
10207:          MintVar : TMintVar;
10208:          Fit_S0 :Bool;
10209:          MaxIter : Int;
10210:          Tol : Float;
10211:          B : TVector;
10212:          V : TMatrix); external 'dmath';
10213: { Unweighted fit of the integrated Michaelis equation }
10214: Proc WMintFit(X, Y, S : TVector;
10215:          Lb, Ub : Int;
10216:          MintVar : TMintVar;
10217:          Fit_S0 :Bool;
10218:          MaxIter : Int;
10219:          Tol : Float;
10220:          B : TVector;
10221:          V : TMatrix); external 'dmath';
10222: { Weighted fit of the integrated Michaelis equation }
10223: Func MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10224: { Returns the value of the integrated Michaelis equation at point X }
10225: Proc HillFit(X, Y : TVector;
10226:          Lb, Ub : Int;
10227:          MaxIter : Int;
10228:          Tol : Float;
10229:          B : TVector;
10230:          V : TMatrix); external 'dmath';
10231: { Unweighted fit of Hill equation }
10232: Proc WHillFit(X, Y, S : TVector;
10233:          Lb, Ub : Int;
10234:          MaxIter : Int;
10235:          Tol : Float;
10236:          B : TVector;
10237:          V : TMatrix); external 'dmath';
10238: { Weighted fit of Hill equation }
10239: Func HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10240: { Returns the value of the Hill equation at point X }
10241: Proc LogiFit(X, Y : TVector;
10242:          Lb, Ub : Int;
10243:          ConstTerm :Bool;
10244:          General :Bool;
10245:          MaxIter : Int;
10246:          Tol : Float;
10247:          B : TVector;
10248:          V : TMatrix); external 'dmath';
10249: { Unweighted fit of logistic Func }
10250: Proc WLogiFit(X, Y, S : TVector;
10251:          Lb, Ub : Int;
10252:          ConstTerm :Bool;
10253:          General :Bool;
10254:          MaxIter : Int;
10255:          Tol : Float;
10256:          B : TVector;
10257:          V : TMatrix); external 'dmath';
10258: { Weighted fit of logistic Func }
10259: Func LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10260: { Returns the value of the logistic Func at point X }
10261: Proc PKFit(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 the acid-base titration curve }
10268: Proc WPKFit(X, Y, S : TVector;
10269:          Lb, Ub : Int; MaxIter : Int;
10270:          Tol : Float;
10271:          B : TVector;
10272:          V : TMatrix); external 'dmath';
10273: { Weighted fit of the acid-base titration curve }
10274: Func PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10275: { Returns the value of the acid-base titration Func at point X }
10276: Proc PowFit(X, Y : TVector;
10277:          Lb, Ub : Int;
10278:          MaxIter : Int;
10279:          Tol : Float;
10280:          B : TVector;
10281:          V : TMatrix); external 'dmath';
10282: { Unweighted fit of power Func }
10283: Proc WPowFit(X, Y, S : TVector;
10284:          Lb, Ub : Int;
10285:          MaxIter : Int;
10286:          Tol : Float;
10287:          B : TVector;
10288:          V : TMatrix); external 'dmath';
10289: { Weighted fit of power Func }
10290:
10291: Func PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10292: { Returns the value of the power Func at point X }
10293: Proc GammaFit(X, Y : TVector;
10294:          Lb, Ub : Int;

```

```

10295:           MaxIter : Int;
10296:           Tol      : Float;
10297:           B        : TVector;
10298:           V        : TMatrix); external 'dmath';
10299: { Unweighted fit of gamma distribution Func }
10300: Proc WGammaFit(X, Y, S : TVector;
10301:               Lb, Ub : Int;
10302:               MaxIter : Int;
10303:               Tol      : Float;
10304:               B        : TVector;
10305:               V        : TMatrix); external 'dmath';
10306: { Weighted fit of gamma distribution Func }
10307: Func GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10308: { Returns the value of the gamma distribution Func at point X }
10309: { -----
10310:   Principal component analysis
10311:   ----- }
10312: Proc VecMean(X : TMatrix;
10313:             Lb, Ub, Nvar : Int;
10314:             M : TVector); external 'dmath';
10315: { Computes the mean vector M from matrix X }
10316: Proc VecSD(X : TMatrix;
10317:           Lb, Ub, Nvar : Int;
10318:           M, S : TVector); external 'dmath';
10319: { Computes the vector of standard deviations S from matrix X }
10320: Proc MatVarCov(X : TMatrix;
10321:               Lb, Ub, Nvar : Int;
10322:               M : TVector;
10323:               V : TMatrix); external 'dmath';
10324: { Computes the variance-covariance matrix V from matrix X }
10325: Proc MatCorrel(V : TMatrix;
10326:               Nvar : Int;
10327:               R : TMatrix); external 'dmath';
10328: { Computes the correlation matrix R from the var-cov matrix V }
10329: Proc PCA(R : TMatrix;
10330:         Nvar : Int;
10331:         Lambda : TVector;
10332:         C, Rc : TMatrix); external 'dmath';
10333: { Performs a principal component analysis of the correlation matrix R }
10334: Proc ScaleVar(X : TMatrix;
10335:              Lb, Ub, Nvar : Int;
10336:              M, S : TVector;
10337:              Z : TMatrix); external 'dmath';
10338: { Scales a set of variables by subtracting means and dividing by SD's }
10339: Proc PrinFac(Z : TMatrix;
10340:             Lb, Ub, Nvar : Int;
10341:             C, F : TMatrix); external 'dmath';
10342: { Computes principal factors }
10343: { -----
10344:   Strings
10345:   ----- }
10346: Func LTrim(S :Str) :Str; external 'dmath';
10347: { Removes leading blanks }
10348: Func RTrim(S :Str) :Str; external 'dmath';
10349: { Removes trailing blanks }
10350: Func Trim(S :Str) :Str; external 'dmath';
10351: { Removes leading and trailing blanks }
10352: Func StrChar(N : Byte; C : Char) :Str; external 'dmath';
10353: { Returns a string made of character C repeated N times }
10354: Func RFill(S :Str; L : Byte) :Str; external 'dmath';
10355: { Completes string S with trailing blanks for a total length L }
10356: Func LFill(S :Str; L : Byte) :Str; external 'dmath';
10357: { Completes string S with leading blanks for a total length L }
10358: Func CFill(S :Str; L : Byte) :Str; external 'dmath';
10359: { Centers string S on a total length L }
10360: Func Replace(S :Str; C1, C2 : Char) :Str; external 'dmath';
10361: { Replaces in string S all the occurrences of C1 by C2 }
10362: Func Extract(S:str; var Index:Byte; Delim:Char):str; external 'dmath';
10363: { Extracts a field from a string }
10364: Proc Parse(S:str;Delim:Char;Field:TStrVector;var N:Byte);external 'dmath';
10365: { Parses a string into its constitutive fields }
10366: Proc SetFormat(NumLength,MaxDec:Int;FloatPoint,NSZero:Bool); external 'dmath';
10367: { Sets the numeric format }
10368: Func FloatStr(X : Float) :Str; external 'dmath';
10369: { Converts a real to a string according to the numeric format }
10370: Func IntStr(N : LongInt) :Str; external 'dmath';
10371: { Converts an Int to a string }
10372: Func CompStr(Z : Complex) :Str; external 'dmath';
10373: { Converts a complex number to a string }
10374: {$IFDEF DELPHI}
10375: Func StrDec(S :Str) :Str; external 'dmath';
10376: { Set decimal separator to the symbol defined in SysUtils }
10377: Func IsNumeric(var S :Str; var X : Float) :Bool; external 'dmath';
10378: { Test if a string represents a number and returns it in X }
10379: Func ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10380: { Reads a floating point number from an Edit control }
10381: Proc WriteNumToFile(var F : Text; X : Float); external 'dmath';
10382: { Writes a floating point number in a text file }
10383: {$ENDIF}

```



```

10384: { -----
10385:   BGI / Delphi graphics
10386: ----- }
10387: Func InitGraphics
10388: {$IFDEF DELPHI}
10389: (Width, Height : Int) :Bool;
10390: {$ELSE}
10391: (Pilot, Mode : Int; BGIPath :Str) :Bool; {$ENDIF} external 'dmath';
10392: { Enters graphic mode }
10393: Proc SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10394:   X1, X2, Y1,Y2 : Int; GraphBorder:Boolean); external 'dmath';
10395: { Sets the graphic window }
10396: Proc SetOxScale(Scale : TScale;
10397:   OxMin, OxMax, OxStep : Float); external 'dmath';
10398: { Sets the scale on the Ox axis }
10399: Proc SetOyScale(Scale : TScale;
10400:   OyMin, OyMax, OyStep : Float); external 'dmath';
10401: { Sets the scale on the Oy axis }
10402: Proc GetOxScale(var Scale : TScale;
10403:   var OxMin, OxMax, OxStep : Float); external 'dmath';
10404: { Returns the scale on the Ox axis }
10405: Proc GetOyScale(var Scale : TScale;
10406:   var OyMin, OyMax, OyStep : Float); external 'dmath';
10407: { Returns the scale on the Oy axis }
10408: Proc SetGraphTitle(Title:str);external 'dmath';{Sets title for the graph }
10409: Proc SetOxTitle(Title :String);external'dmath';{ Sets title for the Ox axis }
10410: Proc SetOyTitle(Title:str);external'dmath';{Sets the title for Oy axis }
10411: Func GetGraphTitle:str;external 'dmath';{ Returns the title for the graph }
10412: Func GetOxTitle:str; external 'dmath'; { Returns the title for the Ox axis }
10413: Func GetOyTitle:str; external 'dmath'; { Returns the title for the Oy axis }
10414: {$IFDEF DELPHI}
10415: Proc SetTitleFont(FontIndex, Width, Height : Int); external 'dmath';
10416: { Sets the font for the main graph title }
10417: Proc SetOxFont(FontIndex, Width, Height : Int); external 'dmath';
10418: { Sets the font for the Ox axis (title and labels) }
10419: Proc SetOyFont(FontIndex, Width, Height : Int); external 'dmath';
10420: { Sets the font for the Oy axis (title and labels) }
10421: Proc SetLgdFont(FontIndex, Width, Height : Int); external 'dmath';
10422: { Sets the font for the legends }
10423: Proc SetClipping(Clip :Bool); external 'dmath';
10424: { Determines whether drawings are clipped at the current viewport
10425:   boundaries, according to the value of the Boolean parameter Clip }
10426: {$ENDIF}
10427: Proc PlotOxAxis({$IFDEF DELPHI}(Canvas : TCanvas) {$ENDIF} external 'dmath';
10428: { Plots the horizontal axis }
10429: Proc PlotOyAxis({$IFDEF DELPHI}(Canvas : TCanvas) {$ENDIF} external 'dmath';
10430: { Plots the vertical axis }
10431: Proc PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF}Grid:TGrid);external 'dmath';
10432: { Plots a grid on the graph }
10433: Proc WriteGraphTitle({$IFDEF DELPHI}(Canvas:TCanvas) {$ENDIF} external 'dmath';
10434: { Writes the title of the graph }
10435: Proc SetMaxCurv(NCurv : Byte); external 'dmath';
10436: { Sets the maximum number of curves and re-initializes their parameters }
10437: Proc SetPointParam
10438: {$IFDEF DELPHI}
10439: (CurvIndex, Symbol, Size : Int; Color : TColor);
10440: {$ELSE}
10441: (CurvIndex, Symbol, Size, Color : Int); {$ENDIF} external 'dmath';
10442: { Sets the point parameters for curve # CurvIndex }
10443: Proc SetLineParam
10444: {$IFDEF DELPHI}
10445: (CurvIndex : Int; Style : TPenStyle; Width : Int; Color : TColor);
10446: {$ELSE}
10447: (CurvIndex, Style, Width, Color : Int); {$ENDIF} external 'dmath';
10448: { Sets the line parameters for curve # CurvIndex }
10449: Proc SetCurvLegend(CurvIndex : Int; Legend :Str); external 'dmath';
10450: { Sets the legend for curve # CurvIndex }
10451: Proc SetCurvStep(CurvIndex, Step : Int); external 'dmath';
10452: { Sets the step for curve # CurvIndex }
10453: Func GetMaxCurv: Byte;external 'dmath';{ Returns the maximum number of curves }
10454: Proc GetPointParam
10455: {$IFDEF DELPHI}
10456: (CurvIndex : Int; var Symbol, Size : Int; var Color : TColor);
10457: {$ELSE}
10458: (CurvIndex : Int; var Symbol, Size, Color : Int); {$ENDIF} external 'dmath';
10459: { Returns the point parameters for curve # CurvIndex }
10460: Proc GetLineParam
10461: {$IFDEF DELPHI}
10462: (CurvIndex : Int; var Style : TPenStyle; var Width : Int; var Color : TColor);
10463: {$ELSE}
10464: (CurvIndex : Int; var Style, Width, Color : Int); {$ENDIF} external 'dmath';
10465: { Returns the line parameters for curve # CurvIndex }
10466: Func GetCurvLegend(CurvIndex : Int) :Str; external 'dmath';
10467: { Returns the legend for curve # CurvIndex }
10468: Func GetCurvStep(CurvIndex : Int) : Int; external 'dmath';
10469: { Returns the step for curve # CurvIndex }
10470: {$IFDEF DELPHI}
10471: Proc PlotPoint(Canvas : TCanvas;
10472:   X, Y : Float; CurvIndex : Int); external 'dmath';

```

```

10473: {$ELSE}
10474: Proc PlotPoint(Xp, Yp, CurvIndex : Int); external 'dmath';
10475: {$ENDIF}
10476: { Plots a point on the screen }
10477: Proc PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10478:               X, Y               : TVector;
10479:               Lb, Ub, CurvIndex   : Int); external 'dmath';
10480: { Plots a curve }
10481: Proc PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10482:                            X, Y, S       : TVector;
10483:                            Ns, Lb, Ub, CurvIndex : Int); external 'dmath';
10484: { Plots a curve with error bars }
10485: Proc PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10486:              Func               : TFunc;
10487:              Xmin, Xmax         : Float;
10488:              {$IFDEF DELPHI}Npt   : Int;{$ENDIF}
10489:              CurvIndex          : Int); external 'dmath';
10490: { Plots a Func }
10491: Proc WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10492:                 NCurv             : Int;
10493:                 ShowPoints, ShowLines : Bool); external 'dmath';
10494: { Writes the legends for the plotted curves }
10495: Proc ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10496:            Nx, Ny, Nc             : Int;
10497:            X, Y, Z                 : TVector;
10498:            F                       : TMatrix); external 'dmath';
10499: { Contour plot }
10500: Func Xpixel(X:Float):Int; external 'dmath';{Converts user abscissa X to screen coordinate }
10501: Func Ypixel(Y:Float):Int; external 'dmath';{Converts user ordinate Y to screen coordinate }
10502: Func Xuser(X:Int):Float; external 'dmath';{Converts screen coordinate X to user abscissa }
10503: Func Yuser(Y:Int):Float; external 'dmath';{Converts screen coordinate Y to user ordinate }
10504: {$IFDEF DELPHI}
10505: Proc LeaveGraphics; external 'dmath';
10506: { Quits graphic mode }
10507: {$ENDIF}
10508: { -----
10509:   LaTeX graphics
10510:   ----- }
10511: Func TeX_InitGraphics(FileName      :Str; PgWidth, PgHeight : Int;
10512:                      Header         :Bool) :Bool; external 'dmath';
10513: { Initializes the LaTeX file }
10514: Proc TeX_SetWindow(X1,X2,Y1 Y2:Int; GraphBorder :Bool); external 'dmath';
10515: { Sets the graphic window }
10516: Proc TeX_LeaveGraphics(Footer:Boolean);external 'dmath';{ Close LaTeX file }
10517: Proc TeX_SetOxScale(Scale:TScale;OxMin,OxMax,OxStep:Float);external 'dmath';
10518: { Sets the scale on the Ox axis }
10519: Proc TeX_SetOyScale(Scale:TScale;OyMin,OyMax,OyStep:Float); external 'dmath';
10520: { Sets the scale on the Oy axis }
10521: Proc TeX_SetGraphTitle(Title:str);external 'dmath';{ Sets title for graph }
10522: Proc TeX_SetOxTitle(Title:str);external 'dmath';{ Sets title for Ox axis }
10523: Proc TeX_SetOyTitle(Title:str);external 'dmath';{ Sets title for Oy axis }
10524: Proc TeX_PlotOxAxis; external 'dmath'; { Plots horizontal axis }
10525: Proc TeX_PlotOyAxis; external 'dmath'; { Plots vertical axis }
10526: Proc TeX_PlotGrid(Grid:TGrid);external 'dmath'; { Plots a grid on the graph }
10527: Proc TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10528: Proc TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10529: { Sets the maximum number of curves and re-initializes their parameters }
10530: Proc TeX_SetPointParam(CurvIndex, Symbol, Size : Int); external 'dmath';
10531: { Sets the point parameters for curve # CurvIndex }
10532: Proc TeX_SetLineParam(CurvIndex, Style : Int;
10533:                      Width : Float; Smooth :Bool); external 'dmath';
10534: { Sets the line parameters for curve # CurvIndex }
10535: Proc TeX_SetCurvLegend(CurvIndex : Int; Legend :Str); external 'dmath';
10536: { Sets the legend for curve # CurvIndex }
10537: Proc TeX_SetCurvStep(CurvIndex, Step : Int); external 'dmath';
10538: { Sets the step for curve # CurvIndex }
10539: Proc TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Int); external 'dmath';
10540: { Plots a curve }
10541: Proc TeX_PlotCurveWithErrorBars(X,Y,S:TVector;Ns,Lb,Ub,CurvIndex Int);external 'dmath';
10542: { Plots a curve with error bars }
10543: Proc TeX_PlotFunc(Func:TFunc;X1 X2:Float;Npt:Int;CurvIndex:Int);external 'dmath';
10544: { Plots a Func }
10545: Proc TeX_WriteLegend(NCurv:Int;ShowPoints,ShowLines:Bool); external 'dmath';
10546: { Writes the legends for the plotted curves }
10547: Proc TeX_ConRec(Nx,Ny Nc:Int; X,Y,Z :TVector; F : TMatrix); external 'dmath';
10548: { Contour plot }
10549: Func Xcm(X:Float):Float; external 'dmath'; { Converts user coordinate X to cm }
10550: Func Ycm(Y:Float):Float; external 'dmath'; { Converts user coordinate Y to cm }
10551:
10552: //*****unit uPSI_SynPdf;
10553: Func RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10554: Func _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10555: Func _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10556: Func PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10557: Func PdfRect1( const Box : TPdfBox) : TPdfRect;
10558: Func PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10559: //Func _GetCharCount( Text : PAnsiChar) : Int
10560: //Proc L2R( W : PWideChar; L : Int)
10561: Func PdfCoord( MM : single) : Int

```

```

10562: Func CurrentPrinterPaperSize : TPDPFPaperSize
10563: Func CurrentPrinterRes : TPoint
10564: Proc GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10565: Proc GDICommentOutline( MetaHandle:HDC; const aTitle : RawUTF8; aLevel : Int)
10566: Proc GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect: TRect)
10567: Const('Usp10', 'String' 'usp10.dll
10568: AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10569: 'fCharShape, fDigitSubstitute, fInhibitLigate, fDisplayZWG, fArabicNumContext, fGcpClusters )
10570: TScriptState_set', 'set of TScriptState_enum
10571: //*****
10572:
10573: Proc SIRegister_PMrand(CL: TPSPascalCompiler); //ParkMiller
10574: begin
10575:   Proc PMrandomize( I : word)
10576:   Func PMrandom : longint
10577:   Func Rrand : extended
10578:   Func Irand( N : word) : word
10579:   Func Brand( P : extended) :Bool
10580:   Func Nrand : extended
10581: end;
10582:
10583: Proc SIRegister_Spring_Cryptography_Utills(CL: TPSPascalCompiler);
10584: begin
10585:   Func Endian( x : LongWord) : LongWord
10586:   Func Endian64( x : Int64) : Int64
10587:   Func spRol( x : LongWord; y : Byte) : LongWord
10588:   Func spRor( x : LongWord; y : Byte) : LongWord
10589:   Func Ror64( x : Int64; y : Byte) : Int64
10590: end;
10591:
10592: Proc SIRegister_MapReader(CL: TPSPascalCompiler);
10593: begin
10594:   Proc ClearModules
10595:   Proc ReadMapFile( FName :Str)
10596:   Func AddressInfo( Address : dword) :Str
10597: end;
10598:
10599: Proc SIRegister_LibTar(CL: TPSPascalCompiler);
10600: begin
10601:   TTarPermission', '(tpReadByOwner, tpWriteByOwner, tpExecuteByOwner, tpReadByGroup, tpWriteByGroup,
tpExecuteByGroup, tpReadByOther, tpWriteByOther, tpExecuteByOther)
10602:   TTarPermissions', 'set of TTarPermission
10603:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10604:   +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10605:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10606:   TTarModes', 'set of TTarMode
10607:   TTarDirRec', 'record Name :Str; Size : INT64; DateTime : TDate'
10608:   +'eTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10609:   +'RING; UID : Int; GID : Int; UserName :Str; GroupName :Str;'
10610:   +'ChecksumOK :Bool; Mode : TTarModes; Magic :Str; MajorDevNo : INTE'
10611:   +'GER; MinorDevNo : Int; FilePos : INT64; end
10612:   SIRegister_TTarArchive(CL);
10613:   SIRegister_TTarWriter(CL);
10614:   Func PermissionString( Permissions : TTarPermissions) :Str
10615:   Func ConvertFilename( Filename :Str) :Str
10616:   Func FileTimeGMT( FileName :Str) : TDateTime;
10617:   Func FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10618:   Proc ClearDirRec( var DirRec : TTarDirRec)
10619: end;
10620:
10621: //*****unit uPSI_TlHelp32;
10622: Proc SIRegister_TlHelp32(CL: TPSPascalCompiler);
10623: begin
10624:   Const('MAX_MODULE_NAME32', 'LongInt'( 255);
10625:   Func CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10626:   Const('TH32CS_SNAPHEAPLIST', LongWord( $00000001);
10627:   Const('TH32CS_SNAPPROCESS', 'LongWord').SetUInt( $00000002);
10628:   Const('TH32CS_SNAPTHREAD', 'LongWord').SetUInt( $00000004);
10629:   Const('TH32CS_SNAPMODULE', 'LongWord').SetUInt( $00000008);
10630:   Const('TH32CS_INHERIT', 'LongWord').SetUInt( $80000000);
10631:   tagHEAPLIST32', 'record dwSize:DWORD; th32ProcessID:DWORD; th32HeapID:DWORD; dwFlags:DWORD; end';
10632:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10633:   AddTypeS('THeapList32', 'tagHEAPLIST32
10634:   Const('HF32_DEFAULT', 'LongInt'( 1);
10635:   Const('HF32_SHARED', 'LongInt'( 2);
10636:   Func Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10637:   Func Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10638:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10639:   +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10640:   +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10641:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10642:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10643:   Const('LF32_FIXED', 'LongWord').SetUInt( $00000001);
10644:   Const('LF32_FREE', 'LongWord').SetUInt( $00000002);
10645:   Const('LF32_MOVEABLE', 'LongWord').SetUInt( $00000004);
10646:   Func Heap32First( var lphe:THeapEntry32; th32ProcessID, th32HeapID:DWORD) :BOOL
10647:   Func Heap32Next( var lphe : THeapEntry32) : BOOL
10648:   DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10649:   AddTypeS('tagTHREADEENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'

```

```

10650:   +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10651:   +'aPri : Longint; dwFlags : DWORD; end
10652:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10653:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10654:   Func Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10655:   Func Thread32Next( hSnapshot : THandle; var lpte : TThreadEntry32):BOOL end;
10656:   Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10657:   Const('EW_REBOOTSYSTEM','LongWord( $0043);
10658:   Const('EW_EXITANDEXECAPP','LongWord( $0044);
10659:   Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10660:   Const('EWX_LOGOFF','LongInt'( 0);
10661:   Const('EWX_SHUTDOWN','LongInt'( 1);
10662:   Const('EWX_REBOOT','LongInt'( 2);
10663:   Const('EWX_FORCE','LongInt'( 4);
10664:   Const('EWX_POWEROFF','LongInt'( 8);
10665:   Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10666:   Func GET_APPCOMMAND_LPARAM( const lParam : Longint) : Shortint
10667:   Func GET_DEVICE_LPARAM( const lParam : Longint) : Word
10668:   Func GET_MOUSEORKEY_LPARAM( const lParam : Longint) : Word
10669:   Func GET_FLAGS_LPARAM( const lParam : Longint) : Word
10670:   Func GET_KEYSTATE_LPARAM( const lParam : Longint) : Word
10671:   Func GetWindowWord( hWnd : HWND; nIndex : Int) : Word
10672:   Func SetWindowWord( hWnd : HWND; nIndex : Int; wNewWord : Word) : Word
10673:   Func GetWindowLong( hWnd : HWND; nIndex : Int) : Longint
10674:   Func SetWindowLong( hWnd : HWND; nIndex : Int; dwNewLong:Longint) : Longint
10675:   Func GetClassWord( hWnd : HWND; nIndex : Int) : Word
10676:   Func SetClassWord( hWnd : HWND; nIndex : Int; wNewWord : Word) : Word
10677:   Func GetClassLong( hWnd : HWND; nIndex : Int) : DWORD
10678:   Func SetClassLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint) : DWORD
10679:   Func GetDesktopWindow : HWND
10680:   Func GetParent( hWnd : HWND) : HWND
10681:   Func SetParent( hWndChild, hWndNewParent : HWND) : HWND
10682:   Func GetTopWindow( hWnd : HWND) : HWND
10683:   Func GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10684:   Func GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10685:   //Delphi DFM
10686:   Func LoadDFMFile2Strings(const AFile:str; AStrings:TStrings; var WasText:bool):Int
10687:   Func SaveStrings2DFMFile( AStrings : TStrings; const AFile :Str) : Int
10688:   Proc GetHighlighters(AOwner:TComponent; AHighlighters:TStringList;AppendToList:Bool);
10689:   Func GetHighlightersFilter(AHighlighters: TStringList):Str;
10690:   Func GetHighlighterFromFileExt(AHighlighters: TStringList;Extension:Str):TSynCustomHighlighter;
10691:   Func ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10692:   Func OpenIcon( hWnd : HWND) : BOOL
10693:   Func CloseWindow( hWnd : HWND) : BOOL
10694:   Func MoveWindow(hWnd:HWND;X Y,nWidth, nHeight : Int; bRepaint : BOOL) : BOOL
10695:   Func SetWindowPos(hWnd:HWND;hWndInsertAfter:HWND;X,Y,cx,cy:Int;uFlags:UINT): BOOL
10696:   Func IsWindowVisible( hWnd : HWND) : BOOL
10697:   Func IsIconic( hWnd : HWND) : BOOL
10698:   Func AnyPopup : BOOL
10699:   Func BringWindowToTop( hWnd : HWND) : BOOL
10700:   Func IsZoomed( hWnd : HWND) : BOOL
10701:   Func IsWindow( hWnd : HWND) : BOOL
10702:   Func IsMenu( hMenu : HMENU) : BOOL
10703:   Func IsChild( hWndParent, hWnd : HWND) : BOOL
10704:   Func DestroyWindow( hWnd : HWND) : BOOL
10705:   Func ShowWindow( hWnd : HWND; nCmdShow : Int) : BOOL
10706:   Func AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10707:   Func ShowWindowAsync( hWnd : HWND; nCmdShow : Int) : BOOL
10708:   Func FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10709:   Func IsWindowUnicode( hWnd : HWND) : BOOL
10710:   Func EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10711:   Func IsWindowEnabled( hWnd : HWND) : BOOL
10712:
10713: Proc SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10714: begin
10715:   const('ShowSetupDialogOptLong','String '--setup
10716:   PrimaryConfPathOptLong','String '--primary-config-path=
10717:   PrimaryConfPathOptShort','String '--pcp=
10718:   SecondaryConfPathOptLong','String '--secondary-config-path=
10719:   SecondaryConfPathOptShort','String '--scp=
10720:   NoSplashScreenOptLong','String '--no-splash-screen
10721:   NoSplashScreenOptShort','String '--nsc
10722:   StartedByStartLazarusOpt','String '--started-by-startlazarus
10723:   SkipLastProjectOpt','String '--skip-last-project
10724:   DebugLogOpt','String '--debug-log=
10725:   DebugLogOptEnable','String '--debug-enable=
10726:   LanguageOpt','String '--language=
10727:   LazarusDirOpt','String '--lazarusdir=
10728:   Proc ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:bool);
10729:   Func GetCommandLineParameters(aCmdLineParams:TStrings;isStartLazarus:Bool):Str
10730:   Func ExtractPrimaryConfigPath( aCmdLineParams : TStrings) :Str
10731:   Func IsHelpRequested :Bool
10732:   Func IsVersionRequested :Bool
10733:   Func GetLanguageSpecified :Str
10734:   Func ParamIsOption( ParamIndex : Int; const Option :Str) :Bool
10735:   Func ParamIsOptionPlusValue(ParamIndex:int;const Option:str;out AValue:str):bool;
10736:   Proc ParseNoGuiCmdLineParams
10737:   Func ExtractCmdLineFileNames : TStrings
10738: end;

```

```

10739:
10740: Proc SRegister_LazFileUtils(CL: TPSPascalCompiler);
10741: begin
10742:   Func CompareFileNames( const Filename1, Filename2 :Str) : Int
10743:   Func CompareFileNamesIgnoreCase( const Filename1, Filename2 :Str) : Int
10744:   Func CompareFileExt( const Filename, Ext :Str; CaseSensitive :Bool) : Int;
10745:   Func CompareFileExt1( const Filename, Ext :Str) : Int;
10746:   Func CompareFilenameStarts( const Filename1, Filename2 :Str) : Int
10747:   Func CompareFileNames(Filename1:PChar;Len1:Int; Filename2:PChar;Len2:Int):Int
10748:   Func CompareFileNamesP(Filename1,Filename2:PChar;IgnoreCase :Bool) : Int
10749:   Func DirPathExists( DirectoryName :Str) :Bool
10750:   Func DirectoryIsWritable( const DirectoryName :Str) :Bool
10751:   Func ExtractFileNameOnly( const AFilename :Str) :Str
10752:   Func FilenameIsAbsolute( const TheFilename :Str) :Bool
10753:   Func FilenameIsWinAbsolute( const TheFilename :Str) :Bool
10754:   Func FilenameIsUnixAbsolute( const TheFilename :Str) :Bool
10755:   Func ForceDirectory( DirectoryName :Str) :Bool
10756:   Proc CheckIfFileIsExecutable( const AFilename :Str)
10757:   Proc CheckIfFileIsSymlink( const AFilename :Str)
10758:   Func FileIsText( const AFilename :Str) :Bool
10759:   Func FileIsText2(const AFilename:str; out FileReadable: bool): bool
10760:   Func FilenameIsTrimmed( const TheFilename :Str) :Bool
10761:   Func FilenameIsTrimmed2( StartPos : PChar; NameLen : Int) :Bool
10762:   Func TrimFilename( const AFilename :Str) :Str
10763:   Func ResolveDots( const AFilename :Str) :Str
10764:   Proc ForcePathDelims( var FileName :Str)
10765:   Func GetForcedPathDelims( const FileName :Str) :Str
10766:   Func CleanAndExpandFilename( const Filename :Str) :Str
10767:   Func CleanAndExpandDirectory( const Filename :Str) :Str
10768:   Func TrimAndExpandFilename(const Filename:str;const BaseDir:str):str
10769:   Func TrimAndExpandDirectory(const Filename:str;const BaseDir:str):Str
10770:   Func TryCreateRelativePath(const Dest,Source:str; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder:Bool; out RelPath :Str) :Bool
10771:   Func CreateRelativePath(const Filename,
BaseDirectory:str;UsePointDirectory:boolean;AlwaysRequireSharedBaseFolder:Bool) :Str
10772:   Func FileIsInPath( const Filename, Path :Str) :Bool
10773:   Func AppendPathDelim( const Path :Str) :Str
10774:   Func ChompPathDelim( const Path :Str) :Str
10775:   Func CreateAbsoluteSearchPath(const SearchPath,BaseDirectory:str):str
10776:   Func CreateRelativeSearchPath(const SearchPath,BaseDirectory:str):Str
10777:   Func MinimizeSearchPath( const SearchPath :Str) :Str
10778:   Func FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10779:   (*Func FileExistsUTF8( const Filename :Str) :Bool
10780:   Func FileAgeUTF8( const FileName :Str) : Longint
10781:   Func DirectoryExistsUTF8( const Directory :Str) :Bool
10782:   Func ExpandFileNameUTF8( const FileName :Str; BaseDir :Str) :Str
10783:   Func FindFirstUTF8(const Path:str;Attr:Longint;out Rslt:TSearchRec): Longint
10784:   Func FindNextUTF8( var Rslt : TSearchRec) : Longint
10785:   Proc FindCloseUTF8( var F : TSearchrec)
10786:   Func FileSetDateUTF8( const FileName :Str; Age : Longint) : Longint
10787:   Func FileGetAttrUTF8( const FileName :Str) : Longint
10788:   Func FileSetAttrUTF8( const Filename :Str; Attr : longint) : Longint
10789:   Func DeleteFileUTF8( const FileName :Str) :Bool
10790:   Func RenameFileUTF8( const OldName, NewName :Str) :Bool
10791:   Func FileSearchUTF8(const Name,DirList:str;ImplicitCurrentDir:Bool):str
10792:   Func FileIsReadOnlyUTF8( const FileName :Str) :Bool
10793:   Func GetCurrentDirUTF8 :Str
10794:   Func SetCurrentDirUTF8( const NewDir :Str) :Bool
10795:   Func CreateDirUTF8( const NewDir :Str) :Bool
10796:   Func RemoveDirUTF8( const Dir :Str) :Bool
10797:   Func ForceDirectoriesUTF8( const Dir :Str) :Bool
10798:   Func FileOpenUTF8( const FileName :Str; Mode : Int) : THandle
10799:   Func FileCreateUTF8( const FileName :Str) : THandle;
10800:   Func FileCreateUTF81( const FileName :Str; Rights :Card): THandle;
10801:   Func FileCreateUtf82(const FileName:str;ShareMode:Int;Rights:Card): THandle;
10802:   Func FileSizeUtf8( const Filename :Str) : int64
10803:   Func GetFileDescription( const AFilename :Str) :Str
10804:   Func GetAppConfigDirUTF8( Global :Bool; Create :Bool) :Str
10805:   Func GetAppConfigFileUTF8(Global:Bool;SubDir:bool;CreateDir: bool):Str
10806:   Func GetTempFileNameUTF8( const Dir, Prefix :Str) :Str*)
10807:   Func IsUNCPath( const Path :Str) :Bool
10808:   Func ExtractUNCVolume( const Path :Str) :Str
10809:   Func ExtractFileRoot( FileName :Str) :Str
10810:   Func GetDarwinSystemFilename( Filename :Str) :Str
10811:   Proc SplitCmdLineParams( const Params:str; ParamList:TStrings; ReadBackslash:bool)
10812:   Func StrToCmdLineParam( const Param :Str) :Str
10813:   Func MergeCmdLineParams( ParamList : TStrings) :Str
10814:   Proc InvalidateFileStateCache( const Filename :Str)
10815:   Func FindAllFiles(const SearchPath:str;SearchMask:Str;SearchSubDirs:Bool):TStringList;
10816:   Func FindAllDirectories(const SearchPath:str;SearchSubDirs:Bool):TStringList
10817:   Func FindAllDocs(const Root, extmask:Str): TStringlist;
10818:   Func ReadFileToString( const Filename:Str):Str
10819:   Proc Incl(var X: longint; N: Longint);
10820:   type TCopyFileFlag = ( cffOverwriteFile,
10821:   cffCreateDestDirectory, cffPreserveTime );
10822:   TCopyFileFlags = set of TCopyFileFlag;*)
10823:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10824:   TCopyFileFlags', 'set of TCopyFileFlag
10825:   Func CopyDirTree(const SourceDir,TargetDir:str;Flags:TCopyFileFlags):Bool

```



```

10826: end;
10827:
10828: Proc SIRegister_lazMasks(CL: TPSPascalCompiler);
10829: begin
10830:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10831:   SIRegister_TMask(CL);
10832:   SIRegister_TParseStringList(CL);
10833:   SIRegister_TMaskList(CL);
10834:   Func MatchesMask(const FileName,Mask:str;const CaseSensitive:Bool):Bool
10835:   Func MatchesWindowsMask(const FileName,Mask:str;const CaseSensitive:Bool): Bool;
10836:   Func MatchesMaskList(const FileName,Mask:Str;Separatr:Char;const CaseSensitive:Bool):Bool;
10837:   Func MatchesWindowsMaskList(const FileName,Mask:str;Separat:Char;const CaseSensitive:Bool):Bool;
10838: end;
10839:
10840: Proc SIRegister_JvShellHook(CL: TPSPascalCompiler);
10841: begin
10842:   //PShellHookInfo', '^TShellHookInfo // will not work
10843:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10844:   SHELLHOOKINFO', 'TShellHookInfo
10845:   LPSHELLHOOKINFO', 'PShellHookInfo
10846:   TjvShellHookEvent', 'Proc ( Sender : TObject; var Msg : TMessage)
10847:   SIRegister_TjvShellHook(CL);
10848:   Func InitJvShellHooks :Bool
10849:   Proc UnInitJvShellHooks
10850: end;
10851:
10852: Proc SIRegister_JvExControls(CL: TPSPascalCompiler);
10853: begin
10854:   TDlgCode', '(dcWantAllKeys,dcWantArrows,dcWantChars,dcButton,dcHasSetSel,dcWantTab,dcNative)
10855:   TDlgCodes', 'set of TDlgCodedcWantMessage', ' dcWantAllKeys);
10856:   SIRegister_IJvExControl(CL);
10857:   SIRegister_IJvDenySubClassing(CL);
10858:   SIRegister_TStructPtrMessage(CL);
10859:   Proc SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10860:   Proc DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl :Bool);
10861:   Proc DrawDotNetControl1(DC:HDC; R:TRect;AColor:TColor;UseFocusedColor:Bool);
10862:   Proc HandleDotNetHighlighting(Cntrl:TWinControl;const Msg:TMessage;MouseOver:Bool;Color:TColor);
10863:   Func CreateWMMMessage( Msg : Int; WParam : Int; LParam : Longint ) : TMessage;
10864:   Func CreateWMMMessage1( Msg : Int; WParam : Int; LParam : TControl ) : TMessage;
10865:   Func SmallPointToLong( const Pt : TSmallPoint ) : Longint
10866:   Func ShiftStateToKeyData( Shift : TShiftState ) : Longint
10867:   Func GetFocusedControl( AControl : TControl ) : TWinControl
10868:   Func DlgcToDlgCodes( Value : Longint ) : TDlgCodes
10869:   Func DlgCodesToDlgc( Value : TDlgCodes ) : Longint
10870:   Proc GetHintColor(var HintInfo:THintInfo AControl:TControl;HintColor:TColor)
10871:   Func DispatchIsDesignMsg( Control : TControl; var Msg : TMessage ) :Bool
10872:   SIRegister_TJvExControl(CL);
10873:   SIRegister_TJvExWinControl(CL);
10874:   SIRegister_TJvExCustomControl(CL);
10875:   SIRegister_TJvExGraphicControl(CL);
10876:   SIRegister_TJvExHintWindow(CL);
10877:   SIRegister_TJvExPubGraphicControl(CL);
10878: end;
10879:
10880: (*-----*)
10881: Proc SIRegister_EncdDecd(CL: TPSPascalCompiler);
10882: begin
10883:   Proc EncodeStream(Input, Output : TStream)
10884:   Proc DecodeStream(Input, Output : TStream)
10885:   Func EncodeString1(const Input :Str) :Str
10886:   Func DecodeString1(const Input :Str) :Str
10887: end;
10888:
10889: (*-----*)
10890: Proc SIRegister_SockAppReg(CL: TPSPascalCompiler);
10891: begin
10892:   SIRegister_TWebAppRegInfo(CL);
10893:   SIRegister_TWebAppRegList(CL);
10894:   Proc GetRegisteredWebApps( AList : TWebAppRegList)
10895:   Proc RegisterWebApp( const AFileName, AProgID :Str)
10896:   Proc UnregisterWebApp( const AProgID :Str)
10897:   Func FindRegisteredWebApp( const AProgID :Str) :Str
10898:   Func CreateRegistry(InitializeNewFile:Bool):TCustomIniFilesUDPPort',String 'UDPPort
10899: end;
10900:
10901: Proc SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10902: begin
10903:   // TStringDynArray', 'array of string
10904:   Func GetEnvVarValue( const VarName :Str) :Str
10905:   Func SetEnvVarValue( const VarName, VarValue :Str) : Int
10906:   Func DeleteEnvVar( const VarName :Str) : Int
10907:   Func CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:str;const
BufSize:Int):Int;
10908:   Func ExpandEnvVars( const Str :Str) :Str
10909:   Func GetAllEnvVars( const Vars : TStrings) : Int
10910:   Proc GetAllEnvVarNames( const Names : TStrings);
10911:   Func GetAllEnvVarNames1 : TStringDynArray;
10912:   Func EnvBlockSize : Int
10913:   TPJEnvVarsEnum', 'Proc ( const VarName :Str; Data : TObject)

```

```

10914:   SIRegister_TPJEnvVarsEnumerator(CL);
10915:   SIRegister_TPJEnvVars(CL);
10916:   FindClass('TOBJECT'),'EPJEnvVars
10917:   FindClass('TOBJECT'),'EPJEnvVars
10918:   //Proc Register
10919: end;
10920:
10921: (*-----*)
10922: Proc SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10923: begin
10924:   'cOneSecInMS','LongInt'( 1000);
10925:   //'cDefTimeSlice','LongInt'( 50);
10926:   //'cDefMaxExecTime',' cOneMinInMS';
10927:   'cAppErrorMask','LongInt'( 1 shl 29);
10928:   Func IsApplicationError( const ErrCode : LongWord) :Bool
10929:   TPJConsoleAppPriority',( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10930:   TPJConsoleColors','record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10931:   Func MakeConsoleColors(const AForeground,ABackground:TPJConsoleColor):TPJConsoleColors;
10932:   Func MakeConsoleColors1(const AForeground,ABackground:TColor): TPJConsoleColors;
10933:   Func MakeConsoleColors2(const AForeground ABackgroud:TAlphaColor): TPJConsoleColors;
10934:   Func MakeSize( const ACX, ACY : LongInt) : TSize
10935:   SIRegister_TPJCustomConsoleApp(CL);
10936:   SIRegister_TPJConsoleApp(CL);
10937: end;
10938:
10939: Proc SIRegister_ip_misc(CL: TPSPascalCompiler);
10940: begin
10941:   INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffff);
10942:   t_encoding', '( uuencode, base64, mime )
10943:   Func internet_date( date : TDateTime) :Str
10944:   Func lookup_hostname( const hostname :Str) : longint
10945:   Func my_hostname :Str
10946:   Func my_ip_address : longint
10947:   Func ip2string( ip_address : longint) :Str
10948:   Func resolve_hostname( ip : longint) :Str
10949:   Func address_from( const s :Str; count : Int) :Str
10950:   Func encode_base64( data : TStream) : TStringList
10951:   Func decode_base64( source : TStringList) : TMemoryStream
10952:   Func posn( const s, t :Str; count : Int) : Int
10953:   Func poscn( c : char; const s :Str; n : Int) : Int
10954:   Func filename_of( const s :Str) :Str
10955:   //Func trim( const s :Str) :Str
10956:   //Proc setlength( var s :Str; l : byte)
10957:   Func TimeZoneBias : longint
10958:   Func eight2seven_quoteprint( const s :Str) :Str
10959:   Func eight2seven_german( const s :Str) :Str
10960:   Func seven2eight_quoteprint( const s :Str) :Str end;
10961:   type in_addr, 'record s_bytes : array[1..4] of byte; end;
10962:   Func socketerror : cint
10963:   Func fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10964:   Func fprecv( s : cint; buf : __pointer; len : size_t; flags : cint): ssize_t
10965:   Func fsend( s : cint; msg : __pointer; len : size_t; flags : cint): ssize_t
10966:   //Func fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10967:   Func fplisten( s : cint; backlog : cint) : cint
10968:   //Func fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10969:   //Func fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10970:   //Func fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10971:   Func NetAddrToStr( Entry : in_addr) :Str
10972:   Func HostAddrToStr( Entry : in_addr) :Str
10973:   Func StrToHostAddr( IP :Str) : in_addr
10974:   Func StrToNetAddr( IP :Str) : in_addr
10975:   SOL_SOCKET','LongWord').SetUInt( $ffff);
10976:   cint8', 'shortint
10977:   cuint8', 'byte
10978:   cchar', 'cint8
10979:   cschar', 'cint8
10980:   cuchar', 'cuint8
10981:   cint16', 'smallint
10982:   cuint16', 'word
10983:   cshort', 'cint16
10984:   csshort', 'cint16
10985:   cushort', 'cuint16
10986:   cint32', 'longint
10987:   cuint32', 'longword
10988:   cint', 'cint32
10989:   csint', 'cint32
10990:   cuint', 'cuint32
10991:   csigned', 'cint
10992:   cunsigned', 'cuint
10993:   cint64', 'int64
10994:   clonglong', 'cint64
10995:   cslonglong', 'cint64
10996:   cbool', 'longbool
10997:   cfloat', 'single
10998:   cdouble', 'double
10999:   clongdouble', 'extended
11000:
11001: Proc SIRegister_uLkJSON(CL: TPSPascalCompiler);
11002: begin

```

```

11003:   TlkJSONtypes', '(jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
11004:   SIRegister_TlkJSONdotnetclass(CL);
11005:   SIRegister_TlkJSONbase(CL);
11006:   SIRegister_TlkJSONnumber(CL);
11007:   SIRegister_TlkJSONstring(CL);
11008:   SIRegister_TlkJSONboolean(CL);
11009:   SIRegister_TlkJSONnull(CL);
11010:   TlkJSONFuncEnum', 'Procedure (ElName: str; Elem: TlkJSONbase; data: TObj; var Continue: Boolean)
11011:   SIRegister_TlkJSONcustomlist(CL);
11012:   SIRegister_TlkJSONlist(CL);
11013:   SIRegister_TlkJSONobjectmethod(CL);
11014:   TlkHashItem', 'record hash : Card; index : Int; end
11015:   TlkHashFunction', 'Func ( const ws : WideString) : Card
11016:   SIRegister_TlkHashTable(CL);
11017:   SIRegister_TlkBalTree(CL);
11018:   SIRegister_TlkJSONobject(CL);
11019:   SIRegister_TlkJSON(CL);
11020:   SIRegister_TlkJSONstreamed(CL);
11021:   Func GenerateReadableText( vObj : TlkJSONbase; var vLevel : Int): Str
11022: end;
11023:
11024: Ex: js:= TlkJSON.create;
11025:   jsonObject:= js.ParseText(jsonstring) as TlkJSONobject;
11026:   try
11027:     writeln('get message: '+jsonObject.Field['amessage'].value);
11028:     writeln(vartostr(jsonObject.Field['book'].Child[3].field['ISBN'].value));
11029:
11030: Proc SIRegister_ZSysUtils(CL: TPSPascalCompiler);
11031: begin
11032:   TZListSortCompare', 'Func (Item1, Item2 : TObj): Int
11033:   SIRegister_TZSortedlist(CL);
11034:   Func zFirstDelimiter( const Delimiters, Str : Str) : Int
11035:   Func zLastDelimiter( const Delimiters, Str : Str) : Int
11036:   //Func MemLCompUnicode( P1, P2 : PWideChar; Len : Int) : Bool
11037:   //Func MemLCompAnsi( P1, P2 : PAnsiChar; Len : Int) : Bool
11038:   Func zStartsWith( const Str, SubStr : WideString) : Bool;
11039:   Func StartsWith1( const Str, SubStr : RawByteString) : Bool;
11040:   Func EndsWith( const Str, SubStr : WideString) : Bool;
11041:   Func EndsWith1( const Str, SubStr : RawByteString) : Bool;
11042:   Func SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
11043:   Func SQLStrToFloatDef1( Str : Str; Def : Extended) : Extended;
11044:   Func SQLStrToFloat( const Str : Ansistr) : Extended
11045:   //Func BufferToStr( Buffer : PWideChar; Length : LongInt) : Str;
11046:   //Func BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : Str;
11047:   Func BufferToBytes( Buffer : TObj; Length : LongInt) : TByteDynArray
11048:   Func StrToBoolEx( Str : Str) : Bool
11049:   Func BoolToStrEx( Bool : Bool) : Str
11050:   Func IsIPAddr( const Str : Str) : Bool //IsIP()
11051:   Func zSplitString( const Str, Delimiters : Str) : TStrings
11052:   Proc PutSplitString( List : TStrings; const Str, Delimiters : Str)
11053:   Proc AppendSplitString( List : TStrings; const Str, Delimiters : Str)
11054:   Func ComposeString( List : TStrings; const Delimiter : Str) : Str
11055:   Func FloatToSQLStr( Value : Extended) : Str
11056:   Proc PutSplitStringEx( List : TStrings; const Str, Delimiter : Str)
11057:   Func SplitStringEx( const Str, Delimiter : Str) : TStrings
11058:   Proc AppendSplitStringEx( List : TStrings; const Str, Delimiter:Str)
11059:   Func zBytesToStr( const Value : TByteDynArray) : Ansistr
11060:   Func zStrToBytes( const Value : Ansistr) : TByteDynArray;
11061:   Func StrToBytes1( const Value : UTF8String) : TByteDynArray;
11062:   Func StrToBytes2( const Value : RawByteString) : TByteDynArray;
11063:   Func StrToBytes3( const Value : WideString) : TByteDynArray;
11064:   Func StrToBytes4( const Value : UnicodeString) : TByteDynArray;
11065:   Func BytesToVar( const Value : TByteDynArray) : Variant
11066:   Func VarToBytes( const Value : Variant) : TByteDynArray
11067:   Func AnsiSQLDateToDateTime( const Value : Str) : TDateTime
11068:   Func TimestampStrToDateTime( const Value : Str) : TDateTime
11069:   Func DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec:Bool): Str
11070:   Func EncodeCString( const Value : Str) : Str
11071:   Func DecodeCString( const Value : Str) : Str
11072:   Func zReplaceChar( const Source, Target : Char; const Str:Str): Str
11073:   Func MemPas( Buffer : PChar; Length : LongInt) : Str
11074:   Proc DecodeSQLVersioning( const FullVersion: Int; out MajorVersion: Int; out MinorVersion: Int; out
SubVersion: Int);
11075:   Func EncodeSQLVersioning( const MajorVersion: Int; const MinorVersion: Int; const SubVersion: Int): Int;
11076:   Func FormatSQLVersion( const SQLVersion : Int) : Str
11077:   Func ZStrToFloat( Value : AnsiChar) : Extended;
11078:   Func ZStrToFloat1( Value : Ansistr) : Extended;
11079:   Proc ZSetString( const Src : AnsiChar; var Dest : Ansistr);
11080:   Proc ZSetString1( const Src: AnsiChar; const Len: Card; var Dest: Ansistr);
11081:   Proc ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
11082:   Proc ZSetString3( const Src: AnsiChar; const Len: Card; var Dest: UTF8String);
11083:   Proc ZSetString4( const Src: AnsiChar; const Len: Card; var Dest: WideString);
11084:   Proc ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
11085:   Proc ZSetString6( const Src: AnsiChar; const Len: Card; var Dest: RawByteString);
11086: end;
11087:
11088: unit uPSI_ZEncoding;
11089:   Func StringToAnsiEx( const s : Str; const FromCP, ToCP : Word) : RawByteString
11090:   Func AnsiToStringEx( const s: RawByteString; const FromCP, ToCP : Word : Str

```

```

11091: Func ZRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
11092: Func ZUnicodeToRaw( const US : WideString; CP : Word ) : RawByteString
11093: Func ZConvertAnsiToRaw( const Src : Ansistr; const RawCP : Word ) : RawByteString
11094: Func ZConvertRawToAnsi( const Src:RawByteString;const RawCP:Word):Ansistr
11095: Func ZConvertAnsiToUTF8( const Src : Ansistr ) : UTF8String
11096: Func ZConvertUTF8ToAnsi( const Src : UTF8String ) : Ansistr
11097: Func ZConvertRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
11098: Func ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
11099: Func ZConvertRawToString( const Src:RawByteString;const RawCP,StringCP:Word):str
11100: Func ZConvertStringToRaw( const Src :Str; const StringCP, RawCP : Word ) : RawByteString
11101: Func ZConvertStringToRawWithAutoEncode( const Src:str;const StringCP,RawCP:Word):RawByteString;
11102: Func ZConvertUTF8ToString( const Src:UTF8String;const StringCP: Word):Str
11103: Func ZConvertStringToUTF8( const Src:Str;const StringCP:Word):UTF8String
11104: Func ZConvertStringToUTF8WithAutoEncode( const Src:str;const StringCP:Word): UTF8String
11105: Func ZConvertStringToAnsi( const Src :Str; const StringCP : Word ) : Ansistr
11106: Func ZConvertStringToAnsiWithAutoEncode( const Src:str;const StringCP:Word): Ansistr
11107: Func ZConvertAnsiToString( const Src:Ansistr;const StringCP:Word):str
11108: Func ZConvertUnicodeToString( const Src:WideString;const StringCP:Word):str
11109: Func ZConvertUnicodeToString_CPUTF8( const Src:WideString; const StringCP : Word):Str
11110: Func ZConvertStringToUnicode( const Src :Str; const StringCP : Word ) : WideString
11111: Func ZConvertString_CPUTF8ToUnicode( const Src:str;const StringCP:Word): WideString
11112: Func ZConvertStringToUnicodeWithAutoEncode( const Src:str;const StringCP:Word):WideString
11113: Func ZMoveAnsiToRaw( const Src : Ansistr; const RawCP : Word ) : RawByteString
11114: Func ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word ) : Ansistr
11115: Func ZMoveAnsiToUTF8( const Src : Ansistr ) : UTF8String
11116: Func ZMoveUTF8ToAnsi( const Src : UTF8String ) : Ansistr
11117: Func ZMoveRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
11118: Func ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
11119: Func ZMoveStringToAnsi( const Src :Str; const StringCP : Word ) : Ansistr
11120: Func ZMoveAnsiToString( const Src : Ansistr; const StringCP : Word ) :Str
11121: Func ZMoveRawToString( const Src:RawByteString;const RawCP,StringCP:Word):Str
11122: Func ZMoveStringToRaw( const Src :Str; const StringCP, RawCP : Word ) : RawByteString
11123: Func ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word ) :Str
11124: Func ZMoveStringToUTF8( const Src :Str; const StringCP : Word ) : UTF8String
11125: Func ZUnknownRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
11126: Func ZUnknownRawToUnicodeWithAutoEncode( const S:RawByteString;const CP:Word): WideString
11127: Func ZUnicodeToUnknownRaw( const US : WideString; CP : Word ) : RawByteString
11128: Func ZDefaultSystemCodePage : Word
11129: Func ZCompatibleCodePages( const CP1, CP2 : Word ) :Bool
11130: Func MPing( const AHost:Str;const ATimes:Int; out AvgMS:Double):Boolean;
11131:
11132: Proc SIRegister_BoldComUtils( CL: TPSPascalCompiler );
11133: begin
11134:   'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0 );
11135:   ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1 );
11136:   ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2 );
11137:   ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3 );
11138:   ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4 );
11139:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5 );
11140:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6 );
11141:   (('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
11142:   ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
11143:   ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
11144:   ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
11145:   ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
11146:   ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
11147:   ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
11148:   ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0 );
11149:   ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1 );
11150:   ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2 );
11151:   ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3 );
11152:   ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4 );
11153:   (('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
11154:   ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
11155:   ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
11156:   ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
11157:   ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
11158:   ('EOAC_NONE','LongWord').SetUInt( $0 );
11159:   ('EOAC_DEFAULT','LongWord').SetUInt( $800 );
11160:   ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1 );
11161:   ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20 );
11162:   ('EOAC_DYNAMIC_CLOACKING','LongWord').SetUInt( $40 );
11163:   ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80 );
11164:   ('RPC_C_AUTHN_WINNT','LongInt'( 10 );
11165:   ('RPC_C_AUTHNZ_NONE','LongInt'( 0 );
11166:   ('RPC_C_AUTHNZ_NAME','LongInt'( 1 );
11167:   ('RPC_C_AUTHNZ_DCE','LongInt'( 2 );
11168:   FindClass('TOBJECT'),'EBoldCom
11169: Func BoldVariantIsType( V : OleVariant; TypeCode : Int ) :Bool
11170: Func BoldMemoryToVariant( const Buffer, BufSize : Int ) : OleVariant
11171: Func BoldStreamToVariant( Stream : TStream ) : OleVariant
11172: Func BoldStringsToVariant( Strings : TStrings ) : OleVariant
11173: Func BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Int ) : Int
11174: Func BoldVariantToStream( V : OleVariant; Stream : TStream ) : Int
11175: Func BoldVariantArrayOfArrayOfStringsToStrings(V:OleVariant;Strings: TStrings) : Int
11176: Func BoldVariantIsNamedValues( V : OleVariant ) :Bool
11177: Func BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
11178: Func BoldGetNamedValue( Data : OleVariant; const Name :Str ) : OleVariant
11179: Proc BoldSetNamedValue( Data : OleVariant; const Name :Str; Value : OleVariant)

```

```

11180: Func BoldCreateGUID : TGUID
11181: Func BoldCreateComObject( const ClsId,IId:TGUID; out Obj:variant;out Res:HResult):Bool
11182: Func BoldCreateRemoteComObject(const HostName:str;const ClsId,IId:TGUID;out Obj:variant;out
Res:HRes):Bool;
11183: Proc BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11184: Proc BoldSetSecurityForInterface( AuthenticationLevel, ImpersonationLevel:longint;Unk:IUnknown);
11185: end;
11186:
11187: (*-----*)
11188: Proc SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11189: begin
11190:   Func ParseISODate( s :Str) : TDateTime
11191:   Func ParseISODateTime( s :Str) : TDateTime
11192:   Func ParseISOTime( str :Str) : TDateTime
11193: end;
11194:
11195: (*-----*)
11196: Proc SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11197: begin
11198:   Func BoldCreateGUIDAsString( StripBrackets :Bool) :Str
11199:   Func BoldCreateGUIDWithBracketsAsString :Str
11200: end;
11201:
11202: Proc SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11203: begin
11204:   FindClass('TOBJECT'),'T-BoldFileHandler
11205:   FindClass('TOBJECT'),'T-BoldDiskFileHandler
11206:   //T-BoldFileHandlerClass', 'class of T-BoldFileHandler
11207:   T-BoldInitializeFileContents', 'Proc ( StringList : TStringList)
11208:   SIRegister_TBoldFileHandler(CL);
11209:   SIRegister_TBoldDiskFileHandler(CL);
11210: Proc BoldCloseAllFilehandlers
11211: Proc BoldRemoveUnchangedFilesFromEditor
11212: Func BoldFileHandlerList : TBoldObjectArray
11213: Func BoldFileHandlerForFile(path,
FileName:str;ModuleType:TBoldModuleType;ShowInEditor:Bool;OnInitializeFileContents:TBoldInitializeFileContents):
TBoldFileHandler
11214: end;
11215:
11216: Proc SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11217: begin
11218:   PCharArr', 'array of PChar
11219: Func BoldInternetOpen(Agent:Str;AccessType:Int;Proxy:str;ProxyByPass:Str;Flags:Int):ptr);
11220: Func BoldInternetOpenUrl(iNet:Poin;URL:str;Headers:str;Flags,Context:Card):Point
11221: Func BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumOfBytesToRead:Card;var
NumberOfBytesRead:Card):LongBool;
11222: Func BoldInternetCloseHandle( HINet : Pointer) : LongBool
11223: Func BoldHttpQueryInfo(hRequest:Pointer;InfoLevel:Card;Buffer:Pointer;BufferLength:Card;Reserved:Card) :
LongBool
11224: Func BoldInternetQueryDataAvailable(hFile:Pointer;var NumberOfBytesAvailable:Card; flags:Card;
Context:Card) : LongBool
11225: Func BoldHttpOpenRequest(hConnect:Pointer;Verb,ObjectName,Version,Referrer:str;AcceptTypes:PCharArr;Flags,
Context:Card):Pointer
11226: Func BoldHttpSendRequest(hRequest:Ptr;Headers:str;Optional:Ptr;OptionalLength:Cardi) : LongBool
11227: Func BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11228: Func BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11229: Func BoldInternetConnect(hInet: HINTERNET;ServerName:str; nServerPort:INTERNET_PORT; Username:str;
Password :Str; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11230: Func BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11231: end;
11232:
11233: Proc SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11234: begin
11235:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11236:   SIRegister_TfrmBoldQueryUser(CL);
11237: Func QueryUser( const Title, Query :Str) : TBoldQueryResult
11238: end;
11239:
11240: (*-----*)
11241: Proc SIRegister_BoldQueue(CL: TPSPascalCompiler);
11242: begin
11243:   //('befIsInDisplayList',' BoldElementFlag0);
11244:   //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11245:   //('befFollowerSelected',' BoldElementFlag2);
11246:   FindClass('TOBJECT'),'T-BoldQueue
11247:   FindClass('TOBJECT'),'T-BoldQueueable
11248:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11249:   SIRegister_TBoldQueueable(CL);
11250:   SIRegister_TBoldQueue(CL);
11251: Func BoldQueueFinalized :Bool
11252: Func BoldInstalledQueue : TBoldQueue
11253: end;
11254:
11255: Proc SIRegister_Barcode(CL: TPSPascalCompiler);
11256: begin
11257:   const mmPerInch,'Extended).setExtended( 25.4);
11258:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,
11259:   + bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc
11260:   +Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'

```



```

11261:   +bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11262:   +odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11263: TBarLineType', '( white, black, black_half )
11264: TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11265: TShowTextPosition', '( stpTopLeft, stpTopRight, stpBottomLeft, stpBottomRight, stpBottomCenter )
11266: TChecksumMethod', '( csmNone, csmModulo10 )
11267: SIRegister_TAsBarcode(CL);
11268: Func CheckSumModulo10( const data :Str) :Str
11269: Func ConvertMmToPixelsX( const Value : Double) : Int
11270: Func ConvertMmToPixelsY( const Value : Double) : Int
11271: Func ConvertInchToPixelsX( const Value : Double) : Int
11272: Func ConvertInchToPixelsY( const Value : Double) : Int
11273: end;
11274:
11275: Proc SIRegister_Geometry(CL: TPSPascalCompiler); //OpenGL
11276: begin
11277:   TVector3f = array[0..2] of single;
11278:   CL.AddTypeS('TAffineVector', 'TVector3f
11279:   CL.AddTypeS('TVertex', 'TAffineVector
11280:   THomogeneousByteVector', 'array[0..3] of Byte
11281:   THomogeneousWordVector', 'array[0..3] of Word
11282:   THomogeneousIntVector', 'array[0..3] of Int
11283:   THomogeneousFltVector', 'array[0..3] of single
11284:   THomogeneousDblVector', 'array[0..3] of double
11285:   THomogeneousExtVector', 'array[0..3] of extended
11286:   TAffineByteVector', 'array[0..2] of Byte
11287:   TAffineWordVector', 'array[0..2] of Word
11288:   TAffineIntVector', 'array[0..2] of Int
11289:   TAffineFltVector', 'array[0..2] of single
11290:   TAffineDblVector', 'array[0..2] of double
11291:   TAffineExtVector', 'array[0..2] of extended
11292:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11293:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11294:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11295:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11296:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11297:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11298:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11299:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11300:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11301:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11302:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11303:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11304:   TMatrix4b', 'THomogeneousByteMatrix
11305:   TMatrix4w', 'THomogeneousWordMatrix
11306:   TMatrix4i', 'THomogeneousIntMatrix
11307:   TMatrix4f', 'THomogeneousFltMatrix
11308:   TMatrix4d', 'THomogeneousDblMatrix
11309:   TMatrix4e', 'THomogeneousExtMatrix
11310:   TMatrix3b', 'TAffineByteMatrix
11311:   TMatrix3w', 'TAffineWordMatrix
11312:   TMatrix3i', 'TAffineIntMatrix
11313:   TMatrix3f', 'TAffineFltMatrix
11314:   TMatrix3d', 'TAffineDblMatrix
11315:   TMatrix3e', 'TAffineExtMatrix
11316:   //'PMatrix', '^TMatrix // will not work
11317:   TMatrixGL', 'THomogeneousFltMatrix
11318:   THomogeneousMatrix', 'THomogeneousFltMatrix
11319:   TAffineMatrix', 'TAffineFltMatrix
11320:   TQuaternion', 'record Vector : TVector4f; end
11321:   TRectangle', 'record Left : Int; Top : Int; Width : integer; Height : Int; end
11322:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11323:   + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11324:   + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11325:   'EPSILON', 'Extended').setExtended( 1E-100);
11326:   'EPSILON2', 'Extended').setExtended( 1E-50);
11327:   Func VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11328:   Func VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11329:   Func VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11330:   Func VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11331:   Func VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11332:   Func VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11333:   Func VectorAngle( V1, V2 : TAffineVector) : Single
11334:   Func VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11335:   Func VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11336:   Func VectorDotProduct( V1, V2 : TVectorGL) : Single
11337:   Func VectorLength( V : array of Single) : Single
11338:   Func VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11339:   Proc VectorNegate( V : array of Single)
11340:   Func VectorNorm( V : array of Single) : Single
11341:   Func VectorNormalize( V : array of Single) : Single
11342:   Func VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11343:   Func VectorReflect( V, N : TAffineVector) : TAffineVector
11344:   Proc VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11345:   Proc VectorScale( V : array of Single; Factor : Single)
11346:   Func VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11347:   Func CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11348:   Func CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11349:   Func CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL

```

```

11350: Func CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11351: Func CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11352: Proc MatrixAdjoint( var M : TMatrixGL)
11353: Func MatrixAffineDeterminant( M : TAffineMatrix) : Single
11354: Proc MatrixAffineTranspose( var M : TAffineMatrix)
11355: Func MatrixDeterminant( M : TMatrixGL) : Single
11356: Proc MatrixInvert( var M : TMatrixGL)
11357: Func MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11358: Proc MatrixScale( var M : TMatrixGL; Factor : Single)
11359: Proc MatrixTranspose( var M : TMatrixGL)
11360: Func QuaternionConjugate( Q : TQuaternion) : TQuaternion
11361: Func QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11362: Func QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11363: Func QuaternionSlerp( QStart, QEnd: TQuaternion; Spin: Int; t: Single) : TQuaternion
11364: Func QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11365: Proc QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11366: Func ConvertRotation( Angles : TAffineVector) : TVectorGL
11367: Func CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11368: //Func MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Bool
11369: Func VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11370: Func VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11371: Func VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11372: Func MakeAffineDblVector( V : array of Double) : TAffineDblVector
11373: Func MakeDblVector( V : array of Double) : THomogeneousDblVector
11374: Func MakeAffineVector( V : array of Single) : TAffineVector
11375: Func MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11376: Func MakeVector( V : array of Single) : TVectorGL
11377: Func PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Bool
11378: Func VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11379: Func VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11380: Func VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11381: Func VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11382: Func ArcCosGL( X : Extended) : Extended
11383: Func ArcSinGL( X : Extended) : Extended
11384: Func ArcTan2GL( Y, X : Extended) : Extended
11385: Func CoTanGL( X : Extended) : Extended
11386: Func DegToRadGL( Degrees : Extended) : Extended
11387: Func RadToDegGL( Radians : Extended) : Extended
11388: Proc SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11389: Func TanGL( X : Extended) : Extended
11390: Func Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11391: Func Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single) : TMatrixGL;
11392: Func Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11393: Func Pitch1( Matrix : TMatrixGL; MasterRight: TAffineVector; Angle: Single) : TMatrixGL;
11394: Func Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11395: Func Roll1( Matrix: TMatrixGL; MasterDirection: TAffineVector; Angle: Single) : TMatrixGL;
11396: end;
11397:
11398: Proc SIRegister_JclRegistry(CL: TPSPascalCompiler);
11399: begin
11400: Func RegCreateKey( const RootKey : HKEY; const Key, Value : Str) : Longint
11401: Func RegDeleteEntry( const RootKey : HKEY; const Key, Name : Str) : Bool
11402: Func RegDeleteKeyTree( const RootKey : HKEY; const Key : Str) : Bool
11403: Func RegReadBool( const RootKey : HKEY; const Key, Name : Str) : Bool
11404: Func RegReadBoolDef( const RootKey: HKEY; const Key, Name: Str; Def: Bool) : Bool
11405: Func RegReadInt( const RootKey : HKEY; const Key, Name : Str) : Int
11406: Func RegReadIntDef( const RootKey : HKEY; const Key, Name : Str; Def : Int) : Int
11407: Func RegReadString( const RootKey : HKEY; const Key, Name : Str) : Str
11408: Func RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : Str) : Str
11409: Func RegReadDWORD( const RootKey : HKEY; const Key, Name : Str) : Int64
11410: Func RegReadDWORDDef( const RootKey: HKEY; const Key, Name : Str; Def: int64) : Int64
11411: Proc RegWriteBool( const RootKey : HKEY; const Key, Name : Str; Value : Bool)
11412: Proc RegWriteInt( const RootKey : HKEY; const Key, Name : Str; Value : Int)
11413: Proc RegWriteString( const RootKey : HKEY; const Key, Name, Value : Str)
11414: Proc RegWriteDWORD( const RootKey : HKEY; const Key, Name : Str; Value : Int64)
11415: Func RegGetValueNames( const RootKey: HKEY; const Key: Str; const List : TStrings) : Bool
11416: Func RegGetKeyNames( const RootKey: HKEY; const Key: Str; const List: TStrings) : Bool
11417: Func RegHasSubKeys( const RootKey : HKEY; const Key : Str) : Bool
11418: Func RegKeyExists( const RootKey : HKEY; const Key : Str) : Bool
11419: AddTypeS( 'TExecKind', (ekMachineRun, ekMachineRunOnce, ekUserRun, ekUserRunOnce, ekServiceRun,
ekServiceRunOnce)
11420: AddClassN( FindClass( 'OBJECT'), 'EJclRegistryError
11421: Func UnregisterAutoExec( ExecKind : TExecKind; const Name : Str) : Bool
11422: Func RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : Str) : Bool
11423: Func RegSaveList( const RootKey: HKEY; const Key: str; const ListName: str; const Items: TStrings) : Bool;
11424: Func RegLoadList( const RootKey: HKEY; const Key: str; const ListName: str; const SaveTo: TStrings) : Bool;
11425: Func RegDelList( const RootKey: HKEY; const Key: str; const ListName: str) : Bool
11426: end;
11427:
11428: Proc SIRegister_JclCOM(CL: TPSPascalCompiler);
11429: begin
11430: CLSID_StdComponentCategoriesMgr, 'TGUID '{0002E005-0000-0000-C000-000000000046}
11431: CATID_SafeForInitializing, 'TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11432: CATID_SafeForScripting, 'TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11433: icMAX_CATEGORY_DESC_LEN, 'LongInt'( 128);
11434: FindClass( 'OBJECT'), 'EInvalidParam
11435: Func IsDCOMInstalled : Bool
11436: Func IsDCOMEnabled : Bool
11437: Func GetDCOMVersion : Str

```

```

11438: Func GetMDACVersion :Str
11439: Func GetMDACVersion2:Str
11440: Func MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11441: Func MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11442: Func MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11443: Func MarshalInterMachineInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11444: Func MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var VarArray:OleVariant):HResult;
11445: Func CreateComponentCategory( const CatID:TGUID; const sDescription:str): HResult
11446: Func RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID): HResult
11447: Func UnRegisterCLSIDInCategory(const ClassID: TGUID; const CatID : TGUID): HResult
11448: Func ResetIStreamToStart( Stream : IStream) :Bool
11449: Func SizeOfIStreamContents( Stream : IStream) : Largeint
11450: Func StreamToVariantArray( Stream : TStream) : OleVariant;
11451: Func StreamToVariantArray1( Stream : IStream) : OleVariant;
11452: Proc VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11453: Proc VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11454: end;
11455:
11456: Proc SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11457: begin
11458:   Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11459:   FahrenheitFreezingPoint','Extended).setExtended( 32.0);
11460:   KelvinFreezingPoint','Extended).setExtended( 273.15);
11461:   CelsiusAbsoluteZero','Extended).setExtended( - 273.15);
11462:   FahrenheitAbsoluteZero','Extended).setExtended( - 459.67);
11463:   KelvinAbsoluteZero','Extended).setExtended( 0.0);
11464:   DegPerCycle','Extended).setExtended( 360.0);
11465:   DegPerGrad','Extended).setExtended( 0.9);
11466:   DegPerRad','Extended).setExtended( 57.295779513082320876798154814105);
11467:   GradPerCycle','Extended).setExtended( 400.0);
11468:   GradPerDeg','Extended).setExtended( 1.11111111111111111111111111111111);
11469:   GradPerRad','Extended).setExtended( 63.661977236758134307553505349006);
11470:   RadPerCycle','Extended).setExtended( 6.283185307179586476925286766559);
11471:   RadPerDeg','Extended).setExtended( 0.017453292519943295769236907684886);
11472:   RadPerGrad','Extended).setExtended( 0.015707963267948966192313216916398);
11473:   CyclePerDeg','Extended).setExtended( 0.0027777777777777777777777777777778);
11474:   CyclePerGrad','Extended).setExtended( 0.0025);
11475:   CyclePerRad','Extended).setExtended( 0.15915494309189533576888376337251);
11476:   ArcMinutesPerDeg','Extended).setExtended( 60.0);
11477:   ArcSecondsPerArcMinute','Extended).setExtended( 60.0);
11478: Func HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11479: Func MakePercentage( const Step, Max : Longint) : Longint
11480: Func CelsiusToKelvin( const T : double) : double
11481: Func CelsiusToFahrenheit( const T : double) : double
11482: Func KelvinToCelsius( const T : double) : double
11483: Func KelvinToFahrenheit( const T : double) : double
11484: Func FahrenheitToCelsius( const T : double) : double
11485: Func FahrenheitToKelvin( const T : double) : double
11486: Func CycleToDeg( const Cycles : double) : double
11487: Func CycleToGrad( const Cycles : double) : double
11488: Func CycleToRad( const Cycles : double) : double
11489: Func DegToCycle( const Degrees : double) : double
11490: Func DegToGrad( const Degrees : double) : double
11491: Func DegToRad( const Degrees : double) : double
11492: Func GradToCycle( const Grads : double) : double
11493: Func GradToDeg( const Grads : double) : double
11494: Func GradToRad( const Grads : double) : double
11495: Func RadToCycle( const Radians : double) : double
11496: Func RadToDeg( const Radians : double) : double
11497: Func RadToGrad( const Radians : double) : double
11498: Func DmsToDeg( const D, M : Int; const S : double) : double
11499: Func DmsToRad( const D, M : Int; const S : double) : double
11500: Proc DegToDms( const Degrees : double; out D, M : Int; out S : double)
11501: Func DegToDmsStr( const Degrees : double; const SecondPrecision :Card):Str
11502: Proc CartesianToPolar( const X, Y : double; out R, Phi : double)
11503: Proc PolarToCartesian( const R, Phi : double; out X, Y : double)
11504: Proc CartesianToCylindric( const X, Y, Z : double; out R, Phi, Zeta : double)
11505: Proc CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11506: Proc CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11507: Proc SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11508: Func CmToInch( const Cm : double) : double
11509: Func InchToCm( const Inch : double) : double
11510: Func FeetToMetre(const Feet : double) : double
11511: Func MetreToFeet(const Metre : double) : double
11512: Func YardToMetre(const Yard : double) : double
11513: Func MetreToYard(const Metre : double) : double
11514: Func NmToKm(const Nm : double) : double
11515: Func KmToNm(const Km : double) : double
11516: Func KmToSm(const Km : double) : double
11517: Func SmToKm(const Sm : double) : double
11518: Func LitreToGalUs( const Litre : double) : double
11519: Func GalUsToLitre( const GalUs : double) : double
11520: Func GalUsToGalCan( const GalUs : double) : double
11521: Func GalCanToGalUs( const GalCan : double) : double
11522: Func GalUsToGalUk( const GalUs : double) : double
11523: Func GalUkToGalUs( const GalUk : double) : double
11524: Func LitreToGalCan( const Litre : double) : double
11525: Func GalCanToLitre( const GalCan : double) : double
11526: Func LitreToGalUk( const Litre : double) : double

```

```

11527: Func GalUkToLitre( const GalUk : double) : double
11528: Func KgToLb( const Kg : double) : double
11529: Func LbToKg( const Lb : double) : double
11530: Func KgToOz( const Kg : double) : double
11531: Func OzToKg( const Oz : double) : double
11532: Func CwtUsToKg(const Cwt : double) : double
11533: Func CwtUkToKg(const Cwt : double) : double
11534: Func KaratToKg(const Karat : double) : double
11535: Func KgToCwtUs(const Kg : double) : double
11536: Func KgToCwtUk(const Kg : double) : double
11537: Func KgToKarat(const Kg : double) : double
11538: Func KgToSton(const Kg : double) : double
11539: Func KgToLton(const Kg : double) : double
11540: Func StonToKg(const STon : double) : double
11541: Func LtonToKg(const Lton : double) : double
11542: Func QrUsToKg(const Qr : double) : double
11543: Func QrUkToKg(const Qr : double) : double
11544: Func KgToQrUs(const Kg : double) : double
11545: Func KgToQrUk(const Kg : double) : double
11546: Func PascalToBar( const Pa : double) : double
11547: Func PascalToAt( const Pa : double) : double
11548: Func PascalToTorr( const Pa : double) : double
11549: Func BarToPascal( const Bar : double) : double
11550: Func AtToPascal( const At : double) : double
11551: Func TorrToPascal( const Torr : double) : double
11552: Func KnotToMs( const Knot : double) : double
11553: Func HpElectricToWatt( const HpE : double) : double
11554: Func HpMetricToWatt( const HpM : double) : double
11555: Func MsToKnot( const ms : double) : double
11556: Func WattToHpElectric( const W : double) : double
11557: Func WattToHpMetric( const W : double) : double
11558: Func getBigPI:Str; //PI of 1000 numbers
11559:
11560: Proc SIRegister_devcutils(CL: TPSPascalCompiler);
11561: begin
11562: Func CDExecuteFile( const FileName, Params, DefaultDir :Str; ShowCmd : Int) : THandle
11563: Proc CDCopyFile( const FileName, DestName :Str)
11564: Proc CDMoveFile( const FileName, DestName :Str)
11565: Func MakeCommaTextToColor( Text :Str; Index : Int; DefaultColor : TColor) : TColor
11566: Proc CDDeleteFiles( Sender : TObject; s :Str)
11567: Func CDGetTempDir :Str
11568: Func CDGetFileSize( FileName :Str) : longint
11569: Func GetFileTime( FileName :Str) : longint
11570: Func GetShortName( FileName :Str) :Str
11571: Func GetFullName( FileName :Str) :Str
11572: Func WinReboot :Bool
11573: Func WinDir :Str
11574: Func RunFile( FileToRun :Str; Params :Str; Dir :Str; Wait :Bool) :Card
11575: Func RunFile_( Cmd, WorkDir :Str; Wait :Bool) :Bool
11576: Func devExecutor : TdevExecutor
11577: end;
11578:
11579: Proc SIRegister_FileAssocs(CL: TPSPascalCompiler);
11580: begin
11581: Proc CheckAssociations // AssociationsCount', 'LongInt'( 7);
11582: Proc Associate( Index : Int)
11583: Proc UnAssociate( Index : Int)
11584: Func IsAssociated( Index : Int) :Bool
11585: Func CheckFiletype(const extension,filetype,description, verb,serverapp:str):Bool
11586: Proc RegisterFiletype(const extension,filetype,description,verb,serverapp,IcoNum:str)
11587: Proc RegisterDDEServer( const filetype, verb, topic, servername, macro :Str)
11588: Proc RefreshIcons;
11589: Func GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11590: Func MergColor(Colors: Array of TColor): TColor;
11591: Func NewColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11592: Proc DimBitmap(ABitmap: TBitmap; Value: Int);
11593: Func GrayColor(ACanvas: TCanvas; clr: TColor; Value: Int): TColor;
11594: Func GetInverseColor(AColor: TColor): TColor;
11595: Proc GrayBitmap(ABitmap: TBitmap; Value: Int);
11596: Proc DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: Int;ShadowColor: TColor);
11597: Proc DrawCheckMark(ACanvas: TCanvas; X, Y: Int);
11598: Proc GetSystemMenuFont(Font: TFont);
11599: end;
11600:
11601: //*****unit uPSI_JvHLPArser;*****
11602: Func IsStringConstant(const St:Str):Bool;
11603: Func IsIntConstant(const St:Str):Bool;
11604: Func IsRealConstant(const St:Str):Bool;
11605: Func IsIdentifier(const ID:Str):Bool;
11606: Func GetStringValue(const St:Str):Str;
11607: Proc ParseString(const S:Str; Ss: TStrings);
11608: Func IsStringConstantW(const St: WideString):Bool;
11609: Func IsIntConstantW(const St: WideString):Bool;
11610: Func IsRealConstantW(const St: WideString):Bool;
11611: Func IsIdentifierW(const ID: WideString):Bool;
11612: Func GetStringValueW(const St: WideString): WideString;
11613: Proc ParseStringW(const S: WideString; Ss: TStrings);
11614:
11615:

```

```

11616: //*****unit uPSI_JclMapi;*****
11617: Func JclSimpleSendMail( const ARecipient, AName, ASubject, ABody :Str; const AAttachment : TFileName;
ShowDialog :Bool; AParentWND : HWND) :Bool
11618: Func JclSimpleSendFax( const ARecipient, AName, ASubject, ABody:Str;const AAttachment: TFileName; ShowDialog
:Bool; AParentWND : HWND) :Bool
11619: Func JclSimpleBringUpSendMailDialog(const ASubject, ABody:Str;const AAttach:TFileName;AParentWND:HWND):Bool
11620: Func MapiCheck( const Res : DWORD; IgnoreUserAbort :Bool) : DWORD
11621: Func MapiErrorMessage( const ErrorCode : DWORD) :Str
11622:
11623: Proc SIRegister_IdNTLM(CL: TPSPascalCompiler);
11624: begin
11625:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11626:   Func BuildType1Message( ADomain, AHost :Str) :Str
11627:   Func BuildType3Message( ADomain, AHost, AUsername:WideString;APassword, ANonce:Str):Str
11628:   Proc RegisterAuthenticationMethod(MethodName:Str; AuthClass:TIdAuthenticationClass)
11629:   Func FindAuthClass( AuthName :Str) : TIdAuthenticationClass
11630:   GBase64CodeTable, 'string' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11631:   GXHECodeTable, 'string' '+-0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11632:   GUUECodeTable, 'string' '!\"#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_
11633: end;
11634:
11635: Proc SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11636: begin
11637:   ('IpAny', 'LongWord').SetUInt( $00000000);
11638:   IpLoopBack, 'LongWord').SetUInt( $7F000001);
11639:   IpBroadcast, 'LongWord').SetUInt( $FFFFFFF);
11640:   IpNone, 'LongWord').SetUInt( $FFFFFFF);
11641:   PortAny, 'LongWord( $0000);
11642:   SocketMaxConnections', 'LongInt'( 5);
11643:   TIpAddr', 'LongWord
11644:   TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4:Byte; end
11645:   Func HostToNetLong( HostLong : LongWord) : LongWord
11646:   Func HostToNetShort( HostShort : Word) : Word
11647:   Func NetToHostLong( NetLong : LongWord) : LongWord
11648:   Func NetToHostShort( NetShort : Word) : Word
11649:   Func StrToIp( Ip :Str) : TIpAddr
11650:   Func IpToStr( Ip : TIpAddr) :Str
11651: end;
11652:
11653: (*-----*)
11654: Proc SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11655: begin
11656:   TAlsmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11657:   +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11658:   +'amShal, AlsmtpClientAuthAutoSelect )
11659:   TAlsmtpClientAuthTypeSet', 'set of TAlsmtpClientAuthType
11660:   SIRegister_TAlsmtpClient(CL);
11661: end;
11662:
11663: Proc SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11664: begin
11665:   TBitNo', 'IntTStByteNo', 'IntTStationNo', 'IntTInOutNo', 'IntTIO', '(EE, AA, NE, NA )
11666:   TBitSet', 'set of TBitNo
11667:   TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11668:   TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11669:   TBitAddr', 'LongInt
11670:   TByteAddrRec', 'record Kind : TAddrKind; ByteNo: Byte; end
11671:   TByteAddr', 'SmallInt
11672:   TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11673:   Func BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11674:   Func BusBitAddr(aIo:TIO; aInOutNo:TInOutNo; aStat:TStatNo; aStByteNo:TStByteNo; aBitNo:TBitNo):TBitAddr;
11675:   Proc BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var
aBitNo:TBitNo);
11676:   Func BitAddrToStr( Value : TBitAddr) :Str
11677:   Func StrToBitAddr( const Value :Str) : TBitAddr
11678:   Func ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11679:   Func BusByteAddr(aIo:TIO; aInOutNo:TInOutNo; aStation:TStationNo; aStByteNo:TStByteNo):TByteAddr
11680:   Proc ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11681:   Func ByteAddrToStr( Value : TByteAddr) :Str
11682:   Func StrToByteAddr( const Value :Str) : TByteAddr
11683:   Proc IncByteAddr( var ByteAddr : TByteAddr; Increment : Int)
11684:   Proc DecByteAddr( var ByteAddr : TByteAddr; Decrement : Int)
11685:   Func InOutStateToStr( State : TInOutState) :Str
11686:   Func MasterErrorToStr( ErrorCode : TErrorCode) :Str
11687:   Func SlaveErrorToStr( ErrorCode : TErrorCode) :Str
11688: end;
11689:
11690: Proc SIRegister_WDosTimers(CL: TPSPascalCompiler);
11691: begin
11692:   TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11693:   +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11694:   DpmiPmVector', 'Int64
11695:   'DInterval', 'LongInt'( 1000);
11696:   //'DEnabled', 'Boolean')BoolToStr( True);
11697:   'DIntFreq', 'string' if64
11698:   //'DMessages', 'Boolean if64);
11699:   SIRegister_TwdxCustomTimer(CL);
11700:   SIRegister_TwdxTimer(CL);
11701:   SIRegister_TwdxRtcTimer(CL);

```



```

11702: SIRegister_TCustomIntTimer(CL);
11703: SIRegister_TIntTimer(CL);
11704: SIRegister_TRtcIntTimer(CL);
11705: Func RealNow : TDateTime
11706: Func MsToDateTime( MilliSecond : LongInt ) : TDateTime
11707: Func DateTimeToMs( Time : TDateTime ) : LongInt
11708: end;
11709:
11710: Proc SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11711: begin
11712: TIdSyslogPRI', 'Int
11713: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11714:   + 'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11715:   + 'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11716:   + 'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11717:   + 'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11718: TIdSyslogSeverity', '(slEmergency, slAlert, slCritical, slError, slWarning, slNotice, slInformational, slDebug)
11719: SIRegister_TIdSysLogMsgPart(CL);
11720: SIRegister_TIdSysLogMessage(CL);
11721: Func FacilityToString( AFac : TIdSyslogFacility) :Str
11722: Func SeverityToString( ASec : TIdSyslogSeverity) :Str
11723: Func NoToSeverity( ASev : Word) : TIdSyslogSeverity
11724: Func logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11725: Func NoToFacility( AFac : Word) : TIdSyslogFacility
11726: Func logFacilityToNo( AFac : TIdSyslogFacility) : Word
11727: end;
11728:
11729: Proc SIRegister_TextUtils(CL: TPSPascalCompiler);
11730: begin
11731: 'UWhitespace', 'String' '(?:\s*)
11732: Func StripSpaces( const AText :Str) :Str
11733: Func CharCount( const AText :Str; Ch : Char) : Int
11734: Func BalancedText(const AText:Str; const Ch1,Ch2: Char; const Count: Int) :Str
11735: Func BalancedTextReg( const AText:str;const Ch1, Ch2 : Char; const Count :Int):str
11736: end;
11737:
11738: Proc SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11739: begin
11740: ExtPascalVersion', 'String' '0.9.8
11741: AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11742:   + 'Opera, brKonqueror, brMobileSafari )
11743: AddTypeS('TCSSUnit', (cssPX,cssPerc,cssEM,cssEX,cssIN,cssCM,cssMM,cssPT, cssPC, cssnone )
11744: AddTypeS('TextProcedure', 'Procedure
11745: Func DetermineBrowser( const UserAgentStr :Str) : TBrowser
11746: Func ExtExtract(const Delims:array of string;var S:str;var Matches:TStringList;Remove:bool):bool;
11747: Func ExtExplode( Delim : char; const S :Str; Separator : char) : TStringList
11748: Func FirstDelimiter( const Delimiters, S :Str; Offset : Int) : Int
11749: Func RPosEx( const Substr, Str :Str; Offset : Int) : Int
11750: Func CountStr( const Substr, Str :Str; UntilStr :Str) : Int
11751: Func StrToJS( const S :Str; UseBR :Bool) :Str
11752: Func CaseOf( const S :Str; const Cases : array of string) : Int
11753: Func RCaseOf( const S :Str; const Cases : array of string) : Int
11754: Func EnumToJSSString( TypeInfo : PTypeInfo; Value : Int) :Str
11755: Func SetPaddings(Top:Int;Right:int;Bottom:intr;Left:int;CSSUnit:TCSSUnit;Header:bool):str;
11756: Func SetMargins(Top:Int;Right:int;Bottom:intr;Left:intr;CSSUnit:TCSSUnit;Header:bool):str;
11757: Func ExtBefore( const BeforeS, AfterS, S :Str) :Bool
11758: Func IsUpperCase( S :Str) :Bool
11759: Func BeautifyJS(const AScript:str;const StartingLevel:Int;SplitHTMLNewLine:bool):str;
11760: Func BeautifyCSS( const AStyle :Str) :Str
11761: Func LengthRegExp( Rex :Str; CountAll :Bool) : Int
11762: Func JSDateToDateTime( JSDate :Str) : TDateTime
11763: end;
11764:
11765: Proc SIRegister_JclShell(CL: TPSPascalCompiler);
11766: begin
11767: TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11768: TSHDeleteOptions', 'set of TSHDeleteOption
11769: TSHRenameOption', '( roSilent, roRenameOnCollision )
11770: TSHRenameOptions', 'set of TSHRenameOption
11771: Func SHDeleteFiles(Parent:HWND;const Files:str;Options:TSHDeleteOptions):Boolean
11772: Func SHDeleteFolder(Parent:HWND; const Folder:str; Options: TSHDeleteOptions): Bool
11773: Func SHRenameFile( const Src, Dest :Str; Options : TSHRenameOptions) :Bool
11774: TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11775: TEnumFolderFlags', 'set of TEnumFolderFlag
11776: TEnumFolderRec', 'record DisplayName:str; Attributes:DWORD; IconLarge:HICON; IconSmall:HICON;
Item:PItemIdList;EnumIdList:IEnumIdList;Folder:IShellFolder;end
11777: Func SHEnumFolderFirst(const Folder:str;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11778: Func SHEnumSpecialFolderFirst(SpecFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11779: Proc SHEnumFolderClose( var F : TEnumFolderRec)
11780: Func SHEnumFolderNext( var F : TEnumFolderRec) :Bool
11781: Func GetSpecialFolderLocation( const Folder : Int) :Str
11782: Func DisplayPropDialog( const Handle : HWND; const FileName :Str) :Bool;
11783: Func DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) :Bool;
11784: Func DisplayContextMenu(const Handle:HWND; const FileName:Str; Pos:TPoint):Bool
11785: Func OpenFolder( const Path :Str; Parent : HWND) :Bool
11786: Func OpenSpecialFolder( FolderID : Int; Parent : HWND) :Bool
11787: Func SHReallocMem( var P : Pointer; Count : Int) :Bool
11788: Func SHAllocMem( out P : Pointer; Count : Int) :Bool
11789: Func SHGetMem( var P : Pointer; Count : Int) :Bool

```

```

11790: Func SHFreeMem( var P : Pointer ) :Bool
11791: Func DriveToPidlBind( const DriveName :Str; out Folder : IShellFolder ) : PitemIdList
11792: Func PathToPidl( const Path :Str; Folder : IShellFolder ) : PitemIdList
11793: Func PathToPidlBind( const FileName :Str; out Folder : IShellFolder ) : PitemIdList
11794: Func PidlBindToParent( const IdList:PitemIdList;out Folder:IShellFolder;out Last:PitemIdList):Bool;
11795: Func PidlCompare( const Pidl1, Pidl2 : PitemIdList ) :Bool
11796: Func PidlCopy( const Source : PitemIdList; out Dest : PitemIdList ) :Bool
11797: Func PidlFree( var IdList : PitemIdList ) :Bool
11798: Func PidlGetDepth( const Pidl : PitemIdList ) : Int
11799: Func PidlGetLength( const Pidl : PitemIdList ) : Int
11800: Func PidlGetNext( const Pidl : PitemIdList ) : PitemIdList
11801: Func PidlToPath( IdList : PitemIdList ) :Str
11802: Func StrRetFreeMem( StrRet : TStrRet ) :Bool
11803: Func StrRetToString( IdList : PitemIdList; StrRet : TStrRet; Free :Bool) :Str
11804: PShellLink', '^TShellLink // will not work
11805: TShellLink', 'record Arguments :Str; ShowCmd : Int; Work'
11806: +ingDirectory :Str; IdList : PitemIDList; Target :Str; Description '
11807: +:Str; IconLocation :Str; IconIndex : Int; HotKey : Word; end
11808: Proc ShellLinkFree( var Link : TShellLink)
11809: Func ShellLinkResolve( const FileName :Str; var Link : TShellLink ) : HRESULT
11810: Func ShellLinkCreate( const Link : TShellLink; const FileName :Str ) : HRESULT
11811: Func ShellLinkCreateSystem( const Link:TShellLink;const Folder:Int;const FileName:str):HRESULT;
11812: Func ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon ) :Bool
11813: Func SHDllGetVersion( const FileName :Str; var Version : TDllVersionInfo ) :Bool
11814: Func GetSystemIcon( IconIndex : Int; Flags :Card ) : HICON
11815: Func OverlayIcon( var Icon : HICON; Overlay : HICON; Large :Bool ) :Bool
11816: Func OverlayIconShortCut( var Large, Small : HICON ) :Bool
11817: Func OverlayIconShared( var Large, Small : HICON ) :Bool
11818: Func SHGetItemInfoTip( const Folder : IShellFolder; Item : PitemIdList ) :Str
11819: Func ShellExecEx(const FileName:str;const Paramtrs:str;const Verb:str;CmdShow:Int):Bool;
11820: Func ShellExec(Wnd:Int;const Operati,FileNam,Paramtrs,Directy:str;ShowCommand:Int):Bool;
11821: Func ShellExecAndWait(const FileName:str;const Paramtrs:str;const Verb:str;CmdShow:Int):Bool;
11822: Func ShellOpenAs( const FileName :Str ) :Bool
11823: Func ShellRasDial( const EntryName :Str ) :Bool
11824: Func ShellRunControlPanel( const NameOrFileName:str; AppletNumber:Int):Boolean
11825: Func GetFileNameIcon( const FileName :Str; Flags :Card ) : HICON
11826: TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11827: Func GetFileExeType( const FileName : TFileName ) : TJclFileExeType
11828: Func ShellFindExecutable( const FileName, DefaultDir :Str ) :Str
11829: Proc keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11830: Func OemKeyScan( wOemChar : Word ) : DWORD
11831: Proc mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11832: end;
11833:
11834: Proc SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11835: begin
11836: xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11837: //Func xmlValidChar( const Ch : AnsiChar ) :Bool;
11838: Func xmlValidChar1( const Ch : UCS4Char ) :Bool;
11839: Func xmlValidChar2( const Ch : WideChar ) :Bool;
11840: Func xmlIsSpaceChar( const Ch : WideChar ) :Bool
11841: Func xmlIsLetter( const Ch : WideChar ) :Bool
11842: Func xmlIsDigit( const Ch : WideChar ) :Bool
11843: Func xmlIsNameStartChar( const Ch : WideChar ) :Bool
11844: Func xmlIsNameChar( const Ch : WideChar ) :Bool
11845: Func xmlIsPubidChar( const Ch : WideChar ) :Bool
11846: Func xmlValidName( const Text : UnicodeString ) :Bool
11847: //xmlSpace','Char #$20 or #$9 or #$D or #$A);
11848: //Func xmlSkipSpace( var P : PWideChar ) :Bool
11849: //Func xmlSkipEq( var P : PWideChar ) :Bool
11850: //Func xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString ) :Bool
11851: //Func xmlGetEntityEncoding(const Buf:Pointer;const BufSize:Int;out HeaderSize:Int):TUnicodeCodecClass
11852: Func xmlResolveEntityReference( const RefName : UnicodeString ) : WideChar
11853: Func xmlTag( const Tag : UnicodeString ) : UnicodeString
11854: Func xmlEndTag( const Tag : UnicodeString ) : UnicodeString
11855: Func xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString ) : UnicodeString
11856: Func xmlEmptyTag( const Tag, Attr : UnicodeString ) : UnicodeString
11857: Proc xmlSafeTextInPlace( var Txt : UnicodeString)
11858: Func xmlSafeText( const Txt : UnicodeString ) : UnicodeString
11859: Func xmlSpaceIndent( const IndentLength : Int; const IndentLevel : Int):UnicodeString
11860: Func xmlTabIndent( const IndentLevel : Int ) : UnicodeString
11861: Func xmlComment( const Comment : UnicodeString ) : UnicodeString
11862: Proc SelfTestcXMLFunctions
11863: end;
11864:
11865: (*-----*)
11866: Proc SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11867: begin
11868: Func AwaitCursor : IUnknown
11869: Func ChangeCursor( NewCursor : TCursor ) : IUnknown
11870: Proc SuspendRedraw( AControl : TWinControl; Suspend :Bool)
11871: Func YesNo( const ACaption, AMsg :Str ) :Bool
11872: Proc strTokenize( const S :Str; Delims : TSysCharSet; Results : TStrings)
11873: Func GetBorlandLibPath( Version : Int; ForDelphi :Bool ) :Str
11874: Func GetExpandedLibRoot( Version : Int; ForDelphi :Bool ) :Str
11875: Proc GetPathList( Version : Int; ForDelphi :Bool; Strings : TStrings)
11876: Proc GetSystemPaths( Strings : TStrings)
11877: Proc MakeEditNumeric( EditHandle : Int)
11878: end;

```

```

11879:
11880: Proc SIRegister_yuvconverts(CL: TPSPascalCompiler);
11881: begin
11882:   AddTypes('TVideoCodec', '(vcUnknown,vcRGB,vcUYU2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11883:   'BI_YUY2','LongWord($32595559);
11884:   'BI_UYVY','LongWord').SetUInt($59565955);
11885:   'BI_BTUV','LongWord').SetUInt($50313459);
11886:   'BI_YVU9','LongWord').SetUInt($39555659);
11887:   'BI_YUV12','LongWord($30323449);
11888:   'BI_Y8','LongWord').SetUInt($20203859);
11889:   'BI_Y211','LongWord').SetUInt($31313259);
11890:   Func BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11891:   Func ConvertCodecToRGB(CoDec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Int):Bool;
11892: end;
11893:
11894: (*-----*)
11895: Proc SIRegister_AviCap(CL: TPSPascalCompiler);
11896: begin
11897:   'WM_USER','LongWord').SetUInt($0400);
11898:   'WM_CAP_START','LongWord').SetUInt($0400);
11899:   'WM_CAP_END','LongWord').SetUInt($0400+85);
11900:   //WM_CAP_START+ 85
11901:   // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11902:   Func capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11903:   Func capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11904:   Func capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11905:   Func capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11906:   Func capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11907:   Func capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11908:   Func capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11909:   Func capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11910:   Func capGetUserData( hwnd : THandle) : LongInt
11911:   Func capDriverConnect( hwnd : THandle; I : Word) : LongInt
11912:   Func capDriverDisconnect( hwnd : THandle) : LongInt
11913:   Func capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11914:   Func capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11915:   Func capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11916:   Func capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11917:   Func capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11918:   Func capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11919:   Func capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11920:   Func capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11921:   Func capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11922:   Func capEditCopy( hwnd : THandle) : LongInt
11923:   Func capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11924:   Func capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11925:   Func capGetAudioFormatSize( hwnd : THandle) : LongInt
11926:   Func capDlgVideoFormat( hwnd : THandle) : LongInt
11927:   Func capDlgVideoSource( hwnd : THandle) : LongInt
11928:   Func capDlgVideoDisplay( hwnd : THandle) : LongInt
11929:   Func capDlgVideoCompression( hwnd : THandle) : LongInt
11930:   Func capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11931:   Func capGetVideoFormatSize( hwnd : THandle) : LongInt
11932:   Func capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11933:   Func capPreview( hwnd : THandle; f : Word) : LongInt
11934:   Func capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11935:   Func capOverlay( hwnd : THandle; f : Word) : LongInt
11936:   Func capPreviewScale( hwnd : THandle; f : Word) : LongInt
11937:   Func capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11938:   Func capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11939:   Func capGrabFrame( hwnd : THandle) : LongInt
11940:   Func capGrabFrameNoStop( hwnd : THandle) : LongInt
11941:   Func capCaptureSequence( hwnd : THandle) : LongInt
11942:   Func capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11943:   Func capCaptureStop( hwnd : THandle) : LongInt
11944:   Func capCaptureAbort( hwnd : THandle) : LongInt
11945:   Func capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11946:   Func capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11947:   Func capCaptureSingleFrame( hwnd : THandle) : LongInt
11948:   Func capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11949:   Func capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11950:   Func capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11951:   Func capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11952:   Func capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11953:   Func capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11954:   Func capPalettePaste( hwnd : THandle) : LongInt
11955:   Func capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11956:   Func capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11957:   //PCapDriverCaps, '^TCapDriverCaps // will not work
11958:   TCapDriverCaps, 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11959:   +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla
11960:   +'; y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid
11961:   +'; eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11962:   //PCapStatus, '^TCapStatus // will not work
11963:   TCapStatus, 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11964:   +'; fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT
11965:   +'; T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO
11966:   +'; OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu
11967:   +'; rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'

```

```

11968:   +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11969:   +' wNumAudioAllocated : WORD; end
11970: //PCaptureParms', '^TCaptureParms // will not work
11971: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11972: +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11973: +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11974: +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11975: +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11976: +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11977: +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11978: +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11979: +'he : BOOL; AVStreamMaster : WORD; end
11980: // PCapInfoChunk', '^TCapInfoChunk // will not work
11981: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11982: 'CONTROLCALLBACK_PREROLL', 'LongInt' ( 1);
11983: 'CONTROLCALLBACK_CAPTURING', 'LongInt' ( 2);
11984: Func capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Int; nWidth, nHeight : Int;
hwndParent : THandle; nID : Int) : THandle
11985: Func capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Int;lpszVer:PChar;cbVer:Int):Bool;
11986: 'IDS_CAP_BEGIN', 'LongInt' ( 300);
11987: 'IDS_CAP_END', 'LongInt' ( 301);
11988: 'IDS_CAP_INFO', 'LongInt' ( 401);
11989: 'IDS_CAP_OUTOFMEM', 'LongInt' ( 402);
11990: 'IDS_CAP_FILEEXISTS', 'LongInt' ( 403);
11991: 'IDS_CAP_ERRORPALOPEN', 'LongInt' ( 404);
11992: 'IDS_CAP_ERRORPALSAVE', 'LongInt' ( 405);
11993: 'IDS_CAP_ERRORDIBSAVE', 'LongInt' ( 406);
11994: 'IDS_CAP_DEFAVIEXT', 'LongInt' ( 407);
11995: 'IDS_CAP_DEFPALEXT', 'LongInt' ( 408);
11996: 'IDS_CAP_CANTOPEN', 'LongInt' ( 409);
11997: 'IDS_CAP_SEQ_MSGSTART', 'LongInt' ( 410);
11998: 'IDS_CAP_SEQ_MSGSTOP', 'LongInt' ( 411);
11999: 'IDS_CAP_VIDEDITERR', 'LongInt' ( 412);
12000: 'IDS_CAP_READONLYFILE', 'LongInt' ( 413);
12001: 'IDS_CAP_WRITEERROR', 'LongInt' ( 414);
12002: 'IDS_CAP_NODISKSPACE', 'LongInt' ( 415);
12003: 'IDS_CAP_SETFILESIZE', 'LongInt' ( 416);
12004: 'IDS_CAP_SAVEASPERCENT', 'LongInt' ( 417);
12005: 'IDS_CAP_DRIVER_ERROR', 'LongInt' ( 418);
12006: 'IDS_CAP_WAVE_OPEN_ERROR', 'LongInt' ( 419);
12007: 'IDS_CAP_WAVE_ALLOC_ERROR', 'LongInt' ( 420);
12008: 'IDS_CAP_WAVE_PREPARE_ERROR', 'LongInt' ( 421);
12009: 'IDS_CAP_WAVE_ADD_ERROR', 'LongInt' ( 422);
12010: 'IDS_CAP_WAVE_SIZE_ERROR', 'LongInt' ( 423);
12011: 'IDS_CAP_VIDEO_OPEN_ERROR', 'LongInt' ( 424);
12012: 'IDS_CAP_VIDEO_ALLOC_ERROR', 'LongInt' ( 425);
12013: 'IDS_CAP_VIDEO_PREPARE_ERROR', 'LongInt' ( 426);
12014: 'IDS_CAP_VIDEO_ADD_ERROR', 'LongInt' ( 427);
12015: 'IDS_CAP_VIDEO_SIZE_ERROR', 'LongInt' ( 428);
12016: 'IDS_CAP_FILE_OPEN_ERROR', 'LongInt' ( 429);
12017: 'IDS_CAP_FILE_WRITE_ERROR', 'LongInt' ( 430);
12018: 'IDS_CAP_RECORDING_ERROR', 'LongInt' ( 431);
12019: 'IDS_CAP_RECORDING_ERROR2', 'LongInt' ( 432);
12020: 'IDS_CAP_AVI_INIT_ERROR', 'LongInt' ( 433);
12021: 'IDS_CAP_NO_FRAME_CAP_ERROR', 'LongInt' ( 434);
12022: 'IDS_CAP_NO_PALETTE_WARN', 'LongInt' ( 435);
12023: 'IDS_CAP_MCI_CONTROL_ERROR', 'LongInt' ( 436);
12024: 'IDS_CAP_MCI_CANT_STEP_ERROR', 'LongInt' ( 437);
12025: 'IDS_CAP_NO_AUDIO_CAP_ERROR', 'LongInt' ( 438);
12026: 'IDS_CAP_AVI_DRAWDIB_ERROR', 'LongInt' ( 439);
12027: 'IDS_CAP_COMPRESSOR_ERROR', 'LongInt' ( 440);
12028: 'IDS_CAP_AUDIO_DROP_ERROR', 'LongInt' ( 441);
12029: 'IDS_CAP_STAT LIVE_MODE', 'LongInt' ( 500);
12030: 'IDS_CAP_STAT_OVERLAY_MODE', 'LongInt' ( 501);
12031: 'IDS_CAP_STAT_CAP_INIT', 'LongInt' ( 502);
12032: 'IDS_CAP_STAT_CAP_FINI', 'LongInt' ( 503);
12033: 'IDS_CAP_STAT_PALETTE_BUILD', 'LongInt' ( 504);
12034: 'IDS_CAP_STAT_OPTPAL_BUILD', 'LongInt' ( 505);
12035: 'IDS_CAP_STAT_I_FRAMES', 'LongInt' ( 506);
12036: 'IDS_CAP_STAT_L_FRAMES', 'LongInt' ( 507);
12037: 'IDS_CAP_STAT_CAP_L_FRAMES', 'LongInt' ( 508);
12038: 'IDS_CAP_STAT_CAP_AUDIO', 'LongInt' ( 509);
12039: 'IDS_CAP_STAT VIDEOCURRENT', 'LongInt' ( 510);
12040: 'IDS_CAP_STAT_VIDEOAUDIO', 'LongInt' ( 511);
12041: 'IDS_CAP_STAT VIDEOONLY', 'LongInt' ( 512);
12042: 'IDS_CAP_STAT FRAMESDROPPED', 'LongInt' ( 513);
12043: 'AVICAP32', 'String' 'AVICAP32.dll
12044: end;
12045:
12046: Proc SRegister_ALFcmMisc(CL: TPSPascalCompiler);
12047: begin
12048: Func AlBoolToInt( Value :Bool) : Int
12049: Func ALMediumPos( LTotal, LBorder, LObject : Int) : Int
12050: Func ALIsValidEmail( const Value : Ansistr) : Bool
12051: Func ALLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
12052: Func ALInc( var x : Int; Count : Int) : Int
12053: Func ALCopyStr(const aSourceString: Ansistr; aStart, aLength: Int): Ansistr
12054: Func ALGetStringFromFile(filename:Ansistr;const ShareMode:Word= fmShareDenyWrite):Ansistr;
12055: Proc ALSaveStringToFile(Str: Ansistr; filename: Ansistr);

```

```

12056: Func ALIsInt(const S: Ansistr):Bool;
12057: Func ALIsDecimal(const S: Ansistr):Bool;
12058: Func ALStringToWideString(const S: Ansistr; const aCodePage: Word): WideString;
12059: Func ALWideStringToString(const WS: WideString; const aCodePage: Word): Ansistr;
12060: Func ALQuotedStr(const S: Ansistr; const Quote: AnsiChar = ''' ): Ansistr;
12061: Func ALDequotedStr(const S: Ansistr; AQuote: AnsiChar): Ansistr;
12062: Func ALUTF8removeBOM(const S: Ansistr): Ansistr;
12063: Func ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): Ansistr;
12064: Func ALRandomStr(const aLength: Longint): Ansistr;
12065: Func ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char):Str;
12066: Func ALRandomStrU(const aLength: Longint):Str;
12067: end;
12068:
12069: Proc SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
12070: begin
12071: Proc ALJSONToTStrings(const AJsonStr:Ansistr;aLst:TALStrings; const aNullStr:Ansistr;const
aTrueStr:Ansistr; const aFalseStr: Ansistr)
12072: end;
12073:
12074: Proc SIRegister_ALWindows(CL: TPSPascalCompiler);
12075: begin
12076: _ALMEMORYSTATUSEx', 'record dwLength : DWORD; dwMemoryLoad : DWO'
12077: +RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
12078: +ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
12079: +; ullAvailExtendedVirtual : Int64; end
12080: TALMemoryStatusEx', '_ALMEMORYSTATUSEx
12081: Func ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEx) : BOOL
12082: Func ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG) : LONGLONG
12083: 'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
12084: 'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
12085: 'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
12086: 'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
12087: 'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
12088: end;
12089:
12090: Proc SIRegister_IPCThrd(CL: TPSPascalCompiler);
12091: begin
12092: SIRegister_THandledObject(CL);
12093: SIRegister_TEvent(CL);
12094: SIRegister_TMutex(CL);
12095: SIRegister_TSharedMem(CL);
12096: 'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
12097: 'TRACE_BUFFER','String 'TRACE_BUFFER
12098: 'TRACE_Mutex','String 'TRACE_MUTEX
12099: //PTraceEntry', '^TTraceEntry // will not work
12100: SIRegister_TIPCTracer(CL);
12101: 'MAX_CLIENTS','LongInt'( 6);
12102: 'IPCTIMEOUT','LongInt'( 2000);
12103: 'IPCBUFFER_NAME','String 'BUFFER_NAME
12104: 'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
12105: 'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
12106: 'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
12107: 'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
12108: 'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
12109: 'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
12110: FindClass('TOBJECT'),'EMonitorActive
12111: FindClass('TOBJECT'),'TIPCThread
12112: TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
12113: +l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
12114: +ach, evClientSwitch, evClientSignal, evClientExit )
12115: TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
12116: TClientFlags', 'set of TClientFlag
12117: //PEventData', '^TEventData // will not work
12118: TEventData', 'record X:SmallInt; Y:SmallInt;Flag: TClientFlag; Flags:TClientFlags; end
12119: TConnectEvent', 'Proc ( Sender : TIPCThread; Connecting :Bool)
12120: TDirUpdateEvent', 'Proc ( Sender : TIPCThread)
12121: TIPCTNotifyEvent', 'Proc ( Sender : TIPCThread; Data : TEventData)
12122: //PIPCEventInfo', '^TIPCEventInfo // will not work
12123: TIPCEventInfo', 'record FID:Int;FKind:TEventKind;FData:TEventData;end
12124: SIRegister_TIPCEvent(CL);
12125: //PClientDirRecords', '^TClientDirRecords // will not work
12126: SIRegister_TClientDirectory(CL);
12127: TIPCState', '( stInActive, stDisconnected, stConnected )
12128: SIRegister_TIPCThread(CL);
12129: SIRegister_TIPCMonitor(CL);
12130: SIRegister_TIPCCClient(CL);
12131: Func IsMonitorRunning( var Hndl : THandle ) :Bool
12132: end;
12133:
12134: (*-----*)
12135: Proc SIRegister_ALGSMComm(CL: TPSPascalCompiler);
12136: begin
12137: SIRegister_TALGSMComm(CL);
12138: Func ALGSMComm_BuildPDUMessage( aSMSCenter,aSMSAddress,aMessage: Ansistr) : Ansistr
12139: Proc ALGSMComm_DecodePDUMessage(aPDUMessage:Ansistr;var aSMSCenter,aSMSAddress,aMessage:Ansistr);
12140: Func ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : Ansistr
12141: Func ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:Ansistr;const UseGreekAlphabet:Bool):Widestring;
12142: Func ALMatchesMask(const Filename, Mask: Ansistr):Bool;
12143: end;

```



```

12144:
12145: Proc SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
12146: begin
12147:   TALHTTPPropertyChangeEvent', 'Procedure(sender:TObject;const PropertyIndex:Int;
12148:   TALHTTPProtocolVersion', '( HTTPPv_1_0, HTTPPv_1_1 )
12149:   TALHTTPMethod', '(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
12150:   TInternetScheme', 'Int
12151:   TALIPv6Binary', 'array[1..16] of Char;
12152:   // TALIPv6Binary = array[1..16] of ansiChar;
12153:   // TInternetScheme = Int;
12154:   SIRegister_TALHTTPRequestHeader(CL);
12155:   SIRegister_TALHTTPCookie(CL);
12156:   SIRegister_TALHTTPCookieCollection(CL);
12157:   SIRegister_TALHTTPResponseHeader(CL);
12158:   Func ALHTTPDecode( const AStr : Ansistr ) : Ansistr
12159:   Proc ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
12160:   //Proc ALExtractHTTPFields(Separators,WhiteSpace,
12161:   Quotes:TSysCharSet;Content:PAnsiChar;Strings:TALStrings;StripQuotes:Bool;
12162:   //Proc ALExtractHeaderFields(Separators,WhiteSpace,
12163:   Strings:TALStrings;Decode:Bool;StripQuotes:Bool)
12164:   //Proc ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
12165:   Quotes:TSysCharSet;Content:PAnsiChar;Strings:TALStrings;Decode:Bool;StripQuotes:Bool)
12166:   Func ALRemoveSchemeFromUrl( aUrl : Ansistr ) : Ansistr
12167:   Func ALExtractSchemeFromUrl( aUrl : Ansistr ) : TInternetScheme
12168:   Func ALExtractHostNameFromUrl( aUrl : Ansistr ) : Ansistr
12169:   Func ALExtractDomainNameFromUrl( aUrl : Ansistr ) : Ansistr
12170:   Func ALExtractUrlPathFromUrl( aUrl : Ansistr ) : Ansistr
12171:   Func ALInternetCrackUrl(aUrl:Ansistr;var SchemeName,HostName,UserName,Password,UrlPath,ExtraInfo:
12172:   Ansistr; var PortNumber:Int):Bool;
12173:   Func ALInternetCrackUrl1(aUrl:Ansistr;var SchemeNam,HostName,UserName,Password,UrlPath,
12174:   Anchor:Ansistr;Query:TALStrings;var PortNumber:Int):Bool
12175:   Func ALInternetCrackUrl2(var Url:Ansistr;var Anchor:Ansistr;Query:TALStrings):Bool;
12176:   Func ALRemoveAnchorFromUrl( aUrl : Ansistr; var aAnchor : Ansistr ) : Ansistr;
12177:   Func ALRemoveAnchorFromUrl1( aUrl : Ansistr ) : Ansistr;
12178:   Func ALCombineUrl( RelativeUrl, BaseUrl : Ansistr ) : Ansistr;
12179:   Func ALCombineUrl1(RelativeUrl,BaseUrl,Anchor:Ansistr; Query:TALStrings): Ansistr;
12180:   Func ALGmtDateTimeToRfc822Str( const aValue : TDateTime ) : Ansistr
12181:   Func ALDateTimeToRfc822Str( const aValue : TDateTime ) : Ansistr
12182:   Func ALTryRfc822StrToGMTDateTime( const S : Ansistr; out Value : TDateTime ) :Bool
12183:   Func ALRfc822StrToGMTDateTime( const s : Ansistr ) : TDateTime
12184:   Func ALTryIPv4StrToNumeric( aIPv4Str : Ansistr; var aIPv4Num :Card ) :Bool
12185:   Func ALIPv4StrToNumeric( aIPv4 : Ansistr ) :Card
12186:   Func ALNumericToIPv4Str( aIPv4 :Card ) : Ansistr
12187:   Func ALZeroIpV6 : TALIPv6Binary
12188:   Func ALTryIPv6StrToBinary( aIPv6Str : Ansistr; var aIPv6Bin : TALIPv6Binary ) :Bool
12189:   Func ALIPv6StrToBinary( aIPv6 : Ansistr ) : TALIPv6Binary
12190:   Func ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary ) : Ansistr
12191:   Func ALBinaryStrToIPv6Binary( aIPv6BinaryStr : Ansistr ) : TALIPv6Binary
12192: end;
12193: Proc SIRegister_ALFcnHTML(CL: TPSPascalCompiler); //JavaScript
12194: begin
12195:   Proc ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const DecodeHTMLText:Bool;
12196:   Func ALUTF8ExtractHTMLText1(HtmlContent:Ansistr;const DecodeHTMLText:Bool):Ansistr;
12197:   Func ALXMLCDATAElementEncode( Src : Ansistr ) : Ansistr
12198:   Func ALXMLTextElementEncode(Src:Ansistr;const useNumericReference:bool):Ansistr
12199:   Func ALUTF8XMLTextElementDecode( const Src : Ansistr ) : Ansistr
12200:   Func ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const useNumRef:bool):Ansistr;
12201:   Func ALUTF8HTMLDecode( const Src : Ansistr ) : Ansistr
12202:   Func ALJavascriptEncode(const Src:Ansistr;const useNumericReference:bool):Ansistr
12203:   Func ALUTF8JavascriptDecode( const Src : Ansistr ) : Ansistr
12204:   Proc ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:Ansistr;const
12205:   DeleteBodyOfUnwantedTag:Bool;const ReplaceUnwantedTagCharBy:AnsiChar)
12206:   Proc ALCompactHtmlTagParams( TagParams : TALStrings)
12207:   Proc ALUTF8JavascriptDecodeV( var Str : Ansistr)
12208:   //Func ALUTF8JavascriptDecode( const Src : Ansistr ) : Ansistr
12209:   {This Func evaluates the Javascript code given in the
12210:   parameter "aCode" and returns result. The Func works
12211:   similar to browser's console, so you can send even the code
12212:   like this "2+2" => returns "4".}
12213:   Example: writeln('runJScript: '+runJS('Math.pow(2, 64)'))
12214:   Func ALRunJavascript( const aCode : Ansistr ) : Ansistr
12215:   Func RunJavascript( const aCode : Ansistr ) : Ansistr
12216:   Func RunJavascript2(const aCode:Ansistr):Ansistr //no CoInitialize of COM-Server
12217:   Func RunJS( const aCode : Ansistr ) : Ansistr //no CoInitialize of COM-Server
12218:   //CL.AddDelphiFunction('Proc ALHideHtmlUnwantedTagForHTMLHandleTagfunct( var HtmlContent : Ansistr; const
12219:   DeleteBodyOfUnwantedTag:Boolean;const ReplaceUnwantedTagCharBy:AnsiChar)
12220:   //CL.AddDelphiFunction('Proc ALCompactHtmlTagParams( TagParams : TALStrings)
12221:   Func ALJavascriptEncodeU(const Src:str; const useNumericReference:boolean) :Str
12222:   Proc ALJavascriptDecodeVU( var Str :Str)
12223:   Func ALJavascriptDecodeU( const Src :Str):Str
12224: end;
12225: Proc SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12226: begin
12227:   SIRegister_TALEmailHeader(CL);

```

```

12226:   SIRegister_TALNewsArticleHeader(CL);
12227: Func AlParseEmailAddress(FriendlyEmail:AnsiStr;var RealName:AString;const decodeRealName:Bool):AnsiStr;
12228: Func AlExtractEmailAddress( FriendlyEmail : AnsiStr) : AnsiStr
12229: Func AlMakeFriendlyEmailAddress( aRealName, aEmail : AnsiStr) : AnsiStr
12230: Func AlEncodeRealName4FriendlyEmailAddress( aRealName : AnsiStr) : AnsiStr
12231: Func AlGenerateInternetMessageID : AnsiStr;
12232: Func AlGenerateInternetMessageID1( ahostname : AnsiStr) : AnsiStr;
12233: Func AlDecodeQuotedPrintableString( src : AnsiStr) : AnsiStr
12234: Func AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiStr;aDefaultCodePage:Int):AnsiStr;
12235: end;
12236:
12237: (*-----*)
12238: Proc SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12239: begin
12240:   Func ALHostToIP( HostName : AnsiStr; var Ip : AnsiStr):Boolean
12241:   Func ALIPAddrToName( IPAddr : AnsiStr) : AnsiStr
12242:   Func ALgetLocalIPs : TALStrings
12243:   Func ALgetLocalHostName : AnsiStr
12244: end;
12245:
12246: Proc SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12247: begin
12248:   Proc AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest : TALWebRequest;ServerVariables:TALStrings);
12249:   Proc AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
TALStrings; ScriptName, ScriptFileName : AnsiStr; Url : AnsiStr);
12250:   Proc AlCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12251:   Proc AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,ScriptFileName:AnsiStr;Url:AnsiStr;
12252:   Proc
AlCGIInitServerVariablesFromWebRequest(WebRequest:TALWebRequest;ServerVariables:TALStrings;ScriptName,ScriptFileName:A
AnsiStr);
12253:   Proc AlCGIExec( InterpreterFilename: AnsiStr; ServerVariables: TALStrings; RequestContentStream: Tstream;
ResponseContentStream: Tstream; ResponseHeader: TALHTTPResponseHeader);
12254:   Proc AlCGIExec1(ScriptName,ScriptFileName,Url, X_REWRITE_URL, InterpreterFilename:AnsiStr; WebRequest :
TALIsapiRequest; overloadedCookies:AnsiStr;overloadedQueryString:AnsiStr;overloadedReferer: AnsiStr;'
+ 'overloadedRequestContentStream:Tstream;var
ResponseContentStr:AnsiStr;ResponseHeader:TALHTTPResponseHeader;
12255:   Proc AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,InterpreterFilename:AnsiStr;WebRequest:
TALIsapiRequest; var ResponseContentString : AnsiStr; ResponseHeader : TALHTTPResponseHeader);
12256: end;
12257:
12258:
12259: Proc SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12260: begin
12261:   TStartupInfoA', 'TStartupInfo
12262:   'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12263:   SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12264:   SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12265:   SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12266:   SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12267:   SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12268:   SE_TCB_NAME','String 'SeTcbPrivilege
12269:   SE_SECURITY_NAME','String 'SeSecurityPrivilege
12270:   SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12271:   SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12272:   SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12273:   SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12274:   SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12275:   SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12276:   SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12277:   SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12278:   SE_BACKUP_NAME','String 'SeBackupPrivilege
12279:   SE_RESTORE_NAME','String 'SeRestorePrivilege
12280:   SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12281:   SE_DEBUG_NAME','String 'SeDebugPrivilege
12282:   SE_AUDIT_NAME','String 'SeAuditPrivilege
12283:   SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12284:   SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12285:   SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12286:   SE_UNDOCK_NAME','String 'SeUndockPrivilege
12287:   SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12288:   SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12289:   SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12290:   Func AlGetEnvironmentString : AnsiStr
12291:   Func ALWinExec32(const FileName,CurrentDir,Environment:AnsiStr;InStream:Tstream;OutStream:TStream):Dword;
12292:   Func ALWinExec321(const FileName:AnsiStr;InputStream:Tstream;OutputStream:TStream):Dword;
12293:   Func ALWinExecAndWait32( FileName : AnsiStr; Visibility : Int) : DWORD
12294:   Func ALWinExecAndWait32V2( FileName : AnsiStr; Visibility : Int) : DWORD
12295:   Func ALNTSetPrivilege( sPrivilege : AnsiStr; bEnabled :Bool) :Bool
12296: end;
12297:
12298: Proc SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12299: begin
12300:   Func AlEmptyDirectory(Directory:AnsiStr;SubDirectory:Bool;IgnoreFiles:array of AnsiStr; const
RemoveEmptySubDirectory :Bool; const FileNameMask : AnsiStr; const MinFileAge : TdateTime):Boolean;
12301:   Func AlEmptyDirectory1( Directory : AnsiStr; SubDirectory :Bool; const RemoveEmptySubDirectory:Bool;const
FileNameMask:ansiStr;const MinFileAge:TdateTime):Bool;
12302:   Func AlCopyDirectory(SrcDirectory,DestDirectory:AnsiStr;SubDirectory:Boolean; const FileNameMask:
AnsiStr; const FailIfExists:Boolean):Bool
12303:   Func ALGetModuleName : AnsiStr
12304:   Func ALGetModuleFileNameWithoutExtension : AnsiStr

```

```

12305: Func ALGetModulePath : Ansistr
12306: Func ALGetFileSize( const AFileName : Ansistr) : int64
12307: Func ALGetFileVersion( const AFileName : Ansistr) : Ansistr
12308: Func ALGetFileCreationDateTime( const aFileName : Ansistr) : TDateTime
12309: Func ALGetFileLastWriteDateTime( const aFileName : Ansistr) : TDateTime
12310: Func ALGetFileLastAccessDateTime( const aFileName : Ansistr) : TDateTime
12311: Proc ALSetFileCreationDateTime(const aFileName:Ansistr;const aCreationDateTime:TDateTime)
12312: Func ALIsDirectoryEmpty( const directory : Ansistr) :Bool
12313: Func ALFileExists( const Path : Ansistr) :Bool
12314: Func ALDirectoryExists( const Directory : Ansistr) :Bool
12315: Func ALCreateDir( const Dir : Ansistr) :Bool
12316: Func ALRemoveDir( const Dir : Ansistr) :Bool
12317: Func ALDeleteFile( const FileName : Ansistr) :Bool
12318: Func ALRenameFile( const OldName, NewName : Ansistr) :Bool
12319: end;
12320:
12321: Proc SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12322: begin
12323:   NativeInt', 'Int NativeUInt', 'Cardinal
12324:   Func ALMimeBase64EncodeString( const S : Ansistr) : Ansistr
12325:   Func ALMimeBase64EncodeStringNoCRLF( const S : Ansistr) : Ansistr
12326:   Func ALMimeBase64DecodeString( const S : Ansistr) : Ansistr
12327:   Func ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12328:   Func ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12329:   Func ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12330:   Proc ALMimeBase64Encode(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12331:   Proc ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray;InputOffset:NativeInt; const
InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12332:   Proc ALMimeBase64EncodeFullLines( const InputBuffer: TByteDynArray; InputOffset : NativeInt;const
InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt)
12333:   Func ALMimeBase64Decode(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt) : NativeInt;
12334:   Func ALMimeBase64DecodePartial(const InputBuffer:TByteDynArray;InputOffset:NativeInt;const
InputByteCount:NativeInt;out OutputBuffer:TByteDynArray;OutputOffset:NativeInt;
12335:   + var ByteBuffer:Card;var ByteBufferSize:Card):NativeInt;
12336:   Func ALMimeBase64DecodePartialEnd(out OutputBuffer:TByteDynArray;OutputOffset:NativeInt;const
ByteBuffer:Card;const ByteBufferSize:Card):NativeInt;
12337:   Proc ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
OutputBuf:TByteDynArray);
12338:   Proc ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12339:   Proc ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12340:   Func ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray):NativeInt;
12341:   Func ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray;var ByteBuffer:Card;var ByteBufferSize:Card) : NativeInt;
12342:   Func ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Card;const
ByteBufferSpace:Card):NativeInt;
12343:   Proc ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12344:   Proc ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12345:   Proc ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12346:   Proc ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12347:   Proc ALMimeBase64EncodeStreamNoCRLF(const InputStream:TStream;const OutputStream:TStream)
12348:   Proc ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12349:   'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12350:   'cALMimeBase64_DECODED_LINE_BREAK','LongInt'(cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12351:   'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12352:   Proc ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12353:   Proc ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12354:   Func ALGetDefaultFileExtFromMimeContentType( aContentType : Ansistr) : Ansistr
12355:   Func ALGetDefaultMIMEContentTypeFromExt( aExt : Ansistr) : Ansistr
12356: end;
12357:
12358: Proc SIRegister_ALXmldoc(CL: TPSPascalCompiler);
12359: begin
12360:   'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12361:   FindClass('TOBJECT'),TALXMLNode
12362:   FindClass('TOBJECT'),TALXMLNodeList
12363:   FindClass('TOBJECT'),TALXMLDocument
12364:   TALXMLParseProcessingInstructionEvent','Proc (Sender:TObject;const Target,Data:AnsiStr)
12365:   TALXMLParseTextEvent','Proc (Sender: TObject; const str: Ansistr)
12366:   TALXMLParseStartElementEvent','Procedure (Sender:TObject;const Name:Ansistr;const Attributes:TALStrings)
12367:   TALXMLParseEndElementEvent','Procedure (Sender: TObject;const Name: Ansistr)
12368:   TALXMLNodeType','(ntReserved,ntElement,ntAttribute,ntText,ntCDATA,ntEntityRef,ntEntity,ntProcessingInstr,
ntComment,ntDocumentntDocType,ntDocFragment,ntNotation)
12369:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12370:   TALXMLDocOptions', 'set of TALXMLDocOption
12371:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12372:   TALXMLParseOptions', 'set of TALXMLParseOption
12373:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12374:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12375:   SIRegister_EALXMLDocError(CL);
12376:   SIRegister_TALXMLNodeList(CL);
12377:   SIRegister_TALXMLNode(CL);
12378:   SIRegister_TALXMLElementNode(CL);
12379:   SIRegister_TALXMLAttributeNode(CL);
12380:   SIRegister_TALXMLTextNode(CL);

```

```

12381: SIRegister_TALXmlDocumentNode(CL);
12382: SIRegister_TALXmlCommentNode(CL);
12383: SIRegister_TALXmlProcessingInstrNode(CL);
12384: SIRegister_TALXmlCDATADataNode(CL);
12385: SIRegister_TALXmlEntityRefNode(CL);
12386: SIRegister_TALXmlEntityNode(CL);
12387: SIRegister_TALXmlDocTypeNode(CL);
12388: SIRegister_TALXmlDocFragmentNode(CL);
12389: SIRegister_TALXmlNotationNode(CL);
12390: SIRegister_TALXMLDocument(CL);
12391: cALXMLUTF8EncodingStr, 'String' 'UTF-8
12392: cALXMLUTF8HeaderStr, 'String' '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>' + #13#10;
12393: CALNSDelim, 'String' '
12394: CALXML, 'String' 'xml
12395: CALVersion, 'String' 'version
12396: CALEncoding, 'String' 'encoding
12397: CALStandalone, 'String' 'standalone
12398: CALDefaultNodeIndent, 'String' '
12399: CALXmlDocument, 'String' 'DOCUMENT
12400: Func ALCreateEmptyXMLDocument(const Rootname: AnsiStr): TalXMLDocument
12401: Proc ALClearXMLDocument(const rootname: AnsiStr; xmlDoc: TalXMLDocument; const EncodingStr: AnsiStr);
12402: Func ALFindXmlNodeByChildNodeValue(xmlrec: TalXmlNode; const ChildNodeName, ChildNodeValue: AnsiStr; const
Recurse: Bool): TalXmlNode
12403: Func ALFindXmlNodeByNameAndChildNodeValue(xmlrec: TalXmlNode; const NodeName: AnsiStr; const
ChildNodeName, ChildNodeValue: AnsiStr; const Recurse: Bool): TalXmlNode
12404: Func ALFindXmlNodeByAttribute(xmlrec: TalXmlNode; const AttributeName, AttributeValue: AnsiStr; const
Recurse: Bool): TalXmlNode
12405: Func ALFindXmlNodeByNameAndAttribute(xmlrec: TalXmlNode; const NodeName: AnsiStr; const AttributeName,
AttributeValue: AnsiStr; const Recurse: Bool): TalXmlNode
12406: Func ALExtractAttrValue(const AttrName, AttrLine: AnsiStr; const Default: AnsiStr): AnsiStr
12407: end;
12408:
12409: Proc SIRegister_TProcess(CL: TPSPascalCompiler);
12410: begin
12411:   //with RegClassS(CL, 'TComponent', 'TProcess') do
12412:   with CL.AddClassN(CL.FindClass('TComponent'), 'TProcess') do begin
12413:     Constructor Create(AOwner: TComponent)
12414:     Proc Free
12415:     Proc Execute
12416:     Func Resume: Integer
12417:     Func Suspend: Integer
12418:     Func Terminate(AExitCode: Integer): Bool
12419:     Func WaitOnExit: DWord
12420:     WindowRect, 'Trect', iptrw);
12421:     StartupInfo, 'TStartupInfo', iptrw);
12422:     ProcessAttributes, 'TSecurityAttributes', iptrw);
12423:     ProcessInformation, 'TProcessInformation', iptrw);
12424:     Handle, 'THandle', iptrw);
12425:     ThreadHandle, 'THandle', iptrw);
12426:     Input, 'TOutputPipeStream', iptrw);
12427:     Output, 'TInputPipeStream', iptrw);
12428:     StdErr, 'TinputPipeStream', iptrw);
12429:     ExitStatus, 'Integer', iptrw);
12430:     Active, 'Boolean', iptrw);
12431:     ApplicationName, 'String', iptrw);
12432:     CommandLine, 'String', iptrw);
12433:     ConsoleTitle, 'String', iptrw);
12434:     CurrentDirectory, 'String', iptrw);
12435:     DeskTop, 'String', iptrw);
12436:     Environment, 'TStrings', iptrw);
12437:     FillAttribute, 'Cardinal', iptrw);
12438:     InheritHandles, 'LongBool', iptrw);
12439:     Options, 'TProcessOptions', iptrw);
12440:     Priority, 'TProcessPriority', iptrw);
12441:     StartUpOptions, 'TStartUpOptions', iptrw);
12442:     Running, 'Boolean', iptrw);
12443:     ShowWindow, 'TShowWindowOptions', iptrw);
12444:     ThreadAttributes, 'TSecurityAttributes', iptrw);
12445:     WindowColumns, 'Cardinal', iptrw);
12446:     WindowHeight, 'Cardinal', iptrw);
12447:     WindowLeft, 'Cardinal', iptrw);
12448:     WindowRows, 'Cardinal', iptrw);
12449:     WindowTop, 'Cardinal', iptrw);
12450:     WindowWidth, 'Cardinal', iptrw);
12451:   end;
12452: end;
12453:
12454: (*-----*)
12455: Proc SIRegister_TalWebSpider(CL: TPSPascalCompiler);
12456: begin
12457:   //with RegClassS(CL, 'TObject', 'TalWebSpider') do
12458:   with CL.AddClassN(CL.FindClass('TObject'), 'TalWebSpider') do begin
12459:     RegisterMethod('Proc Crawl
12460:     RegisterMethod('Proc UpdateLinkToLocalPath
12461:     OnCrawlBeforeDownload, 'TalWebSpiderCrawlBeforeDownloadEvent', iptrw);
12462:     OnCrawlAfterDownload, 'TalWebSpiderCrawlAfterDownloadEvent', iptrw);
12463:     OnCrawlDownloadSuccess, 'TalWebSpiderCrawlDownloadSuccessEvent', iptrw);
12464:     OnCrawlDownloadRedirect, 'TalWebSpiderCrawlDownloadRedirectEvent', iptrw);
12465:     OnCrawlDownloadError, 'TalWebSpiderCrawlDownloadErrorEvent', iptrw);

```

```

12466: OnCrawlGetNextLink', 'TalWebSpiderCrawlGetNextLinkEvent', iptrw);
12467: OnCrawlFindLink', 'TalWebSpiderCrawlFindLinkEvent', iptrw);
12468: OnCrawlEnd', 'TalWebSpiderCrawlEndEvent', iptrw);
12469: OnUpdateLinkToLocalPathGetNextFile', 'TalWebSpiderUpdateLinkToLocalPathGetNextFileEvent'rw);
12470: OnUpdateLinkToLocalPathFindLink', 'TalWebSpiderUpdateLinkToLocalPathFindLinkEvent', iptrw);
12471: OnUpdateLinkToLocalPathEnd', 'TalWebSpiderUpdateLinkToLocalPathEndEvent', iptrw);
12472: RegisterProperty('HttpClient', 'TalHttpClient', iptrw);
12473: end;
12474: end;
12475:
12476: https://sourceforge.net/p/alcinoo/code/HEAD/tree/source/ALWebSpider.pas#l184
12477: {-----}
12478: TALTrivialWebSpider = Class(TObject)
12479: Private
12480:   FWebSpider: TalWebSpider;
12481:   fStartUrl: Ansistr;
12482:   fLstUrlCrawled: TALStrings;
12483:   fLstErrorEncountered: TALStrings;
12484:   FPageDownloadedBinTree: TALStringKeyAVLBinaryTree;
12485:   FPageNotYetDownloadedBinTree: TALStringKeyAVLBinaryTree;
12486:   FCurrentDeepLevel: Integer;
12487:   FCurrentLocalFileNameIndex: Integer;
12488:   fMaxDeepLevel: Integer;
12489:   fOnCrawlBeforeDownload: TALWebSpiderCrawlBeforeDownloadEvent;
12490:   fUpdateLinkToLocalPath: Bool;
12491:   fExcludeMask: Ansistr;
12492:   fStayInStartDomain: Bool;
12493:   fSaveDirectory: Ansistr;
12494:   fSplitDirectoryAmount: integer;
12495:   FHttpClient: TalHttpClient;
12496:   fIncludeMask: Ansistr;
12497:   fOnCrawlAfterDownload: TALWebSpiderCrawlAfterDownloadEvent;
12498:   fOnCrawlFindLink: TALTrivialWebSpiderCrawlFindLinkEvent;
12499:   fDownloadImage: Bool;
12500:   fOnUpdateLinkToLocalPathProgress: TALTrivialWebSpiderUpdateLinkToLocalPathProgressEvent;
12501:   fOnCrawlProgress: TALTrivialWebSpiderCrawlProgressEvent;
12502:   Proc WebSpiderCrawlDownloadError(Sender: TObject; const URL, ErrorMessage: Ansistr; HTTPResponseHeader:
TALHTTPResponseHeader; var StopCrawling: Bool);
12503:   Proc WebSpiderCrawlDownloadRedirect(Sender: TObject; const Url, RedirectedTo: Ansistr;
HTTPResponseHeader: TALHTTPResponseHeader; var StopCrawling: Bool);
12504:   Proc WebSpiderCrawlDownloadSuccess(Sender: TObject; const Url: Ansistr; HTTPResponseHeader:
TALHTTPResponseHeader; HTTPResponseContentTStream; var StopCrawling: Bool);
12505:   Proc WebSpiderCrawlFindLink(Sender: TObject; const HtmlTagString: AnsiStr; HtmlTagParams:
TALStrings; const URL: Ansistr);
12506:   Proc WebSpiderCrawlGetNextLink(Sender: TObject; var Url: Ansistr);
12507:   Proc WebSpiderUpdateLinkToLocalPathFindLink(Sender: TObject; const HtmlTagString: AnsiStr;
HtmlTagParams: TALStrings; const URL: Ansistr; var LocalPath: Ansistr);
12508:   Proc WebSpiderUpdateLinkToLocalPathGetNextFile(Sender: TObject; var FileN, BaseHref: AnsiStr);
12509:   Func GetNextLocalFileName(const aContentType: Ansistr): Ansistr;
12510: Protected
12511: Public
12512:   Constructor Create;
12513:   Destructor Destroy; override;
12514:   Proc Crawl(const aUrl: Ansistr); overload; {Launch the Crawling of the page}
12515:   Proc Crawl(const aUrl: AnsiStr; LstUrlCrawled: TALStrings; LstErrorEncountered: TALStrings);
12516:   Property HttpClient: TalHttpClient read FHttpClient write FHttpClient;
12517:   Property DownloadImage: Bool read fDownloadImage write fDownloadImage default false;
12518:   Property StayInStartDomain: Bool read fStayInStartDomain write fStayInStartDomain default true;
12519:   Property UpdateLinkToLocalPath: Bool read fUpdateLinkToLocalPath write fUpdateLinkToLocalPath default
True;
12520:   Property MaxDeepLevel: Integer read fMaxDeepLevel write fMaxDeepLevel default -1;
12521:   Property ExcludeMask: Ansistr read fExcludeMask write fExcludeMask;
12522:   Property IncludeMask: Ansistr read fIncludeMask write fIncludeMask;
12523:   Property SaveDirectory: Ansistr read fSaveDirectory write fSaveDirectory;
12524:   Property SplitDirectoryAmount: integer read fSplitDirectoryAmount write fSplitDirectoryAmount default
5000;
12525:   Property OnCrawlBeforeDownload: TALWebSpiderCrawlBeforeDownloadEvent read fOnCrawlBeforeDownload write
fOnCrawlBeforeDownload; {When a page is successfully downloaded}
12526:   Property OnCrawlAfterDownload: TALWebSpiderCrawlAfterDownloadEvent read fOnCrawlAfterDownload write
fOnCrawlAfterDownload; {When a page is successfully downloaded}
12527:   Property OnCrawlFindLink: TALTrivialWebSpiderCrawlFindLinkEvent read fOnCrawlFindLink write
fOnCrawlFindLink; {When a link is found}
12528:   Property OnCrawlProgress: TALTrivialWebSpiderCrawlProgressEvent read fOnCrawlProgress write
fOnCrawlProgress;
12529:   Property OnUpdateLinkToLocalPathProgress: TALTrivialWebSpiderUpdateLinkToLocalPathProgressEvent read
fOnUpdateLinkToLocalPathProgress write fOnUpdateLinkToLocalPathProgress;
12530: end;
12531:
12532: Proc SRegister_ALOpenOffice(CL: TPSPascalCompiler);
12533: begin
12534:   CL.AddClassN(CL.FindClass('TObject'), 'EALOpenOfficeException
12535:   Proc ConnectOpenOffice
12536:   Proc DisconnectOpenOffice( aTerminateOpenOffice :Bool)
12537:   Func IsOpenOfficeConnected :Bool
12538:   Func CreateUnoService( const aServiceName : Ansistr) : Variant
12539:   Func CreateUnoStruct(const aStructureName: Ansistr; const aMaxIndex: integer): Variant
12540:   Func HasUnoInterfaces( aObject: Variant; aInterfaceList: array of Ansistr): Bool
12541:   Func CreateOOProperties( aPropertyList : array of Variant) : Variant
12542:   Func MakeOOPropertyValue(aPropertyName: Ansistr; aPropertyValue: Variant): Variant

```



```

12546: Func CreateOOCalcDocument : Variant
12547: Func CreateOOWordDocument : Variant;
12548: Func CreateOOImpressDocument : Variant
12549: Func CreateOODrawDocument : Variant
12550: Proc SaveOODocument(aDocument: Variant; aFileName: Ansistr;aFileType : Ansistr)
12551: Proc CreateOOSheet( aDocument : Variant; const aSheetName : Ansistr)
12552: Proc SetColumnWidth(aSheet:Variant;const aColumnIndex:int;const aWidthInCentimetres:int);
12553: Proc SetColumnWidth1(aSheet:Variant;aColumn:Variant;const aWidthInCentimetres:integer);
12554: Proc SetCellBold( aCell : Variant)
12555: Proc SetCellBorder( aCellRange : Variant; const aBorderColor : Longword)
12556: Func IsVariantNullOrEmpty( aVariant : Variant) :Bool
12557: Func VarDummyArray : Variant
12558: Func ConvertToURL( aWinAddress : Ansistr) : Ansistr
12559: Func ConvertFromURL( aUrlAddress : Ansistr) : Ansistr
12560: Func OORGB( aRedByte, aGreenByte, aBlueByte : byte) : Longword
12561: Func AlCopyStr2(const aSourceString: Ansistr; aStart, aLength: Integer): Ansistr;
12562: end;
12563:
12564: Proc SIRegister_ALExecute2(CL: TPSPascalCompiler);
12565: begin
12566: //CL.AddTypes('TStartupInfoA', 'TStartupInfo
12567: CL.AddConstantN('SE_CREATE_TOKEN_NAME','String').SetString( 'SeCreateTokenPrivilege
12568: SE_ASSIGNPRIMARYTOKEN_NAME','String').SetString( 'SeAssignPrimaryTokenPrivilege
12569: SE_LOCK_MEMORY_NAME','String').SetString( 'SeLockMemoryPrivilege
12570: SE_INCREASE_QUOTA_NAME','String').SetString( 'SeIncreaseQuotaPrivilege
12571: SE_UNSOLICITED_INPUT_NAME','String').SetString( 'SeUnsolicitedInputPrivilege
12572: SE_MACHINE_ACCOUNT_NAME','String').SetString( 'SeMachineAccountPrivilege
12573: Func AlGetEnvironmentStringX : Ansistr
12574: Func ALWinExec0X(const aCommandLine:Ansistr;const aCurrentDirectory:Ansistr;const
aEnvironment:Ansistr;const aInputStream:Tstream;const aOutputStream Stream;const aOwnerThread
TThread):Dword;
12575: Func ALWinExec1X(const aCommandLine: Ansistr; const aInputStream: Tstream; const aOutputStream: TStream;
const aOwnerThread: TThread): Dword;
12576: Proc ALWinExec2X( const aCommandLine:Ansistr; const aCurrentDirectory : Ansistr);
12577: Proc ALWinExec3X(const aUserName:Ansistr;const aPassword:Ansistr;const aCommandLine:Ansistr;const
aCurrentDirectory:Ansistr;const aLogonFlags:dword);
12578: Func ALWinExecAndWait4X(const aCommandLine: Ansistr;const aCurrentDirectory:Ansistr;const
aEnvironment:Ansistr;const aVisibility:integer): DWORD;
12579: Func ALWinExecAndWait5X(const aCommandLine:Ansistr; const aVisibility:integer):DWORD;
12580: Func ALWinExecAndWaitV2X(const aCommandLine:Ansistr;const aVisibility integer):DWORD
12581: Func ALNTSetPrivilegeX( const sPrivilege : Ansistr; bEnabled :Bool) :Bool
12582: Func ALStartServiceX(const aServiceName:Ansistr;const aComputerName:Ansistr;const aTimeout:integer):bool;
12583: Func ALStopServiceX(const aServiceName:Ansistr;const aComputerName:Ansistr;const aTimeout:integer):bool;
12584: Func ALMakeServiceAutoRestartingX( const aServiceName : Ansistr; const aComputerName : Ansistr; const
aTimeToRestartInSec:integer; const aTimeToResetInSec: integer):Bool
12585: end;
12586:
12587: Proc SIRegister_ALIsapiHTTP(CL: TPSPascalCompiler);
12588: begin
12589: CL.AddClassN(CL.FindClass('TOBJECT'),'EALIsapiRequestContentSizeTooBig
12590: CL.AddClassN(CL.FindClass('TOBJECT'),'EALIsapiRequestConnectionDropped
12591: SIRegister_TALWebRequest(CL);
12592: SIRegister_TALISAPIRequest(CL);
12593: SIRegister_TALWebResponse(CL);
12594: SIRegister_TALISAPIResponse(CL);
12595: Func ALIsapiHttpStatusString( StatusCode : Integer) : Ansistr
12596: CL.AddConstantN('HSE_IO_SYNC','LongWord').SetUInt( $00000001);
12597: 'HSE_IO_ASYNC','LongWord').SetUInt( $00000002);
12598: 'HSE_IO_DISCONNECT_AFTER_SEND','LongWord').SetUInt( $00000004);
12599: 'HSE_IO_SEND_HEADERS','LongWord').SetUInt( $00000008);
12600: end;
12601:
12602: Proc SIRegister_uUsb(CL: TPSPascalCompiler);
12603: begin
12604: //CL.AddTypes('PDevBroadcastHdr', '^DEV_BROADCAST_HDR // will not work
12605: CL.AddTypeS(DEV_BROADCAST_HDR','record dbch_size:DWORD;dbch_devicetype:DWORD;dbch_reserved:DWORD;end;
12606: CL.AddTypeS('TDevBroadcastHdr', 'DEV_BROADCAST_HDR
12607: //CL.AddTypes('PDevBroadcastDeviceInterface','^DEV_BROADCAST_DEVICEINTERFACE//will not work
12608: CL.AddTypeS(DEV_BROADCAST_DEVICEINTERFACE', 'record dbcc_size : DWORD; dbcc_'
+devicetype: DWORD;dbcc_reserved:DWORD; dbcc_classguid: TGUID; dbcc_name:Char; end
12610: CL.AddTypeS('TDevBroadcastDeviceInterface', 'DEV_BROADCAST_DEVICEINTERFACE
12611: //CL.AddConstantN('GUID_DEVINTERFACE_USB_DEVICE','TGUID').SetString( '{A5DCBF10-6530-11D2-901F-
00C04FB951ED)
12612: 'DBT_DEVICEARRIVAL','LongWord').SetUInt( $8000);
12613: 'DBT_DEVICEREMOVECOMPLETE','LongWord').SetUInt( $8004);
12614: 'DBT_DEVTYP_DEVICEINTERFACE','LongWord').SetUInt( $00000005);
12615: CL.AddTypeS('TUsbNotifyProc', 'Proc ( Sender : TObject; const DeviceName:Str)
12616: SIRegister_TUsbNotifier(CL);
12617: end;
12618:
12619: Proc SIRegister_uWebcam(CL: TPSPascalCompiler);
12620: begin
12621: CL.AddConstantN('WM_CAP_START','LongWord').SetUInt( $0400);
12622: 'WM_CAP_DRIVER_CONNECT','LongInt').SetInt( $0400 + 10);
12623: 'WM_CAP_DRIVER_DISCONNECT','LongInt').SetInt( $0400 + 11);
12624: 'WM_CAP_SAVEDIB','LongInt').SetInt( $0400 + 25);
12625: 'WM_CAP_GRAB_FRAME','LongInt').SetInt( $0400 + 60);
12626: 'WM_CAP_STOP','LongInt').SetInt( $0400 + 68);
12627: Func Connectwebcam( WebcamID : integer) :Bool

```

```

12628: Proc CaptureWebCam( FilePath :Str)
12629: Proc CloseWebcam( );
12630: Proc WebcamInit;
12631: Func WebcamList: TStringlist;
12632: end;
12633:
12634: Proc WebcamInit;
12635: begin
12636:   LibHandle := LoadLibrary('avicap32.dll
12637:   CapGetDriverDescriptionA:= GetProcAddress(LibHandle,'capGetDriverDescriptionA
12638:   CapCreateCaptureWindowA := GetProcAddress(LibHandle,'capCreateCaptureWindowA
12639: end;
12640: Example: webcaminit;
12641:   writeln(webcamlist.text)
12642:   if Connectwebcam(0) then CaptureWebCam(Exepath+'foto4.png
12643:   CloseWebcam();
12644:
12645: Top5 Functions mX4.7.1.80
12646: Proc LoadResourceFile3HTML(aFile:str; ms:TMemoryStream);
12647: Func VarArrayToStr(const vArray: variant):Str;
12648: Func VarStrNull(const V:OleVariant):str; //avoid problems with null strings
12649: Func GetWMIObject(const objectName:Str): IDispatch; //create the Wmi instance
12650: Func GetAntiVirusProductInfo: TStringlist;
12651: Func EnDeCrypt(const Value :Str) :Str;
12652: //http://www.delphibasics.info/home/delphibasicsnippets/executingpreparedshellcodeindelphi
12653:
12654: Proc SIRegister_TeCanvas(CL: TPSPascalCompiler);
12655: //based on TEEProc, TeCanvas, TEEEngine, TChart
12656: begin
12657:   'TeePiStep','Double').setExtended( Pi / 180.0);
12658:   'TeeDefaultPerspective','LongInt'( 100);
12659:   'TeeMinAngle','LongInt'( 270);
12660:   'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12661:   'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12662:   'teeclCream','LongWord( TColor ( $F0FBFF ));
12663:   'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12664:   'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12665:   'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12666:   'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12667:   'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12668:   'TA_LEFT','LongInt'( 0);
12669:   'TA_RIGHT','LongInt'( 2);
12670:   'TA_CENTER','LongInt'( 6);
12671:   'TA_TOP','LongInt'( 0);
12672:   'TA_BOTTOM','LongInt'( 8);
12673:   'teePATCOPY','LongInt'( 0);
12674:   'NumCirclePoints','LongInt'( 64);
12675:   'teeDEFAULT_CHARSET','LongInt'( 1);
12676:   'teeANTIALIASED_QUALITY','LongInt'( 4);
12677:   'TA_LEFT','LongInt'( 0);
12678:   'bs_Solid','LongInt'( 0);
12679:   'teepf24Bit','LongInt'( 0);
12680:   'teepfDevice','LongInt'( 1);
12681:   'CM_MOUSELEAVE','LongInt'( 10000);
12682:   'CM_SYSCOLORCHANGE','LongInt'( 10001);
12683:   'DC_BRUSH','LongInt'( 18);
12684:   'DC_PEN','LongInt'( 19);
12685:   teeCOLORREF, 'LongWord
12686:   TLogBrush', 'record lbStyle : Int; lbColor : TColor; lbHatch: Int; end
12687:   //TNotifyEvent', 'Proc ( Sender : TObject)
12688:   SIRegister_TFilterRegion(CL);
12689:   SIRegister_IFormCreator(CL);
12690:   SIRegister_TTeeFilter(CL);
12691:   //TFilterClass', 'class of TTeeFilter
12692:   SIRegister_TFilterItems(CL);
12693:   SIRegister_TConvolveFilter(CL);
12694:   SIRegister_TBlurFilter(CL);
12695:   SIRegister_TTeePicture(CL);
12696:   TPenEndStyle', '( esRound, esSquare, esFlat )
12697:   SIRegister_TChartPen(CL);
12698:   SIRegister_TChartHiddenPen(CL);
12699:   SIRegister_TDottedGrayPen(CL);
12700:   SIRegister_TDarkGrayPen(CL);
12701:   SIRegister_TWhitePen(CL);
12702:   SIRegister_TChartBrush(CL);
12703:   TTeeView3DScrolled', 'Proc ( IsHoriz :Bool)
12704:   TTeeView3DChangedZoom', 'Proc ( NewZoom : Int)
12705:   SIRegister_TView3DOptions(CL);
12706:   FindClass('TOBJECT'),'TTeeCanvas
12707:   TTeeTransparency', 'Int
12708:   SIRegister_TTeeBlend(CL);
12709:   FindClass('TOBJECT'),'TCanvas3D
12710:   SIRegister_TTeeShadow(CL);
12711:   teeTGradientDirection', '(gdTopBottom,gdBottomTop,gdLeftRight,gdRightLeft,gdFromCenter, gdFromTopLeft,
gdFromBottomLeft,gdRadial,gdDiagonalUp,gdDiagonalDown )
12712:   FindClass('TOBJECT'),'TSubGradient
12713:   SIRegister_TCustomTeeGradient(CL);
12714:   SIRegister_TSubGradient(CL);
12715:   SIRegister_TTeeGradient(CL);

```

```

12716: SIRegister_TTeeFontGradient(CL);
12717: SIRegister_TTeeFont(CL);
12718: TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12719: TCanvasTextAlign', 'Int
12720: TTeeCanvasHandle', 'HDC
12721: SIRegister_TTeeCanvas(CL);
12722: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12723: SIRegister_TFloatXYZ(CL);
12724: TPoint3D', 'record x : Int; y : Int; z : Int; end
12725: TRGB', 'record blue: byte; green: byte; red: byte; end
12726: {TRGB=packed record
12727:   Blue : Byte;
12728:   Green : Byte;
12729:   Red : Byte;
12730: //IFDEF CLX //Alpha : Byte; // Linux end;}
12731:
12732: TTeeCanvasCalcPoints', 'Function(x,z:Int;var P0,P1:TPoint3D;var Color0,Color1:TColor):Bool
12733: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12734: TCanvas3DPlane', '( cpX, cpY, cpZ )
12735: //IInterface', 'IUnknown
12736: SIRegister_TCanvas3D(CL);
12737: SIRegister_TTeeCanvas3D(CL);
12738: TTrianglePoints', 'Array[0..2] of TPoint;
12739: TFourPoints', 'Array[0..3] of TPoint;
12740: Func ApplyDark( Color : TColor; HowMuch : Byte ) : TColor
12741: Func ApplyBright( Color : TColor; HowMuch : Byte ) : TColor
12742: Func Point3D( const x, y, z : Int ) : TPoint3D
12743: Proc SwapDouble( var a, b : Double)
12744: Proc SwapInt( var a, b : Int)
12745: Proc RectSize( const R : TRect; var RectWidth, RectHeight : Int)
12746: Proc teeRectCenter( const R : TRect; var X, Y : Int)
12747: Func RectFromPolygon( const Points : array of TPoint; NumPoints : Int): TRect
12748: Func RectFromTriangle( const Points : TTrianglePoints) : TRect
12749: Func RectangleInRectangle( const Small, Big : TRect) :Bool
12750: Proc ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12751: Proc UnClipCanvas( ACanvas : TCanvas)
12752: Proc ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12753: Proc ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Int)
12754: Proc ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Int)
12755: 'TeeCharForHeight','String 'W
12756: 'DarkerColorQuantity','Byte').SetUInt( 128);
12757: 'DarkColorQuantity','Byte').SetUInt( 64);
12758: TButtonGetColorProc', 'Func : TColor
12759: SIRegister_TTeeButton(CL);
12760: SIRegister_TButtonColor(CL);
12761: SIRegister_TComboFlat(CL);
12762: Proc TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12763: Func TeePoint( const aX, aY : Int ) : TPoint
12764: Func TEETPointInRect( const Rect : TRect; const P : TPoint) :Bool;
12765: Func PointInRect1( const Rect : TRect; x, y : Int) :Bool;
12766: Func TeeRect( Left, Top, Right, Bottom : Int) : TRect
12767: Func OrientRectangle( const R : TRect) : TRect
12768: Proc TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Int)
12769: Func PolygonBounds( const P : array of TPoint) : TRect
12770: Func PolygonInPolygon( const A, B : array of TPoint) :Bool
12771: Func RGBValue( const Color : TColor) : TRGB
12772: Func EditColor( AOwner : TComponent; AColor : TColor) : TColor
12773: Func EditColorDialog( AOwner : TComponent; var AColor : TColor) :Bool
12774: Func PointAtDistance( AFrom, ATo : TPoint; ADist : Int) : TPoint
12775: Func TeeCull( const P : TFourPoints) :Bool;
12776: Func TeeCull1( const P0, P1, P2 : TPoint) :Bool;
12777: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12778: Proc SmoothStretch( Src, Dst : TBitmap);
12779: Proc SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12780: Func TeeDistance( const x, y : Double) : Double
12781: Func TeeLoadLibrary( const FileName :Str) : HInst
12782: Proc TeeFreeLibrary( hLibModule : HMODULE)
12783: Proc TeeBlendBitmaps(const Percent:Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12784: //Proc TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12785: Proc TeeShadowSmooth(Bitmap,Back:TBitmap;Left,Top,Width,Height,horz,
vert:Int;Smoothness:Double;FullDraw:Bool;ACanvas:TCanvas3D;Clip:Bool)
12786: SIRegister_ICanvasHyperlinks(CL);
12787: SIRegister_ICanvasToolTips(CL);
12788: Func Supports( const Instance : IInterface; const IID : TGUID) :Bool
12789: end;
12790:
12791: Proc SIRegister_ovcmisc(CL: TPSPascalCompiler);
12792: begin
12793: TOvcHdc', 'Int
12794: TOvcHWND', 'Cardinal
12795: TOvcHdc', 'HDC
12796: TOvcHWND', 'HWND
12797: Func LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12798: Func LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12799: Func ovCompStruct( const S1, S2, Size :Card) : Int
12800: Func DefaultEpoch : Int
12801: Func DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12802: Proc FixRealPrim( P : PChar; DC : Char)
12803: Func GetDisplayString(Canvas: TCanvas; const S:Str; MinChars, MaxWidth : Int) :Str

```

```

12804: Func GetLeftButton : Byte
12805: Func GetNextDlgItem( Ctrl : TOvcHwnd ) : hWnd
12806: Proc GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12807: Func GetShiftFlags : Byte
12808: Func ovCreateRotatedFont( F : TFont; Angle : Int ) : hFont
12809: Func GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Int;Ctl3D:Boolean): Int
12810: Func ovExtractWord( N : Int; const S :Str; WordDelims : TCharSet) :Str
12811: Func ovIsForegroundTask :Bool
12812: Func ovTrimLeft( const S :Str) :Str
12813: Func ovTrimRight( const S :Str) :Str
12814: Func ovQuotedStr( const S :Str) :Str
12815: Func ovWordCount( const S :Str; const WordDelims : TCharSet) : Int
12816: Func ovWordPosition(const N:Int;const S:Str;const WordDelims : TCharSet) : Int
12817: Func PtrDiff( const P1, P2 : PChar) : Word
12818: Proc PtrInc( var P, Delta : Word)
12819: Proc PtrDec( var P, Delta : Word)
12820: Proc FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Int)
12821: Proc TransStretchBlt(DstDC:TOvcHdc;DstX,DstY,DstW,DstH:Int;SrcDC:TOvcHdc;SrcX,SrcY,SrcW,SrcH:Int;MaskDC:TOvcHdc;MaskX,MaskY:Int)
12822: Func ovMinI( X, Y : Int) : Int
12823: Func ovMaxI( X, Y : Int) : Int
12824: Func ovMinL( X, Y : LongInt) : LongInt
12825: Func ovMaxL( X, Y : LongInt) : LongInt
12826: Func GenerateComponentName( PF : TWinControl; const Root :Str) :Str
12827: Func PartialCompare( const S1, S2 :Str) :Bool
12828: Func PathEllipsis( const S :Str; MaxWidth : Int) :Str
12829: Func ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12830: Proc ovCopyParentImage( Control : TControl; Dest : TCanvas)
12831: Proc ovDrawTransparentBitmap(Dest:TCanvas;X,Y,W,H:Int;Rect:TRect;Bitmap:TBitmap; TransparentColor:TColor)
12832: Proc DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;TransparentColor : TColorRef)
12833: Func ovWidthOf( const R : TRect) : Int
12834: Func ovHeightOf( const R : TRect) : Int
12835: Proc ovDebugOutput( const S :Str)
12836: Func GetArrowWidth( Width, Height : Int) : Int
12837: Proc StripCharSeq( CharSeq :Str; var Str :Str)
12838: Proc StripCharFromEnd( aChr : Char; var Str :Str)
12839: Proc StripCharFromFront( aChr : Char; var Str :Str)
12840: Func SystemParametersInfo(uiAction,uiParam:UINT;pvParam : UINT; fWinIni : UINT) : BOOL
12841: Func SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12842: Func SystemParametersInfoA(uiAction,uiParam:UINT;pvParam:UINT;fWinIni : UINT) : BOOL
12843: Func CreateEllipticRgn( p1, p2, p3, p4 : Int) : HRGN
12844: Func CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12845: Func CreateFontIndirect( const p1 : TLogFont) : HFONT
12846: Func CreateMetaFile( p1 : PChar) : HDC
12847: Func DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12848: Func DrawText(hDC:HDC;lpString:PChar;nCount:Int;var lpRect: TRect;uFormat:UINT):Int
12849: Func DrawTextS(hDC:HDC;lpString:str;nCount:Int;var lpRect:TRect;uFormat:UINT):Int
12850: Func SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12851: Func SetGraphicsMode( hdc : HDC; iMode : Int) : Int
12852: Func SetMapMode( DC : HDC; p2 : Int) : Int
12853: Func SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12854: //Func SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12855: Func SetPixel( DC : HDC; X, Y : Int; Color : COLORREF) : COLORREF
12856: Func SetPixelV( DC : HDC; X, Y : Int; Color : COLORREF) : BOOL
12857: //Func SetPixelFormat(DC:HDC;PixelFormat:Int;FormatDef:PPixelFormatDescriptor) : BOOL
12858: Func SetPolyFillMode( DC : HDC; PolyFillMode : Int) : Int
12859: Func StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,SrcHeight:Int;Rop:DWORD):BOOL
12860: Func SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Int) : BOOL
12861: Func StretchDIBits(DC:HDC;DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,SrcHeight:Int;Bits:int;var BitsInfo:TBitmapInfo;Usage:UINT;Rop:DWORD):Int
12862: Func SetROP2( DC : HDC; p2 : Int) : Int
12863: Func SetStretchBltMode( DC : HDC; StretchMode : Int) : Int
12864: Func SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12865: Func SetTextCharacterExtra( DC : HDC; CharExtra : Int) : Int
12866: Func SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12867: Func SetTextAlign( DC : HDC; Flags : UINT) : UINT
12868: Func SetTextJustification( DC : HDC; BreakExtra, BreakCount : Int) : Int
12869: Func UpdateColors( DC : HDC) : BOOL
12870: Func GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12871: Func GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12872: Func GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12873: Func GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12874: Func IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Int) : Int
12875: Func InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12876: Func MaskBlt(DestDC:HDC;XDest,YDest,Width,Height:Int;SrcDC:HDC;XSrc,YSrc:Int; Mask:HBITMAP;xMask,yMask:Int;Rop: DWORD) : BOOL
12877: Func PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,yMask:Int):BOOL;
12878: Func OffsetClipRgn( DC : HDC; XOffset, YOffset : Int) : Int
12879: Func OffsetRgn( RGN : HRGN; XOffset, YOffset : Int) : Int
12880: Func PatBlt( DC : HDC; X, Y, Width, Height : Int; Rop : DWORD) : BOOL
12881: Func Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Int) : BOOL
12882: Func PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12883: Func PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12884: Func PtInRegion( RGN : HRGN; X, Y : Int) : BOOL
12885: Func PtVisible( DC : HDC; X, Y : Int) : BOOL
12886: Func RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12887: Func RectVisible( DC : HDC; const Rect : TRect) : BOOL

```

```

12888: Func Rectangle( DC : HDC; X1, Y1, X2, Y2 : Int) : BOOL
12889: Func RestoreDC( DC : HDC; SavedDC : Int) : BOOL
12890: end;
12891:
12892: Proc SIRegister_ovcfilel(CL: TPSPascalCompiler);
12893: begin
12894: SIRegister_TOvcAbstractStore(CL);
12895: //PEXPropInfo, '^TExPropInfo // will not work
12896: // TExPropInfo, 'record PI : TPropInfo; AObject : TObject; end
12897: SIRegister_TOvcPropertyList(CL);
12898: SIRegister_TOvcDataFiler(CL);
12899: Proc UpdateStoredList(AForm: TWinControl; AStoredList : TStrings; FromForm:Bool)
12900: Proc UpdateStoredList1( AForm : TCustomForm; AStoredList: TStrings;FromForm:Bool)
12901: Func CreateStoredItem( const CompName, PropName :Str) :Str
12902: Func ParseStoredItem( const Item :Str; var CompName, PropName :Str) :Bool
12903: //Func GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12904: end;
12905:
12906: Proc SIRegister_ovccoco(CL: TPSPascalCompiler);
12907: begin
12908: 'ovsetsize', 'LongInt'( 16);
12909: 'etSyntax', 'LongInt'( 0);
12910: 'etSymantic', 'LongInt'( 1);
12911: 'chCR', Char #13;
12912: 'chLF', Char #10;
12913: 'chLineSeparator', chCR;
12914: SIRegister_TCCocoError(CL);
12915: SIRegister_TCommentItem(CL);
12916: SIRegister_TCommentList(CL);
12917: SIRegister_TSymbolPosition(CL);
12918: TGenListType, '( glNever, glAlways, glOnError )
12919: TovBitSet, 'set of Int
12920: //PStartTable, '^TStartTable // will not work
12921: TovCharSet, 'set of AnsiChar
12922: TAfterGenListEvent', 'Proc ( Sender : TObject; var PrintErrorCount :Bool)
12923: TCommentEvent', 'Proc ( Sender : TObject; CommentList : TCommentList)
12924: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode:longint;const Data:str):str
12925: TovErrorEvent', 'Proc ( Sender : TObject; Error : TCocoError)
12926: TovErrorProc', 'Proc (ErrorCode:Int;Symbol:TSymbolPosition;const Data:str;ErrorType:Int)
12927: TFailureEvent', 'Proc (Sender : TObject; NumErrors : Int)
12928: TGetCH', 'Func (pos : longint) : char
12929: TStatusUpdateProc', 'Proc ( Sender : TObject; Status:Str; LineNum:Int)
12930: SIRegister_TCCocoRScanner(CL);
12931: SIRegister_TCCocoRGrammar(CL);
12932: ' _EF', 'Char #0);
12933: ' _TAB', 'Char').SetString( #09);
12934: ' _CR', 'Char').SetString( #13);
12935: ' _LF', 'Char').SetString( #10);
12936: ' _EL', ' ').SetString( _CR);
12937: ' _EOF', 'Char').SetString( #26);
12938: 'LineEnds', 'TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12939: 'minErrDist', 'LongInt'( 2);
12940: Func ovPadL(S:Str; ch : char; L : Int) :Str
12941: end;
12942:
12943: TFormatSettings = record
12944: CurrencyFormat: Byte;
12945: NegCurrFormat: Byte;
12946: ThousandSeparator: Char;
12947: DecimalSeparator: Char;
12948: CurrencyDecimals: Byte;
12949: DateSeparator: Char;
12950: TimeSeparator: Char;
12951: ListSeparator: Char;
12952: CurrencyString:Str;
12953: ShortDateFormat:Str;
12954: LongDateFormat:Str;
12955: TimeAMString:Str;
12956: TimePMString:Str;
12957: ShortTimeFormat:Str;
12958: LongTimeFormat:Str;
12959: ShortMonthNames: array[1..12] of string;
12960: LongMonthNames: array[1..12] of string;
12961: ShortDayNames: array[1..7] of string;
12962: LongDayNames: array[1..7] of string;
12963: TwoDigitYearCenturyWindow: Word;
12964: end;
12965:
12966: Proc SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12967: begin
12968: Func ovFormatSettings : TFormatSettings
12969: end;
12970:
12971: Proc SIRegister_ovcstr(CL: TPSPascalCompiler);
12972: begin
12973: TOvcCharSet, 'set of Char
12974: ovBTable, 'array[0..255] of Byte
12975: //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;

```



```

12976: Func BinaryBPChar( Dest : PChar; B : Byte) : PChar
12977: Func BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12978: Func BinaryWPChar( Dest : PChar; W : Word) : PChar
12979: Proc BMMakeTable( MatchString : PChar; var BT : ovBTable)
12980: Func BMSearch(var Buffer, BufLength:Card; var BT:ovBTable; MatchString:PChar; var Pos:Card):Bool;
12981: Func BMSearchUC(var Buffer, BufLength:Card; var BT:ovBTable; MatchString:PChar; var Pos:Card):Bool
12982: Func CharStrPChar( Dest : PChar; C : Char; Len :Card) : PChar
12983: Func DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12984: Func HexBPChar( Dest : PChar; B : Byte) : PChar
12985: Func HexLPChar( Dest : PChar; L : LongInt) : PChar
12986: Func HexPtrPChar( Dest : PChar; P : TObject) : PChar
12987: Func HexWPChar( Dest : PChar; W : Word) : PChar
12988: Func LoCaseChar( C : Char) : Char
12989: Func OctallPChar( Dest : PChar; L : LongInt) : PChar
12990: Func StrChDeletePrim( P : PChar; Pos :Card) : PChar
12991: Func StrChInsertPrim( Dest : PChar; C : Char; Pos:Card) : PChar
12992: Func StrChPos( P : PChar; C : Char; var Pos :Card) :Bool
12993: Proc StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12994: Func StrStCopy( Dest, S : PChar; Pos, Count :Card) : PChar
12995: Func StrStDeletePrim( P : PChar; Pos, Count :Card) : PChar
12996: Func StrStInsert( Dest, S1, S2 : PChar; Pos :Card) : PChar
12997: Func StrStInsertPrim( Dest, S : PChar; Pos :Card) : PChar
12998: Func StrStPos( P, S : PChar; var Pos :Card) :Bool
12999: Func StrToLongPChar( S : PChar; var I : LongInt) :Bool
13000: Proc TrimAllSpacesPChar( P : PChar)
13001: Func TrimEmbeddedZeros( const S :Str) :Str
13002: Proc TrimEmbeddedZerosPChar( P : PChar)
13003: Func TrimTrailPrimPChar( S : PChar) : PChar
13004: Func TrimTrailPChar( Dest, S : PChar) : PChar
13005: Func TrimTrailingZeros( const S :Str) :Str
13006: Proc TrimTrailingZerosPChar( P : PChar)
13007: Func UpCaseChar( C : Char) : Char
13008: Func ovcCharInSet( C : Char; const CharSet : TOvcCharSet) :Bool
13009: Func ovc32StringIsCurrentCodePage( const S : WideString) :Bool;
13010: //Func ovc32StringIsCurrentCodePage( const S:PWideChar; CP :Card) :Bool;
13011: end;
13012:
13013: Proc SIRegister_AfUtils( CL: TPSPascalCompiler);
13014: begin
13015:   //PRaiseFrame', '^TRaiseFrame // will not work
13016:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
13017:   +ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
13018: Proc SafeCloseHandle( var Handle : THandle)
13019: Proc ExchangeInt( X1, X2 : Int)
13020: Proc FillInt( const Buffer, Size, Value : Int)
13021: Func LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
13022: Func afCompareMem( P1, P2 : TObject; Length : Int) :Bool
13023:
13024: FILENAME_ADVAPI32 = 'ADVAPI32.DLL';
13025: Func AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
13026: Func AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
13027: Func AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
13028:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
13029:   SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
13030:   const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
13031:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
13032: Func AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
13033:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
13034:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
13035:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
13036:   ObjectTypeInfoLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
13037:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
13038: Func AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
13039:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
13040:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
13041:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
13042:   ObjectTypeInfoLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
13043:   var GrantedAccess: DWORD; var AccessStatusList: DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
13044: Func BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
13045: Func ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
13046: Func CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
13047:   lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
13048:   lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
13049:   dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
13050:   const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
13051: Func GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
13052: Func GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
13053:   pSecurityDescriptor: PSecurityDescriptor; nLength: DWORD; var lpnLengthNeeded: DWORD): BOOL; stdcall;
13054: Func GetUserNames(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
13055: Func InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
13056:   dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
13057: Func LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
13058:   dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
13059: Func LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
13060:   Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
13061:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
13062: Func LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
13063:   Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
13064:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;

```

```

13065: Func LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
13066:   lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
13067: Func LookupPrivilegeName(lpSystemName: PKOLChar;
13068:   var lpLuid: TLargeInt; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
13069: Func LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
13070:   var lpLuid: TLargeInt): BOOL; stdcall;
13071: Func ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
13072:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
13073: Func ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
13074:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
13075: Func ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
13076:   ObjectType: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
13077:   ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
13078:   var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
13079:   var GenerateOnClose: BOOL): BOOL; stdcall;
13080: Func ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
13081:   HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
13082:   var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
13083: Func OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
13084: Func OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
13085: Func PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
13086:   ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
13087: Func ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
13088:   lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
13089:   var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
13090: Func RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
13091:   var phkResult: HKEY): Longint; stdcall;
13092: Func RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
13093:   var phkResult: HKEY): Longint; stdcall;
13094: Func RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
13095:   Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
13096:   lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
13097:   lpdwDisposition: PDWORD): Longint; stdcall;
13098: Func RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
13099: Func RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
13100: Func RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
13101:   var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
13102:   lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
13103: Func RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD): Longint; stdcall;
13104: Func RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
13105:   var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
13106:   lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
13107: Func RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
13108: Func RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
13109: Func RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
13110:   ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
13111: Func RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
13112:   lpcbClass: PDWORD; lpReserved: Pointer;
13113:   lpSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
13114:   lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
13115:   lpftLastWriteTime: PFileTime): Longint; stdcall;
13116: Func RegQueryMultipleValues(hKey: HKEY; var Vallist;
13117:   NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotSize: DWORD): Longint; stdcall;
13118: Func RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
13119:   lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
13120: Func RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
13121:   lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
13122: Func RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
13123:   lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
13124: Func RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
13125: Func RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
13126:   lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
13127: Func RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
13128:   dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
13129: Func RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
13130:   Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
13131: Func RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
13132: Func RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
13133: Func ReportEvent(hEventLog: THandle; wType, wCategory: Word;
13134:   dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
13135:   dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
13136: Func SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
13137:   pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
13138: Func wAddAtom(lpString: PKOLChar): ATOM
13139: Func wBeginUpdateResource(pFileName: PKOLChar; bDeleteExistingResources: BOOL): THandle
13140: //Func
wCallNamedPipe(lpNamedPipeName: PKOLChar; lpInBuffer: Pointer; nInBufferSize: DWORD; lpOutBuffer: Pointer; nOutBuff
lpBytesRead: DWORD; nTimeOut: DWORD): BOOL
13141: //Func wCommConfigDialog(lpszName: PKOLChar; hWnd: HWND; var lpCC: TCommConfig): BOOL
13142: Func
wCompareString(Locale: LCID; dwCmpFlags: DWORD; lpString1: PKOLChar; cchCount1: Int; lpString2: PKOLChar; cchCount2: Int): Int
13143: Func wCopyFile(lpExistingFileName, lpNewFileName: PKOLChar; bFailIfExists: BOOL): BOOL
13144: //Func wCopyFileEx(lpExistingFileName,
lpNewFileName: PKOLChar; lpProgressRoutine: TFNProgressRoutine; lpData: Pointer; pbCancel: PBool; dwCopyFlags: DWORD
13145: Func wCreateDirectory(lpPathName: PKOLChar; lpSecurityAttributes: PSecurityAttributes): BOOL
13146: Func wCreateDirectoryEx(lpTemplateDirectory, lpNewDirectory: PKOLChar; lpSecAttrib: PSecurityAttributes): BOOL;
13147: Func wCreateEvent(lpEventAttribs: PSecurityAttributes; bManualReset,
bInitialState: BOOL; lpName: PKOLChar): THandle;
13148: Func wCreateFile(lpFileName: PKOLChar; dwDesiredAccess, dwShareMode: DWORD; lpSecurityAttributes:
PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes: DWORD; hTemplateFile: THandle): THandle

```

```

13149: Func wCreateFileMapping(hFile:THandle;lpFileMappingAttributes:PSecurityAttributes; flProtect,
dwMaximumSizeHigh,dwMaximumSizeLow:DWORD;lpName:PKOLChar):THandle
13150: Func wCreateHardLink(lpFileName,lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
13151: Func
CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle;
13152: Func wCreateNamedPipe(lpName:PKOLChar;dwOpenMode,dwPipeMode,nMaxInstances,nOutBufferSize,nInBufferSize,
nDefaultTimeOut:DWORD;lpSecurityAttributes:PSecurityAttributes): THandle
13153: //Func CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
lpProcessInfo:TProcessInformation):BOOL
13154: Func wCreateSemaphore(lpSemaphoreAttributes:PSecurityAttributes;lInitialCount,
lMaximumCount:Longint;lpName:PKOLChar):THandle
13155: Func
wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
13156: Func wDefineDosDevice(dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
13157: Func wDeleteFile( lpFileName : PKOLChar) : BOOL
13158: Func wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
13159: //Func
wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
13160: //Func wEnumDateFormats(lpDateFmtEnumProc:TFNDateFmtEnumProc;Locale:LCID;dwFlags:DWORD): BOOL
13161: //Func wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
13162: //Func wEnumResourceTypes(hModule:HMODULE;lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
13163: //Func wEnumSystemCodePages(lpCodePageEnumProc:TFNCodepageEnumProc;dwFlags:DWORD): BOOL
13164: //Func wEnumSystemLocales(lpLocaleEnumProc:TFNLocaleEnumProc;dwFlags:DWORD): BOOL
13165: //Func wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
13166: Func wExpandEnvironmentStrings(lpSrc:PKOLChar;lpDst:PKOLChar;nSize:DWORD): DWORD
13167: Proc wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
13168: //Func wFillConsoleOutputCharacter(hConsoleOutput:THandle;cCharacter:KOLChar;nLength:DWORD; dwWriteCoord:
TCoord;var lpNumberOfCharsWritten:DWORD):BOOL
13169: Func wFindAtom( lpString : PKOLChar) : ATOM
13170: Func wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
13171: Func wFindFirstFile(lpFileName:PKOLChar;var lpFindFileData:TWIN32FindData): THandle
13172: //Func wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindIndexInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFindIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
13173: Func wFindNextFile(hFindFile:THandle;var lpFindFileData:TWIN32FindData):BOOL
13174: Func wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
13175: Func wFindResourceEx(hModule:HMODULE;lpType,lpName:PKOLChar;wLanguage:Word) : HRSRC
13176: Func wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
13177: //Func wFormatMessage(dwFlags:DWORD;lpSource:Pointer;dwMessageId:DWORD;dwLanguageId:DWORD;
lpBuffer:PKOLChar;nSize:DWORD;Arguments:Pointer):DWORD
13178: Func wFreeEnvironmentStrings( EnvBlock : PKOLChar): BOOL
13179: Func wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Int) : UINT
13180: Func wGetBinaryType(lpApplicationName:PKOLChar;var lpBinaryType:DWORD):BOOL
13181: Func wGetCommandLine: PKOLChar
13182: //Func wGetCompressedFileSize(lpFileName:PKOLChar;lpFileSizeHigh:PDWORD): DWORD
13183: Func wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
13184: Func wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
13185: //Func wGetCurrencyFormat(Locale:LCID;dwFlags:DWORD;lpValue:PKOLChar;lpFormat :
PCurrencyFmt;lpCurrencyStr:PKOLChar;cchCurrency:Int):Int
13186: Func wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
13187: //Func wGetDateFormat(Locale:LCID; dwFlags:DWORD;lpDate:PSystemTime;lpFormat: PKOLChar;
lpDateStr:PKOLChar; cchDate: Int):Int
13188: //Func wGetDefaultCommConfig(lpszName:PKOLChar;var lpCC:TCommConfig;var lpdwSize:DWORD):BOOL
13189: Func wGetDiskFreeSpace(lpRootPathName:PKOLChar;var lpSectorsPerCluster,lpBytesPerSector,
lpNumberOfFreeClusters,lpTotalNumberOfClusters:DWORD):BOOL
13190: //Func wGetDiskFreeSpaceEx(lpDirectoryName:PKOLChar;var lpFreeBytesAvailableToCaller,lpTotalNumberOfBytes,
lpTotalNumberOfFreeBytes:PLargeInt):BOOL
13191: Func wGetDriveType( lpRootPathName : PKOLChar) : UINT
13192: Func wGetEnvironmentStrings : PKOLChar
13193: Func wGetEnvironmentVariable(lpName:PKOLChar;lpBuffer:PKOLChar;nSize:DWORD): DWORD;
13194: Func wGetFileAttributes( lpFileName : PKOLChar) : DWORD
13195: //Func
wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
13196: Func wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
13197: //Func wGetLocaleInfo(Locale:LCID;LCType:LCTYPE;lpLCData:PKOLChar;cchData:Int):Int
13198: Func wGetLogicalDriveStrings(nBufferLength:DWORD;lpBuffer:PKOLChar):DWORD
13199: Func wGetModuleFileName(hModule:HINST;lpFilename:PKOLChar;nSize:DWORD):DWORD
13200: Func wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
13201: //Func wGetNamedPipeHandleState(hNamedPipe: THandle;lpState,lpCurInstances,lpMaxCollectionCount,
lpCollectDataTimeout:PDWORD;lpUserName:PKOLChar;nMaxUserNameSize:DWORD): BOOL
13202: //Func
wGetNumberFormat(Locale:CID;dwFlags:DWORD;lpValue:PKOLChar;lpFormat:PNumberFmt;lpNumberStr:PKOLChar;cchNumb
13203: Func wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Int;lpFileName:PKOLChar):UINT;
13204: Func
wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
13205: Func wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
13206: Func wGetPrivateProfileString(lpAppName,lpKeyName,
lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD
13207: Func wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Int) : UINT
13208: Func wGetProfileSection(lpAppName:PKOLChar;lpReturnedString:PKOLChar;nSize:DWORD):DWORD
13209: Func wGetProfileString(lpAppName,lpKeyName,lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
13210: Func wGetShortPathName(lpszLongPath:PKOLChar;lpszShortPath:PKOLChar;cchBuffer:DWORD):DWORD
13211: //Proc wGetStartupInfo( var lpStartupInfo : TStartupInfo)
13212: //Func wGetStringTypeEx(Locale:LCID;dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;var lpCharType):BOOL
13213: Func wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
13214: Func wGetTempFileName(lpPathName,lpPrefixString:PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
13215: Func wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD

```

```

13216: //Func
wGetTimeFormat( Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
13217: //Func wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13218: //Func GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize :
DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength;lpFileSystemFlags:DWORD;
lpFileSystemNameBuffer:PKOLChar;nFileSystemNameSize: DWORD) : BOOL
13219: Func wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
13220: Func wGlobalAddAtom( lpString : PKOLChar) : ATOM
13221: Func wGlobalFindAtom( lpString : PKOLChar) : ATOM
13222: Func wGlobalGetAtomName( nAtom:ATOM;lpBuffer:PKOLChar;nSize:Int):UINT
13223: Func wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
13224: Func
wLCMapString( Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
13225: Func wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
13226: Func wLoadLibraryEx( lpLibFileName:PKOLChar;hFile:THandle;dwFlags:DWORD) : HMODULE
13227: Func wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
13228: Func wMoveFileEx( lpExistingFileName, lpNewFileName:PKOLChar;dwFlags:DWORD) : BOOL
13229: //Func wMoveFileWithProgress( lpExistingFileName, lpNewFileName:PKOLChar;lpProgressRoutine:
TFNProgressRoutine;lpData:Pointer;dwFlags:DWORD) : BOOL
13230: Func wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
13231: Func wOpenFileMapping( dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpName : PKOLChar):THandle
13232: Func wOpenMutex( dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpName:PKOLChar) : THandle
13233: Func wOpenSemaphore( dwDesiredAccess:DWORD; bInheritHandle:BOOL;lpName:PKOLChar):THandle
13234: Func wOpenWaitableTimer( dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
13235: Proc wOutputDebugString( lpOutputString : PKOLChar)
13236: //Func wPeekConsoleInput( hConsoleInput:THandle;varlpBuffer:TInputRecord;nLength:DWORD;var
lpNumberOfEventsRead:DWORD) :BOOL;
13237: Func wQueryDosDevice( lpDeviceName:PKOLChar;lpTargetPath:PKOLChar;ucchMax:DWORD) :DWORD
13238: //Func wQueryRecoveryAgents( pl:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
13239: //Func wReadConsole( hConsoleInput:THandle;lpBuffer:Pointer;nNumberOfCharsToRead:DWORD; var
lpNumberOfCharsRead:DWORD;lpReserved:Pointer) :BOOL
13240: //Func wReadConsoleInput( hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
lpNumbOfEventsRead:DWORD) :BOOL;
13241: //Func wReadConsoleOutput( hConsoleOutput:THandle;lpBuffer:Pointer;dwBufferSize,dwBufferCoord:TCoord;var
lpReadRegion:TSmallRect) : BOOL
13242: //Func wReadConsoleOutputCharacter( hConsoleOutput:THandle;lpCharacter:PKOLChar;nLength :
DWORD;dwReadCoord:TCoord;var lpNumberOfCharsRead:DWORD) : BOOL
13243: Func wRemoveDirectory( lpPathName : PKOLChar) : BOOL
13244: //Func wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
lpClipRectangle:PSmallRect;dwDestinationOrigin:TCoord;var lpFill:TCharInfo):BOOL
13245: Func wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar) :DWORD;
13246: Func wSetComputerName( lpComputerName: PKOLChar) : BOOL
13247: Func wSetConsoleTitle( lpConsoleTitle: PKOLChar) : BOOL
13248: Func wSetCurrentDirectory( lpPathName: PKOLChar) : BOOL
13249: //Func wSetDefaultCommConfig( lpszName: PKOLChar; lpCC:PCommConfig; dwSize: DWORD) : BOOL
13250: Func wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
13251: Func wSetFileAttributes( lpFileName:PKOLChar;dwFileAttributes:DWORD) : BOOL
13252: //Func wSetLocaleInfo( Locale:LCID;LCTYPE LCTYPE;lpLCData:PKOLChar) : BOOL
13253: Func wSetVolumeLabel( lpRootPathName:PKOLChar;lpVolumeName:PKOLChar) : BOOL
13254: //Func wUpdateResource( hUpdate:THandle;lType,lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD) :BOOL
13255: Func wVerLanguageName( wLang:DWORD;szLang:PKOLChar;nSize:DWORD) :DWORD
13256: Func wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
13257: //Func wWriteConsole( hConsoleOutput:THandle;const lpBuffer:Pointer;nNumberOfCharsToWrite:DWORD;var
lpNumberOfCharsWritten:DWORD;lpReserved:Pointer) :BOOL
13258: //Func wWriteConsoleInput( hConsoleInput:THandle;const lpBuffer:TInputRecord;nLength:DWORD;var
lpNumberOfEventsWritten:DWORD) :BOOL
13259: //Func wWriteConsoleOutput( hConsoleOutput:THandle;lpBuffer:Pointer;dwBufferSize,dwBufferCoord:TCoord;var
lpWriteRegion:TSmallRect) :BOOL
13260: //Func
wWriteConsoleOutputCharacter( hConsoleOutput:THandle;lpCharacter:PKOLChar;nLength:DWORD;dwWriteCoord:TCoord;
lpNumberOfCharsWritten:DWORD) :BOOL
13261: Func wWritePrivateProfileSection( lpAppName,lpString,lpFileName:PKOLChar) :BOOL
13262: Func wWritePrivateProfileString( lpAppName,lpKeyName,lpString,lpFileName:PKOLChar) :BOOL
13263: Func wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
13264: Func wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
13265: Func wlstcat( lpString1, lpString2 : PKOLChar) : PKOLChar
13266: Func wlstcmp( lpString1, lpString2 : PKOLChar) : Int
13267: Func wlstcmpi( lpString1, lpString2 : PKOLChar) : Int
13268: Func wlstcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
13269: Func wlstcpyi( lpString1, lpString2 : PKOLChar; iMaxLength : Int) : PKOLChar
13270: Func wlstlen( lpString : PKOLChar) : Int
13271: //Func
wMultinetGetConnectionPerformance( lpNetResource:PNetResource;lpNetConnectInfoStruc:PNetConnectInfoStruct) :D
13272: //Func wWNetAddConnection2( var lpNetResource:TNetResource;lpPassw,
lpUserName:PKOLChar;dwFlags:DWORD) :DWORD;
13273: //Func wWNetAddConnection3( hwndOwner:HWND;var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar;dwFlags:DWORD) :DWORD
13274: Func wWNetAddConnection( lpRemoteName, lpPassword,lpLocalName:PKOLChar) :DWORD
13275: Func wWNetCancelConnection2( lpName:PKOLChar;dwFlags:DWORD;fForce:BOOL) :DWORD
13276: Func wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
13277: //Func wWNetConnectionDialog1( var lpConnDlgStruct:TConnectDlgStruct) : DWORD
13278: //Func wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
13279: //Func wWNetEnumResource( hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD) :DWORD;
13280: Func wWNetGetConnection( lpLocalName:PKOLChar;lpRemoteName:PKOLChar;var lpnLength:DWORD) :DWORD;
13281: Func wWNetGetLastError( var lpError:DWORD;lpErrorBuf:PKOLChar;nErrorBufSize: DWORD;
lpNameBuf:PKOLChar;nNameBufSize:DWORD) :DWORD
13282: //Func wWNetGetNetworkInformation( lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct) :DWORD;
13283: Func wWNetGetProviderName( dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD) :DWORD;

```



```

13284: //Func wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
13285: //Func wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
lpBufferSize:DWORD):DWORD;
13286: Func wWNetGetUser(lpName:PKOLChar;lpUserName PKOLChar;var lpnLength:DWORD): DWORD
13287: //Func wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
13288: //Func wWNetSetConnection(lpName:PKOLChar;dwProperties:DWORD; pvValues : Pointer): DWORD
13289: //Func wWNetUseConnection(hwndOwner:HWND;var
lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
lpBufferSize:DWORD;var lpResult:DWORD):DWORD
13290: Func wGetFileVersionInfo(lptstrFilenam:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
13291: Func wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
13292: Func wVerFindFile(uFlags:DWORD;szFileName,szWinDir,szAppDir,szCurDir:PKOLChar;var
lpuCurDirLen:UINT;szDestDir:PKOLChar;var lpuDestDirLen:UINT):DWORD
13293: Func wVerInstallFile(uFlags:DWORD;szSrcFileName,szDestFileName,szSrcDir,szDestDir,szCurDir,
szTmpFile:PKOLChar;var lpuTmpFileLen:UINT):DWORD
13294: //Func wVerQueryValue(pBlock:Pter;lpSubBlock:PKOLChar;varlplpBuffer:Ptr;varpuLen:UINT):BOOL;
13295: //Func wGetPrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
13296: //Func wWritePrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
13297: Func wAddFontResource( FileName : PKOLChar) : Int
13298: //Func wAddFontResourceEx(pl:PKOLChar;p2 DWORD;p3:PDesignVector): Int
13299: Func wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
13300: Func wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
13301: //Func wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
13302: //Func wCreateDC(lpszDriver,lpszDevice,lpszOutput:PKOLChar; lpdvmInit: PDeviceMode) : HDC
13303: //Func wCreateEnhMetaFile(DC:HDC;FileName:PKOLChar;Rect:PREct;Desc: PKOLChar) : HDC
13304: Func wCreateFont( nHeight,nWidth,nEscapement,nOrientation,fnWeight:Int;fdwItalic, fdwUnderline,
fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
13305: Func wCreateFontIndirect( const p1 : TLogFont) : HFONT
13306: //Func wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
13307: // Func wCreateIC(lpszDriver,lpszDevice, lpszOutput:PKOLChar; lpdvmInit:PDeviceMode): HDC
13308: Func wCreateMetaFile( p1 : PKOLChar) : HDC
13309: Func wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
13310: //Func wDeviceCapabilities(pDriverNa,pDeviceNam,
pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
13311: // Func wEnumFontFamilies(DC:HDC; p2: PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
13312: // Func wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL;
13313: //Func wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmproc:TFNFontEnumProc;lpszData:PKOLChar):Int;
13314: //Func wEnumICMProfiles(DC:HDC; ICMProc:TFNICMEnumProc; p3 LPARAM) : Int
13315: //Func wExtTextOut(DC:HDC;X,Y:Int;Options:Longint;Rect:PREct;Str:PKOLChar;Count:Longint;Dx:PInt:BOOL
13316: //Func wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs): BOOL
13317: //Func wGetCharABCWidthsFloat(DC:HDC; FirstChar,LastChar: UINT;const ABCFloatSturcts): BOOL
13318: //Func wGetCharWidth32(DC:HDC;FirstChar,LastChar:UINT; const Widths): BOOL
13319: //Func wGetCharWidth(DC:HDC;FirstChar,LastChar:UINT; const Widths): BOOL
13320: // Func wGetCharWidthFloat(DC:HDC;FirstChar,LastChar UINT;const Widths):BOOL
13321: // Func wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
13322: Func wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
13323: Func wGetEnhMetaFileDescription(p1 HENHMETAFILE;p2:UINT;p3:PKOLChar):UINT
13324: // Func wGetGlyphIndices(DC:HDC;p2:PKOLChar;p3:Int;p4:PWORD;p5:DWORD) : DWORD
13325: //Func wGetGlyphOutline(DC:HDC;uChar,uFormat:UINT;const
lpgm:TGlyphMetrics;cbBuffer:DWORD;lpvBuffer:Pointer;const lpmat2:TMat2): DWORD
13326: Func wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
13327: // Func wGetLogColorSpace(pl:HCOLORSPACE;var ColorSpace:TLogColorSpace;Size:DWORD):BOOL
13328: Func wGetMetaFile( p1 : PKOLChar) : HMETAFILE
13329: // Func wGetObject( p1 : HGDIOBJ; p2 : Int; p3 : Pointer) : Int
13330: //Func wGetOutlineTextMetrics(DC:HDC;p2:UINT;OTMetricStructs:Pointer): UINT
13331: //Func wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Int;p5,p6:PInt;var p7:TSize):BOOL
13332: Func wGetTextExtentPoint32(DC:HDC;Str:PKOLChar;Count:Int;var Size:TSize):BOOL
13333: Func wGetTextExtentPoint(DC:HDC;Str:PKOLChar;Count:Int;var Size:TSize):BOOL
13334: Func wGetTextFace( DC : HDC; Count : Int; Buffer : PKOLChar) : Int
13335: //Func wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
13336: Func wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Int) : BOOL
13337: Func wRemoveFontResource( FileName : PKOLChar) : BOOL
13338: //Func wRemoveFontResourceEx(pl:PKOLChar;p2 DWORD;p3: PDesignVector): BOOL
13339: //Func wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
13340: Func wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
13341: //Func wStartDoc( DC : HDC; const p2 : TDocInfo) : Int
13342: Func wTextOut( DC : HDC; X, Y : Int; Str : PKOLChar; Count : Int) : BOOL
13343: Func wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
13344: Func wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
13345: //Func wwglUseFontOutlines(pl:HDC;p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
13346: Func wAppendMenu(hMenu:HMENU;uFlags,uIDNewItem:UINT;lpNewItem:PKOLChar):BOOL
13347: Func wCallMsgFilter( var lpMsg : TMsg; nCode : Int) : BOOL
13348: //Func wCallWindowProc(lpPrevWndFunc:TFNWndProc;hwnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
13349: //Func wChangeDisplaySettings(var lpDevMode:TDeviceMode; dwFlags : DWORD) : Longint
13350: // Func wChangeDisplaySettingsEx(lpszDeviceName:PKOLChar;var lpDevMode:
TDeviceMode;wnd:HWND;dwFlags:DWORD;lParam:Pointer):Longint
13351: Func wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
13352: Func wCharLower( lpsz : PKOLChar) : PKOLChar
13353: Func wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
13354: Func wCharNext( lpsz : PKOLChar) : PKOLChar
13355: //Func wCharNextEx(CodePage:Word;lpCurrentChar:LPCSTR;dwFlags:DWORD) : LPSTR
13356: Func wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
13357: // Func wCharPrevEx(CodePage:Word;lpStart,lpCurrentChar:LPCSTR; dwFlags:DWORD) : LPSTR
13358: Func wCharToOem(lpszSrc:PKOLChar; lpszDst : PKOLChar) : BOOL
13359: Func wCharToOemBuff(lpszSrc:PKOLChar;lpszDst:PKOLChar;cchDstLength:DWORD) : BOOL
13360: Func wCharUpper( lpsz : PKOLChar) : PKOLChar

```



```

13361: Func wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
13362: Func wCopyAcceleratorTable(hAccelSrc:HACCEL;var lpAccelDst,cAccelEntries:Int) :Int
13363: Func wCreateAcceleratorTable( var Accel, Count : Int) : HACCEL
13364: //Func wCreateDesktop(lpszDesktop,
lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
13365: //Func wCreateDialogIndirectParam(hInstance:HINST;const lpTemplate:TDLgTemplate;hWndParent
:HWND;lpDialogFunc:TFNDlgProc;dwInitParam:LPARAM):HWND
13366: //Func
wCreateDialogParam(hInstance:HINST;lpTemplateName:PKOLChar;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitPa
13367: Func wCreateMDIWindow(lpClassName,lpWindowName:PKOLChar;dwStyle:DWORD;X,Y,nWidth,
nHeight:Int;hWndParent:HWND;hInstance:HINST;lpParam:LPARAM):HWND
13368: //Func wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle :DWORD;X,Y,nWidth,
nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
13369: //Func wCreateWindowStation(lpwinSta:PKOLChar;dwReserv,
dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
13370: Func wDefDlgProc(hDlg:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
13371: Func wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
13372: Func wDefMDIChildProc(hWnd:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
13373: Func wDefWindowProc(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM): RESULT
13374: //Func wDialogBoxIndirectParam(hInstance:HINST;const
lpDialogTemplate:TDLgTemplate;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitParam:LPARAM):Int
13375: //Func
wDialogBoxParam(hInstance:HINST;lpTemplateName:PKOLChar;hWndParent:HWND;lpDialogFunc:TFNDlgProc;dwInitParam
LPARAM): Int
13376: Func wDispatchMessage(const lpMsg : TMsg) : Longint
13377: Func wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,nIDStaticPath:Int;uFileType:UINT):Int;
13378: Func wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,nIDStaticPath:Int;uFileType:UINT):Int;
13379: Func wDlgDirSelectComboBoxEx(hDlg:HWND;lpString:PKOLChar;nCount,nIDComboBox:Int): BOOL
13380: Func wDlgDirSelectEx(hDlg:HWND;lpString:PKOLChar;nCount,nIDListBox:Int):BOOL
13381: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARAM;x,y,cx,
cy:Int;Flags:UINT):BOOL;
13382: Func wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Int;var lpRect:TRect;uFormat:UINT):Int;
13383: Func wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
13384: Func wFindWindowEx(Parent,Child:HWND;ClassName,WindowName:PKOLChar): HWND
13385: //Func wGetAltTabInfo(hwnd:HWND;iItem:Int;var
pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
13386: // Func wGetClassInfo(hInstance:HINST;lpClassName:PKOLChar;var lpWndClass:TWndClass):BOOL
13387: //Func wGetClassInfoEx(Instance:HINST;ClassName:PKOLChar;var WndClass:TWndClassEx):BOOL
13388: Func wGetClassLong( hWnd : HWND; nIndex : Int) : DWORD
13389: Func wGetClassName(hWnd:HWND; lpClassName:PKOLChar;nMaxCount:Int) : Int
13390: Func wGetClipboardFormatName(format:UINT;lpszFormatName:PKOLChar;cchMaxCount:Int):Int;
13391: Func wGetDlgItemText(hDlg:HWND;nIDDlgItem:Int;lpString:PKOLChar;nMaxCount:Int):UINT
13392: Func wGetKeyNameText(lpParam:Longint;lpString:PKOLChar;nSize:Int):Int
13393: Func wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
13394: //Func wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
13395: Func wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Int;uFlag:UINT):Int;
13396: Func wGetMessage(var lpMsg:TMsg;hWnd:HWND;wMsgFilterMin:wMsgFilterMax:UINT):BOOL
13397: Func wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
13398: //Func wGetTabbedTextExtent(hDC:HDC;lpString:PKOLChar;nCount,nTabPositions:Int;var
lpnTabStopPositions):DWORD
13399: //Func wGetObjectInfo(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var lpnLengthNeed:DWORD)BOOL;
13400: Func wGetWindowLong( hWnd : HWND; nIndex : Int) : Longint
13401: Func wGetWindowModuleFileName(hwnd:HWND;pszFileName:PKOLChar;cchFileNameMax:UINT): UINT
13402: Func wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Int) : Int
13403: Func wGetWindowTextLength( hWnd : HWND) : Int
13404: //Func wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
nHeigt:Int):BOOL;
13405: Func wInsertMenu(hMenu:HMENU;uPosition,uFlags,uIDNewItem:UINT; lpNewItem: PKOLChar): BOOL
13406: //Func wInsertMenuItem(p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13407: Func wIsCharAlpha( ch : KOLChar) : BOOL
13408: Func wIsCharAlphaNumeric( ch : KOLChar) : BOOL
13409: Func wIsCharLower( ch : KOLChar) : BOOL
13410: Func wIsCharUpper( ch : KOLChar) : BOOL
13411: Func wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
13412: Func wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
13413: Func wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
13414: Func wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
13415: Func wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
13416: Func wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
13417: Func wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Int;Flags:UINT): THandle
13418: Func wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
13419: Func wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
13420: //Func wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
13421: Func wLoadString(hInstance:HINST;uID:UINT;lpBuffer:PKOLChar;nBufferMax:Int):Int
13422: Func wMapVirtualKey( uCode, uMapType : UINT) : UINT
13423: Func wMapVirtualKeyEx( uCode, uMapType : UINT; dwHkl : HKL) : UINT
13424: Func wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Int
13425: Func wMessageBoxEx(hWnd:HWND;lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word)Int
13426: //Func wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
13427: Func wModifyMenu(hMnu: HMENU; uPosition,uFlags,uIDNewItem:UINT; lpNewItem:PKOLChar): BOOL
13428: //Func wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
13429: //Func wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
13430: //Func wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
13431: Func wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
13432: Func wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD) : HDESK
13433: Func wOpenWindowStation(lpszWinSta:PKOLChar;fInherit:BOOL;dwDesiredAccess:DWORD):HWINSTA
13434: Func wPeekMessage(var lpMsg:TMsg;hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
13435: Func wPostMessage(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):BOOL
13436: Func wPostThreadMessage(idThread:DWORD;Msg:UINT;wParam:WPARAM;lParam:LPARAM): BOOL

```

```

13437: Func wRealGetWindowClass(hwnd:HWND;pszType PKOLChar;cchType:UINT):UINT
13438: // Func wRegisterClass( const lpWndClass : TWndClass ) : ATOM
13439: // Func wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
13440: Func wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
13441: //Func wRegisterDeviceNotification(hRecipient:THandle;NotificationFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13442: Func wRegisterWindowMessage( lpString : PKOLChar ) : UINT
13443: Func wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
13444: Func wSendDlgItemMessage( hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13445: Func wSendMessage( hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM): LRESULT
13446: //Func wSendMessageCallback(hWnd: HWND; Msg:UINT;wParam:
WPARAM;lParam:LPARAM;lpResultCallback:TFNSendAsyncProc;dwData DWORD):BOOL
13447: Func wSendMessageTimeout( hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM;fuFlags,uTimeout:UINT;var
lpdwResult:DWORD):LRESULT
13448: Func wSendNotifyMessage( hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):BOOL
13449: Func wSetClassLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint ) : DWORD
13450: Func wSetDlgItemText( hDlg: HWND;nIDDlgItem:Int;lpString : PKOLChar ) : BOOL
13451: //Func wSetMenuItemInfo(p1:HMENU;p2:UINT;p3:BOOL;const p4:TMenuItemInfo):BOOL
13452: Func wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle ) : BOOL
13453: // Func wSetUserObjectInformation(hObj:THandle;nIndex:Int;pvInfo:Pointer;nLength:DWORD):BOOL
13454: Func wSetWindowLong( hWnd : HWND; nIndex : Int; dwNewLong : Longint ) : Longint
13455: Func wSetWindowText( hWnd : HWND; lpString : PKOLChar ) : BOOL
13456: //Func wSetWindowsHook(nFilterType:Int; pfnFilterProc:TFNHookProc): HHOOK
13457: //Func wSetWindowsHookEx(idHook:Int;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13458: // Func wSystemParametersInfo(uiAction,uiParam:UINT; pvParam:Pointer; fWinIni: UINT):BOOL
13459: Func wTabbedTextOut( hDC:HDC;X,Y: Int;lpString:PKOLChar;nCount,nTabPositions:Int;var lpnTabStopPositions,
nTabOrigin:Int):Longint;
13460: Func wTranslateAccelerator( hWnd:HWND;hAccTable:HACCEL;var lpMsg:TMsg): Int
13461: Func wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
13462: Func wVkKeyScan( ch : KOLChar ) : SHORT
13463: Func wVkKeyScanEx( ch : KOLChar; dwhkl : HKL ) : SHORT
13464: Func wWinHelp( hWndMain:HWND;lpszHelp:PKOLChar;uCommand:UINT;dwData:DWORD):BOOL
13465: Func wwsprintf( Output : PKOLChar; Format : PKOLChar ) : Int
13466: Func wwvsprintf( Output : PKOLChar; Format: PKOLChar; arglist:va_list): Int
13467:
13468: //TestDrive!
13469: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13470: 'PROC_CONVERTSIDSTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSida
13471: Func GetDomainUserSidS(const domainName:Str;const userName:Str;var foundDomain:Str):str;
13472: Func GetLocalUserSidStr( const UserName :Str): Str
13473: Func getPid4user(const domain:str;const user:str;var pid:dword):boolean
13474: Func Impersonate2User( const domain :Str; const user :Str):Bool
13475: Func GetProcessUserByPid(pid:DWORD;var UserName,Domain:Ansistr):Bool
13476: Func KillProcessbyname( const exename :Str; var found : Int ) : Int
13477: Func getWinProcessList : TStringList
13478: Func WaitTilClose( hWnd: Int): Int;
13479: Func DoUserMsgs:Bool;
13480: Func MsgFunc( hWnd,Msg,wParam,lParam:Int):Int; stdcall;
13481: Proc ShowMsg(hParent:Int; const Mess, Title:Str); //modal but NOT blockable
13482: Proc DeleteMsgForm(Handle: Int);
13483: Proc DisableForms;
13484: Func FoundTopLevel( hWnd, LParam: Int): BOOL; StdCall;
13485: end;
13486:
13487: Proc SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13488: begin
13489: 'AfMaxSyncSlots','LongInt'( 64);
13490: 'AfSynchronizeTimeout','LongInt'( 2000);
13491: TAFSyncSlotID, 'DWORD
13492: TAFSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13493: TAFSafeSyncEvent', 'Proc ( ID : TAFSyncSlotID)
13494: TAFSafeDirectSyncEvent', 'Procedure
13495: Func AfNewSyncSlot( const AEvent : TAFSafeSyncEvent) : TAFSyncSlotID
13496: Func AfReleaseSyncSlot( const ID : TAFSyncSlotID) :Bool
13497: Func AfEnableSyncSlot( const ID : TAFSyncSlotID; Enable:Bool) :Bool
13498: Func AfValidateSyncSlot( const ID : TAFSyncSlotID) :Bool
13499: Func AfSyncEvent( const ID : TAFSyncSlotID; Timeout : DWORD) :Bool
13500: Func AfDirectSyncEvent(Event:TAFSafeDirectSyncEvent;Timeout:DWORD):Boolean
13501: Func AfIsSyncMethod :Bool
13502: Func AfSyncWnd : HWND
13503: Func AfSyncStatistics : TAFSyncStatistics
13504: Proc AfClearSyncStatistics
13505: end;
13506:
13507: Proc SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13508: begin
13509: 'fBinary','LongWord'($00000001);
13510: 'fParity','LongWord'($00000002);
13511: 'fOutxCtsFlow','LongWord').SetUInt($00000004);
13512: 'fOutxDsrFlow','LongWord'($00000008);
13513: 'fDtrControl','LongWord'($00000030);
13514: 'fDtrControlDisable','LongWord'($00000000);
13515: 'fDtrControlEnable','LongWord'($00000010);
13516: 'fDtrControlHandshake','LongWord'($00000020);
13517: 'fDsrSensitivity','LongWord'($00000040);
13518: 'fTXContinueOnXoff','LongWord'($00000080);
13519: 'fOutX','LongWord'($00000100);
13520: 'fInX','LongWord'($00000200);
13521: 'fErrorChar','LongWord'($00000400);
13522: 'fNull','LongWord'($00000800);

```

```

13523: 'fRtsControl','LongWord')( $00003000);
13524: 'fRtsControlDisable','LongWord')( $00000000);
13525: 'fRtsControlEnable','LongWord')( $00001000);
13526: 'fRtsControlHandshake','LongWord')( $00002000);
13527: 'fRtsControlToggle','LongWord')( $00003000);
13528: 'fAbortOnError','LongWord')( $00004000);
13529: 'fDummy2','LongWord')( $FFFF8000);
13530: TAFCoreEvent, '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13531: FindClass('TOBJECT'),'EAFComPortCoreError
13532: FindClass('TOBJECT'),'TAFComPortCore
13533: TAFComPortCoreEvent', 'Proc ( Sender : TAFComPortCore; Even'
13534: + 'tKind : TAFCoreEvent; Data : DWORD)
13535: SIRegister_TAFComPortCoreThread(CL);
13536: SIRegister_TAFComPortEventThread(CL);
13537: SIRegister_TAFComPortWriteThread(CL);
13538: SIRegister_TAFComPortCore(CL);
13539: Func FormatDeviceName( PortNumber : Int ) :Str
13540: end;
13541:
13542: Proc SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13543: begin
13544:   TAFIOFileStreamEvent', 'Func ( const fileName :Str; mode: Word) : TStream
13545:   TAFIOFileStreamExistsEvent', 'Func ( const fileName :Str) :Bool
13546:   SIRegister_ApplicationFileIO(CL);
13547:   TDataFileCapability', '( dfcRead, dfcWrite )
13548:   TDataFileCapabilities', 'set of TDataFileCapability
13549:   SIRegister_TDataFile(CL);
13550:   //TDataFileClass', 'class of TDataFile
13551:   Func ApplicationFileIODefined :Bool
13552:   Func CreateFileStream(const fileName:str;mode:WordfmShareDenyNone):TStream
13553:   Func FileStreamExists(const fileName:Str):Bool
13554:   //Proc Register
13555: end;
13556:
13557: Proc SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13558: begin
13559:   TALFBXFieldType', '( uftUnknown, uftNumeric, uftChar, uftVarchar'
13560:   +', uftCString, uftSmallint, uftInt, uftQuad, uftFloat, uftDoublePrecisi'
13561:   + 'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull)
13562:   TALFBXScale', 'Int
13563:   FindClass('TOBJECT'),'EALFBXConvertError
13564:   SIRegister_EALFBXError(CL);
13565:   SIRegister_EALFBXException(CL);
13566:   FindClass('TOBJECT'),'EALFBXGfixError
13567:   FindClass('TOBJECT'),'EALFBXDSQLError
13568:   FindClass('TOBJECT'),'EALFBXDynError
13569:   FindClass('TOBJECT'),'EALFBXGBakError
13570:   FindClass('TOBJECT'),'EALFBXGSecError
13571:   FindClass('TOBJECT'),'EALFBXLicenseError
13572:   FindClass('TOBJECT'),'EALFBXGStatError
13573:   //EALFBXExceptionClass', 'class of EALFBXError
13574:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13575:   + '37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13576:   + 'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13577:   + 'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13578:   + 'csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13579:   + 'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13580:   + '4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13581:   + '8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13582:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13583:   + 'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13584:   + 'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13585:   + 'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout)
13586:   TALFBXTransParams', 'set of TALFBXTransParam
13587:   Func ALFBXStrToCharacterSet( const CharacterSet : Ansistr) : TALFBXCharacterSet
13588:   Func ALFBXCreateDBParams( Params : Ansistr; Delimiter : Char) : Ansistr
13589:   Func ALFBXCreateBlobParams( Params : Ansistr; Delimiter : Char) : Ansistr
13590:   'cALFBXMaxParamLength', 'LongInt( 125);
13591:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13592:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13593:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13594:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13595:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete, '
13596:   + 'stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13597:   + 't, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13598:   SIRegister_TALFBXSQLDA(CL);
13599:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13600:   SIRegister_TALFBXPoolStream(CL);
13601:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13602:   TALFBXBlobData', 'record Size : Int; Buffer :Str; end
13603:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13604:   //TALFBXArrayDesc', 'TISCArrDesc
13605:   //TALFBXBlobDesc', 'TISCBlobDesc
13606:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13607:   //TALFBXArrayInfo', 'record index : Int; size: Int; info: TALFBXArrayDesc; end
13608:   SIRegister_TALFBXSQLResult(CL);
13609:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13610:   SIRegister_TALFBXSQLParams(CL);
13611:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams

```

```

13612:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13613:   + 'atementType : TALFBXStatementType; end
13614:   FindClass('TOBJECT'), 'TALFBXLibrary
13615:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13616:   TALFBXOnConnectionLost', 'Proc ( Lib : TALFBXLibrary)
13617:   //TALFBXOnGetDBExceptionClass', 'Proc ( Number : Int; out'
13618:   //+ ' Excep : EALFBXExceptionClass)
13619:   SIRegister TALFBXLibrary(CL);
13620:   'cALFBXDateOffset', 'LongInt'( 15018);
13621:   'cALFBXTimeCoeff', 'LongInt'( 864000000);
13622:   //Proc ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13623:   //Proc ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13624:   //Func ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13625:   Proc ALFBXDecodesSQLDate( v : Int; out Year : SmallInt; out Month, Day : Word)
13626:   Proc ALFBXDecodesSQLTime(v:Card;out Hour,Minute,Second:Word;out Fractions: LongWord)
13627:   //Proc ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13628:   //Proc ALFBXEncodeTimeStamp1( const Date : Int; v : PISCTimeStamp);
13629:   //Proc ALFBXEncodeTimeStamp2( const Time :Card; v : PISCTimeStamp);
13630:   Func ALFBXEncodesSQLDate(Year : Int; Month, Day : Int) : Int
13631:   Func ALFBXEncodesSQLTime(Hour,Minute,Second:Word;var Fractions:LongWord):Card
13632:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13633:   TALFBXDPBInfo', 'record Name : Ansistr; ParamType : TALFBXParamType; end
13634:   Func ALFBXSQLQuote( const name : Ansistr) : Ansistr
13635:   Func ALFBXSQLUnQuote( const name : Ansistr) : Ansistr
13636: end;
13637:
13638: Proc SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13639: begin
13640:   TALFBXClientSQLParam', 'record Value : Ansistr; IsNull :Bool; end
13641:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13642:   TALFBXClientSelectDataSQL', 'record SQL : Ansistr; Params : T'
13643:   +ALFBXClientSQLParams; RowTag : Ansistr; ViewTag : Ansistr; Skip : in'
13644:   +teger; First : Int; CacheThreshold : Int; end
13645:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13646:   TALFBXClientUpdateDataSQL', 'record SQL : Ansistr; Params: TALFBXClientSQLParams; end
13647:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13648:   TALFBXClientMonitoringIOStats', 'record
page_reads:int64;page_writes:int64;page_fetches:int64;page_marks:int64;end
13649:   SIRegister TALFBXClient(CL);
13650:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13651:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13652:   SIRegister_TALFBXConnectionWithStmntPoolContainer(CL);
13653:   SIRegister_TALFBXConnectionWithoutStmntPoolContainer(CL);
13654:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13655:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13656:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13657:   SIRegister_TALFBXConnectionPoolClient(CL);
13658:   SIRegister_TALFBXEventThread(CL);
13659:   Func AlMySqlClientSlashedStr( const Str : Ansistr) : Ansistr
13660: end;
13661:
13662: Proc SIRegister_ovcBidi(CL: TPSPascalCompiler);
13663: begin
13664:   _OSVERSIONINFOA = record
13665:     dwOSVersionInfoSize: DWORD;
13666:     dwMajorVersion: DWORD;
13667:     dwMinorVersion: DWORD;
13668:     dwBuildNumber: DWORD;
13669:     dwPlatformId: DWORD;
13670:     szCSDVersion: array[0..127] of AnsiChar;{ Maintenance Ansistr for PSS usage}
13671: end;
13672:   TOSVersionInfoA', ' _OSVERSIONINFOA
13673:   TOSVersionInfo', 'TOSVersionInfoA
13674:   'WS_EX_RIGHT', 'LongWord')( $00001000);
13675:   'WS_EX_LEFT', 'LongWord')( $00000000);
13676:   'WS_EX_RTLREADING', 'LongWord')( $00002000);
13677:   'WS_EX_LTRREADING', 'LongWord')( $00000000);
13678:   'WS_EX_LEFTSCROLLBAR', 'LongWord')( $00004000);
13679:   'WS_EX_RIGHTSCROLLBAR', 'LongWord')( $00000000);
13680:   Func SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13681:   'LAYOUT_RTL', 'LongWord')( $00000001);
13682:   'LAYOUT_BTT', 'LongWord')( $00000002);
13683:   'LAYOUT_VBH', 'LongWord')( $00000004);
13684:   'LAYOUT_BITMAPORIENTATIONPRESERVED', 'LongWord')( $00000008);
13685:   'NOMIRRORBITMAP', 'LongWord')( DWORD ( $80000000 ));
13686:   Func SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13687:   Func GetLayout( dc : hdc) : DWORD
13688:   Func IsBidi : Bool
13689:   Func GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13690:   Func GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13691:   Func SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13692:   Func GetPriorityClass( hProcess : THandle) : DWORD
13693:   Func OpenClipboard( hWndNewOwner : HWND) : BOOL
13694:   Func CloseClipboard : BOOL
13695:   Func GetClipboardSequenceNumber : DWORD
13696:   Func GetClipboardOwner : HWND
13697:   Func SetClipboardViewer( hWndNewViewer : HWND) : HWND
13698:   Func GetClipboardViewer : HWND
13699:   Func ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL

```

```

13700: Func SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13701: Func GetClipboardData( uFormat : UINT) : THandle
13702: Func RegisterClipboardFormat( lpszFormat : PChar) : UINT
13703: Func CountClipboardFormats : Int
13704: Func EnumClipboardFormats( format : UINT) : UINT
13705: Func GetClipboardFormatName( format:UINT;lpszFormatName:PChar;cchMaxCount:Int):Int
13706: Func EmptyClipboard : BOOL
13707: Func IsClipboardFormatAvailable( format : UINT) : BOOL
13708: Func GetPriorityClipboardFormat( var paFormatPriorityList, cFormats: Int): Int
13709: Func GetOpenClipboardWindow : HWND
13710: Func EndDialog( hDlg : HWND; nResult : Int) : BOOL
13711: Func GetDlgItem( hDlg : HWND; nIDDlgItem : Int) : HWND
13712: Func SetDlgItemInt( hDlg:HWND;nIDDlgItem:Int;uValue:UINT;bSigned:BOOL): BOOL
13713: Func GetDlgItemInt( hDlg:HWND;nIDDlgItem:Int;var lpTranslated:BOOL;bSigned: BOOL):UINT
13714: Func SetDlgItemText( hDlg : HWND; nIDDlgItem : Int; lpString : PChar) : BOOL
13715: Func CheckDlgButton( hDlg : HWND; nIDButton : Int; uCheck : UINT) : BOOL
13716: Func CheckRadioButton( hDlg:HWND;nIDFirstButton,nIDLastButton,nIDCheckButton:Int):BOOL
13717: Func IsDlgButtonChecked( hDlg : HWND; nIDButton : Int) : UINT
13718: Func SendDlgItemMessage( hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13719: end;
13720:
13721: Proc SIRegister_DXPUtills(CL: TPSPascalCompiler);
13722: begin
13723:   Func glExecuteAndWait( cmdLine:Str; visibility:Word; timeout:Card; killAppOnTimeOut:Bool): Int;
13724:   Func GetTemporaryFilePath :Str
13725:   Func GetTemporaryFileName :Str
13726:   Func FindFileInPaths( const fileName, paths :Str) :Str
13727:   Func PathsToString( const paths : TStrings) :Str
13728:   Proc StringToPaths( const pathsString :Str; paths : TStrings)
13729:     //Func MacroExpandPath( const aPath :Str) :Str
13730:   end;
13731:
13732: Proc SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13733: begin
13734:   SIRegister_TALMultiPartBaseContent(CL);
13735:   SIRegister_TALMultiPartBaseContents(CL);
13736:   SIRegister_TALMultiPartBaseStream(CL);
13737:   SIRegister_TALMultiPartBaseEncoder(CL);
13738:   SIRegister_TALMultiPartBaseDecoder(CL);
13739:   Func ALMultiPartExtractBoundaryFromContentType( aContentType:Ansistr): Ansistr
13740:   Func ALMultiPartExtractSubValueFromHeaderLine( aHeaderLine:Ansistr; aName:Ansistr):Ansistr;
13741:   Func ALMultiPartSetSubValueInHeaderLine( aHeaderLine:Ansistr; aName,AValue:Ansistr):Ansistr;
13742: end;
13743:
13744: Proc SIRegister_SmallUtils(CL: TPSPascalCompiler);
13745: begin
13746:   TdriveSize', 'record FreeS : Int64; Totals : Int64; end
13747:   TWinVerRec', 'record WinPlatform : Int; WinMajorVersion : In
13748:   +teger; WinMinorVersion :Int; WinBuildNumber : Int; WinCSDVersion:Str; end
13749:   Func aAllocPadedMem( Size :Card) : TObject
13750:   Proc aFreePadedMem( var P : TObject);
13751:   Proc aFreePadedMem1( var P : PChar);
13752:   Func aCheckPadedMem( P : Pointer) : Byte
13753:   Func aGetPadMemSize( P : Pointer) :Card
13754:   Func aAllocMem( Size :Card) : Pointer
13755:   Func aStrLen( const Str : PChar) :Card
13756:   Func aStrLCopy( Dest : PChar; const Source : PChar; MaxLen :Card) : PChar
13757:   Func aStrECopy( Dest : PChar; const Source : PChar) : PChar
13758:   Func aStrCopy( Dest : PChar; const Source : PChar) : PChar
13759:   Func aStrEnd( const Str : PChar) : PChar
13760:   Func aStrScan( const Str : PChar; aChr : Char) : PChar
13761:   Func aStrMove( Dest : PChar; const Source : PChar; Count :Card) : PChar
13762:   Func aPCharLength( const Str : PChar) :Card
13763:   Func aPCharUpper( Str : PChar) : PChar
13764:   Func aPCharLower( Str : PChar) : PChar
13765:   Func aStrCat( Dest : PChar; const Source : PChar) : PChar
13766:   Func aLastDelimiter( const Delimiters, S :Str) : Int
13767:   Func aCopyTail( const S :Str; Len : Int) :Str
13768:   Func aInt2Thos( I : Int64) :Str
13769:   Func aUpperCase( const S :Str) :Str
13770:   Func aLowerCase( const S :Str) :Str
13771:   Func aCompareText( const S1, S2 :Str) : Int
13772:   Func aSameText( const S1, S2 :Str) :Bool
13773:   Func aInt2Str( Value : Int64) :Str
13774:   Func aStr2Int( const Value :Str) : Int64
13775:   Func aStr2IntDef( const S :Str; Default : Int64) : Int64
13776:   Func aGetFileExt( const FileName :Str) :Str
13777:   Func aGetFilePath( const FileName :Str) :Str
13778:   Func aGetFileName( const FileName :Str) :Str
13779:   Func aChangeExt( const FileName, Extension :Str) :Str
13780:   Func aAdjustLineBreaks( const S :Str) :Str
13781:   Func aGetWindowStr( WinHandle : HWND) :Str
13782:   Func aDiskSpace( Drive :Str) : TdriveSize
13783:   Func aFileExists( FileName :Str) :Bool
13784:   Func aFileSize( FileName :Str) : Int64
13785:   Func aDirectoryExists( const Name :Str) :Bool
13786:   Func aSysErrorMessage( ErrorCode : Int) :Str
13787:   Func aShortPathName( const LongName :Str) :Str
13788:   Func aGetWindowVer : TWinVerRec

```



```

13789: Proc InitDriveSpacePtr;
13790: end;
13791:
13792: Proc SIRegister_MakeApp(CL: TPSPascalCompiler);
13793: begin
13794:   aZero,'LongInt'( 0);
13795:   'makeappDEF','LongInt'(- 1);
13796:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13797:   'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13798:   'CS_KEYCVTWINDOW','LongInt'( 4);
13799:   'CS_DBLCLKS','LongInt'( 8);
13800:   'CS_OWNDC','LongWord'($20);
13801:   'CS_CLASSDC','LongWord'($40);
13802:   'CS_PARENTDC','LongWord'($80);
13803:   'CS_NOKEYCVT','LongWord'($100);
13804:   'CS_NOCLOSE','LongWord'($200);
13805:   'CS_SAVEBITS','LongWord'($800);
13806:   'CS_BYTEALIGNCLIENT','LongWord'($1000);
13807:   'CS_BYTEALIGNWINDOW','LongWord'($2000);
13808:   'CS_GLOBALCLASS','LongWord'($4000);
13809:   'CS_IME','LongWord'($10000);
13810:   'CS_DROPSHADOW','LongWord'($20000);
13811:   //PPanelFunc','^TPanelFunc // will not work
13812:   TPanelStyle','(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13813:   TFontLook','( flBold, flItalic, flUnderLine, flStrikeOut )
13814:   TFontLooks','set of TFontLook
13815:   TMessagefunc','function(hWnd,iMsg,wParam,lParam:Int):Int)
13816:   Func SetWinClass(const ClassName:str; pMessFunc: Tmessagefunc; wcStyle : Int): Word
13817:   Func SetWinClassO( const ClassNam:str; pMessFunc: TObject; wcStyle : Int): Word
13818:   Func SetWinClass(const ClassName:str; pMessFunc: TObject; wcStyle : Int): Word
13819:   Func MakeForm(Left,Top,Width,Height:Int;const Caption:str;WinStyle:Int):Int
13820:   Proc RunMsgLoop( Show :Bool)
13821:   Func MakeFont(Height,Width:Int;const FontName:str; Look:TFontLooks;Roman:Boolean):Int
13822:   Func MakeButton(Left,Top,Width,Height:Int;pCaption:PChar;hParent,ID_Number:Card;hFont:Int):Int;
13823:   Func MakeListBox(Left,Top,Width,Height,Parent:Int;const ListItems:str;WinStyle:Int):Int
13824:   Func MakeComboBox(Left,Top,Width,Height,Parent:Int;const ListItems:str;WinStyle:Int):Int
13825:   Func MakePanel(Left,Top,Width,Height,hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13826:   Func MakeSubMenu(const ItemList :Str; ID1, ID2 :Card; hMenu : Int) : Int
13827:   Func id4menu( a, b : Byte; c : Byte; d : Byte) :Card
13828:   Proc DoInitMakeApp //set first to init formclasscontrol!
13829:   end;
13830:
13831: Proc SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13832: begin
13833:   TScreenSaverOption','( ssoAutoAdjustFormProperties, ssoAutoHook'
13834:   +KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13835:   TScreenSaverOptions','set of TScreenSaverOption
13836:   'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13837:   TScreenSaverPreviewEvent','Proc ( Sender : TObject; previewHwnd: HWND)
13838:   SIRegister_TScreenSaver(CL);
13839:   //Proc Register
13840:   Proc SetScreenSaverPassword
13841:   end;
13842:
13843: Proc SIRegister_XCollection(CL: TPSPascalCompiler);
13844: begin
13845:   FindClass('TOBJECT'),'TXCollection
13846:   SIRegister_EFilerException(CL);
13847:   SIRegister_TXCollectionItem(CL);
13848:   //TXCollectionItemClass','class of TXCollectionItem
13849:   SIRegister_TXCollection(CL);
13850:   Proc RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13851:   Proc DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13852:   Proc RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13853:   Proc UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13854:   Func FindXCollectionItemClass( const className :Str) : TXCollectionItemClass
13855:   Func GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13856:   end;
13857:
13858: Proc SIRegister_XOpenGL(CL: TPSPascalCompiler);
13859: begin
13860:   TMapTexCoordMode','(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13861:   Proc xglMapTexCoordToNull
13862:   Proc xglMapTexCoordToMain
13863:   Proc xglMapTexCoordToSecond
13864:   Proc xglMapTexCoordToDual
13865:   Proc xglMapTexCoordToArbitrary( const units : array of Cardinal);
13866:   Proc xglMapTexCoordToArbitrary1( const bitWiseUnits :Card);
13867:   Proc xglMapTexCoordToArbitraryAdd( const bitWiseUnits :Card)
13868:   Proc xglBeginUpdate
13869:   Proc xglEndUpdate
13870:   Proc xglPushState
13871:   Proc xglPopState
13872:   Proc xglForbidSecondTextureUnit
13873:   Proc xglAllowSecondTextureUnit
13874:   Func xglGetBitWiseMapping :Card
13875:   end;
13876:

```

```

13877: Proc SIRegister_VectorLists(CL: TPSPascalCompiler);
13878: begin
13879:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13880:   TBaseListOptions', 'set of TBaseListOption
13881:   SIRegister_TBaseList(CL);
13882:   SIRegister_TBaseVectorList(CL);
13883:   SIRegister_TAffineVectorList(CL);
13884:   SIRegister_TVectorList(CL);
13885:   SIRegister_TTexPointList(CL);
13886:   SIRegister_TXIntList(CL);
13887:   //PSingleArrayList', '^TSingleArrayList // will not work
13888:   SIRegister_TSingleList(CL);
13889:   SIRegister_TByteList(CL);
13890:   SIRegister_TQuaternionList(CL);
13891:   Proc QuickSortLists(startIndex,endIndex:Int;refList:TSingleList; objList : TList);
13892:   Proc QuickSortLists1(startIndex,endIndex:Int;refList: TSingleList; objList: TBaseList);
13893:   Proc FastQuickSortLists(startIndex,endIndex:Int;refList:TSingleList;objList:TPersistentObjectList);
13894: end;
13895:
13896: ***** File C:\maXbox\maxbox3\maxbox3\source\REST\upsi_UIntList.pas
13897: File C:\maXbox\maxbox3\maxbox3\source\JvJCLUtils.pas
13898: { compile-time registration functions }
13899: 24: Proc SIRegister_TIntList(CL: TPSPascalCompiler);
13900: Proc SIRegister_UIntList(CL: TPSPascalCompiler);
13901: { run-time registration functions }
13902: 28: Proc RIRegister_TIntList(CL: TPSRuntimeClassImporter);
13903: Proc RIRegister_UIntList(CL: TPSRuntimeClassImporter);
13904: (*-----*)
13905: 49: Proc SIRegister_TIntList(CL: TPSPascalCompiler);
13906: 51: //with RegClassS(CL,'TPersistent','TIntList') do
13907:   with ClassN(CL.FindClass('TPersistent'),'TIntList') do begin
13908:     with ClassN(CL.FindClass('TList'),'TIntList') do begin
13909:       //with RegClassS(CL,'TPersistent','TIntList') do
13910:
13911: TIntList = class(TPersistent)
13912: private
13913:   FUpDateCount: Int;
13914:   FList: PIntItemList;
13915:   FCount: Int;
13916:   FCapacity: Int;
13917:   FSorted:Bool;
13918:   FDuplicates: TDuplicates;
13919:   FOnChange: TNotifyEvent;
13920:   FOnChanging: TNotifyEvent;
13921:   Proc ExchangeItems(Index1, Index2: Int);
13922:   Proc Grow;
13923:   Proc QuickSort(L, R: Int; SCompare: TIntListSortCompare);
13924:   Proc InsertItem(Index: Int; const S: int64);
13925:   Proc SetSorted(Value:Bool);
13926: protected
13927:   Proc Error(const Msg:Str; Data: Int);
13928:   Proc Changed; virtual;
13929:   Proc Changing; virtual;
13930:   Func Get(Index: Int): int64;
13931:   Func GetCapacity: Int;
13932:   //Func GetCount: Int;
13933:   Func GetObject(Index: Int): TObject;
13934:   Proc Put(Index: Int; const S: int64);
13935:   Proc PutObject(Index: Int; AObject: TObject);
13936:   Proc SetCapacity(NewCapacity: Int);
13937:   Proc SetUpdateState(Updating:Bool);
13938: public
13939:   Func GetCount: Int;
13940:   destructor Destroy; override;
13941:   Func Add(const S: int64): Int;
13942:   Func AddObject(const S: int64; AObject: TObject): Int; virtual;
13943:   Proc Clear;
13944:   Proc Delete(Index: Int);
13945:   Proc Exchange(Index1, Index2: Int);
13946:   Func Find(const S: int64; var Index: Int):Bool; virtual;
13947:   Func IndexOf(const S: int: Int;
13948:   Proc Insert(Index: Int; const S: int64);
13949:   Proc Sort; virtual;
13950:   Proc CustomSort(Compare: TIntListSortCompare); virtual;
13951:   Proc LoadFromFile(const FileName:Str); virtual;
13952:   Proc LoadFromStream(Stream:TStream); virtual;
13953:   Proc SaveToFile(const FileName:Str); virtual;
13954:   Proc SaveToStream(Stream: TStream);
13955:   property Duplicates: TDuplicates read FDuplicates write FDuplicates;
13956:   property Sorted:Bool read FSorted write SetSorted;
13957:   property OnChange: TNotifyEvent read FOnChange write FOnChange;
13958:   property OnChanging: TNotifyEvent read FOnChanging write FOnChanging;
13959:   property Ints [Index: Int]: int64 read Get write Put; default;
13960:   property Count: Int read GetCount;
13961:   property Objects[Index: Int]: TObject read GetObject write PutObject;
13962: end;
13963:
13964: procedure SIRegister_TInteger(CL: TPSPascalCompiler); //biginteger bigint
13965: begin

```

```

13966: //with RegClassS(CL,'TObject', 'TInteger') do
13967: with CL.AddClassN(CL.FindClass('TObject'),'TInteger') do begin
13968:   Constructor Create1( const initialValue : int64)
13969:   Constructor Create
13970:   Procedure Free
13971:   RegisterProperty('Digits', 'TDigits', iptr);
13972:   Procedure Assign( const I2 : TInteger);
13973:   Procedure Assign1( const I2 : int64);
13974:   Procedure Assign2( const I2 : string);
13975:   Procedure AbsoluteValue;
13976:   Procedure Add( const I2 : TInteger);
13977:   Procedure Add1( const I2 : int64);
13978:   Procedure AssignZero;
13979:   Procedure AssignOne;
13980:   Procedure Subtract( const I2 : TInteger);
13981:   Procedure Subtract1( const I2 : int64);
13982:   Procedure Mult( const I2 : TInteger);
13983:   Procedure Mult1( const I2 : int64);
13984:   Procedure FastMult( const I2 : TInteger);
13985:   Procedure Divide( const I2 : TInteger);
13986:   Procedure Divide1( const I2 : int64);
13987:   Procedure Modulo( const I2 : TInteger);
13988:   Procedure Modulo1( const N : int64);
13989:   Procedure ModPow( const I2, m : TInteger);
13990:   Procedure InvMod( I2 : TInteger);
13991:   Procedure DivideRem( const I2 : TInteger; var remain : TInteger);
13992:   Procedure DivideRemTrunc( const I2 : TInteger; var remain : TInteger);
13993:   Procedure DivideRemFloor( const I2 : TInteger; var remain : TInteger);
13994:   Procedure DivideRemEuclidean( const I2 : TInteger; var remain : TInteger);
13995:   Function Compare( I2 : TInteger) : integer;
13996:   Function Compare1( I2 : int64) : integer;
13997:   Procedure Factorial;
13998:   Function ConvertToDecimalString( commas : boolean) : string;
13999:   Function ConvertToInt64( var N : int64) : boolean;
14000:   Function DigitCount : integer;
14001:   Procedure SetSign( s : integinteger);
14002:   Function GetSign : integer;
14003:   Function IsOdd : boolean;
14004:   Function IsPositive : boolean;
14005:   Function IsNegative : boolean;
14006:   Function IsProbablyPrime : boolean;
14007:   Function IsZero : boolean;
14008:   Procedure ChangeSign;
14009:   Procedure Pow( const exponent : int64);
14010:   Procedure Sgroot;
14011:   Procedure Square;
14012:   Procedure FastSquare;
14013:   Procedure Gcd( const I2 : TInteger);
14014:   Procedure Gcd1( const I2 : int64);
14015:   Procedure NRoot( const Root : int64);
14016:   Function GetBase : integer;
14017:   Function BitCount : integer;
14018:   Function ConvertToHexString : string;
14019:   Function AssignRandomPrime( BitLength:integer; seed:String; mustMatchBitLength:boolean):boolean;
14020:   Function AssignHex( HexStr : string) : boolean;
14021:   Procedure RandomOfSize( size : integer);
14022:   Procedure Random( maxint : TInteger);
14023:   Procedure Getnextprime;
14024: end;
14025: end;
14026:
14027: procedure SIRegister_TBigFloat(CL: TPSPascalCompiler); //bigdecimal bigdec
14028: begin
14029:   //with RegClassS(CL,'TObject', 'TBigFloat') do
14030:   with CL.AddClassN(CL.FindClass('TObject'),'TBigFloat') do begin
14031:     decpart, 'TFloatInt', iptrw);
14032:     sigdigits, 'word', iptrw);
14033:     exponent, 'integer', iptrw);
14034:     Constructor Create;
14035:     Constructor Create1( const MaxSig : TMaxSig);
14036:     Procedure Assign( A : TBigFloat);
14037:     Procedure Assign3( A : TBigFloat; SigDig : word);
14038:     Procedure Assign4( A : TInteger);
14039:     Procedure Assign5( N : int64);
14040:     Procedure Assign6( N : int64; SigDig : integer);
14041:     Procedure Assign7( d : extended);
14042:     Procedure Assign8( S : string);
14043:     Procedure Assign9( S : string; SigDig : word);
14044:     Procedure AssignZero;
14045:     Procedure AssignOne;
14046:     Procedure Add( B : TBigFloat);
14047:     Procedure Add1( B : int64);
14048:     Procedure AbsAdd( B : TBigFloat);
14049:     Procedure Subtract( B : TBigFloat);
14050:     Procedure Subtract1( B : int64);
14051:     Procedure Mult( B : TBigFloat);
14052:     Procedure Mult1( B : TBigFloat; const MaxSig : TMaxSig);
14053:     Procedure Mult2( B : TInteger);
14054:     Procedure Mult3( B : int64);

```

```

14055:   Procedure MultRaw( B : TBigFloat);
14056:   Procedure Reciprocal( const MaxSig : TMaxSig);
14057:   Procedure Divide( B : TBigFloat; const MaxSig : TMaxSig);
14058:   Procedure Divide1( B : TInteger; const MaxSig : TMaxSig);
14059:   Procedure Divide2( B : int64; const MaxSig : TMaxSig);
14060:   Procedure Square( const MaxSig : TMaxSig);
14061:   Function Compare( B : TBigFloat) : integer';
14062:   Function IsZero : boolean';
14063:   Procedure MaxBigFloat( B : TBigFloat);
14064:   Procedure MinBigFloat( B : TBigFloat);
14065:   Procedure Sqrt;
14066:   Procedure Sqrt1( const MaxSig : TMaxSig);
14067:   Procedure NRoot( N : integer; const MaxSig : TMaxSig);
14068:   Procedure IntPower( intpower : integer; const MaxSig : TMaxSig);
14069:   Procedure Power( power : TBigfloat; const MaxSig : TMaxSig);
14070:   Procedure Log( const MaxSig : TMaxSig);
14071:   Procedure Log10( const MaxSig : TMaxSig);
14072:   Procedure Exp( const MaxSig : TMaxSig);
14073:   Procedure PiConst( const MaxSig : TMaxSig);
14074:   Procedure Log2Const( const MaxSig : TMaxSig);
14075:   Procedure RoundToPrec( const MaxSig : TMaxSig);
14076:   Procedure RoundToPrecl;
14077:   Procedure Trunc( const x : integer);
14078:   Procedure Floor( const x : integer);
14079:   Procedure Ceiling( const x : integer);
14080:   Procedure Round( const x : integer);
14081:   Procedure AbsoluteValue';
14082:   Procedure Negate';
14083:   Procedure SetSigDigits( const newsigdigits : integer);
14084:   Function ConvertToString( const View : TView) : string';
14085:   Function ToString( const View : TView) : string';
14086:   Function ConvertToExtended( var num : extended) : boolean';
14087:   Function ConvertToInt64( var N : int64) : boolean';
14088:   Function IntPart : int64';
14089: end;
14090: end;
14091:
14092: Proc SIRegister_MeshUtils(CL: TPSPascalCompiler);
14093: begin
14094:   Proc ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
14095:   Proc ConvertStripToList1( const strip : TIntList; list : TIntList);
14096:   Proc ConvertStripToList2(const strip:TAffineVectorList;const indices:TIntList;list:TAffineVectorList);
14097:   Proc ConvertIndexedListToList(const data:TAffineVectlist;const indices:TIntList;list:TAffineVectorList);
14098:   Func BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList;const
normals:TAffineVectorList;const texCoords:TAffineVectorList):TIntList
14099:   Proc RemapReferences( reference : TAffineVectorList; const indices : TIntList);
14100:   Proc RemapReferences1( reference : TIntList; const indices : TIntList);
14101:   Proc RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntList)
14102:   Func RemapIndicesToIndicesMap( remapIndices : TIntList) : TIntList
14103:   Proc RemapTrianglesIndices( indices, indicesMap : TIntList)
14104:   Proc RemapIndices( indices, indicesMap : TIntList)
14105:   Proc UnifyTrianglesWinding( indices : TIntList)
14106:   Proc InvertTrianglesWinding( indices : TIntList)
14107:   Func BuildNormals( reference : TAffineVectorList; indices : TIntList) : TAffineVectorList
14108:   Func
BuildNonOrientedEdgesList(triangleIndices:TIntList;triangleEdges:TIntList;edgesTriangles:TIntList):TIntList
14109:   Proc WeldVertices(vertices:TAffineVectorList;indicesMap:TIntList;weldRadius: Single)
14110:   Func StripifyMesh(indices:TIntList;maxVertexIndex:Int;agglomerateLoneTriangles:Boolean):
TPersistentObjectList;
14111:   Proc IncreaseCoherency(indices : TIntList; cacheSize : Int)
14112:   Proc
SubdivideTriangles(smoothFactor:Single;vertices:TAffineVectorList;triangleIndices:TIntList;normals:TAffineVectorList;o
14113: end;
14114:
14115: Proc SIRegister_JclSysUtils(CL: TPSPascalCompiler);
14116: begin
14117:   Proc GetAndFillMem( var P : TObject; const Size : Int; const Value : Byte)
14118:   Proc FreeMemAndNil( var P : TObject)
14119:   Func PCharOrNil( const S :Str) : PChar
14120:   SIRegister_TJclReferenceMemoryStream(CL);
14121:   FindClass('TObject'),'EJclVMTError
14122: {Func GetVirtualMethodCount( AClass : TClass) : Int
14123: Func GetVirtualMethod( AClass : TClass; const Index : Int) : Pointer
14124: Proc SetVirtualMethod( AClass : TClass; const Index : Int; const Method:Pointer)
14125:   PDynamicIndexList', '^TDynamicIndexList // will not work
14126:   PDynamicAddressList', '^TDynamicAddressList // will not work
14127: Func GetDynamicMethodCount( AClass : TClass) : Int
14128: Func GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
14129: Func GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
14130: Func HasDynamicMethod( AClass : TClass; Index : Int) :Bool
14131: Func GetDynamicMethod( AClass : TClass; Index : Int) : Pointer
14132: Func GetInitTable( AClass : TClass) : PTypeInfo
14133:   PFieldEntry', '^TFieldEntry // will not work}
14134:   TFieldEntry', 'record OffSet : Int; IDX : Word; Name : ShortString; end
14135: Func JIsClass( Address : Pointer) :Bool
14136: Func JIsObject( Address : Pointer) :Bool
14137: Func GetImplementorOfInterface( const I : IInterface) : TObject
14138:   TDigitCount', 'Int
14139:   SIRegister_TJclNumericFormat(CL);

```

```

14140: Func JIntToStrZeroPad( Value, Count : Int) : Ansistr
14141: TTextHandler', 'Proc ( const Text :Str)
14142: // 'ABORT_EXIT_CODE', 'LongInt'( ERROR_CANCELLED 1223);
14143: Func JExecute(const CommandLine:Str;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
14144: Func JExecute1(const CommandLine:Str;var Output:Str;RawOutput:Bool;AbortPtr:PBool):Card;
14145: Func JExecute2(const CommandLine:Str;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
14146: Func JExecute3('cmd /C dir *.*',@TTextHandlerQ, true, false));
14147: Func JExecute3('cmd /C dir *.*',Nil, true, false));
14148: Func ReadKey : Char //to and from the DOS console !
14149: TModuleHandle', 'HINST
14150: //TModuleHandle', 'Pointer
14151: 'INVALID_MODULEHANDLE_VALUE', 'LongInt'( TModuleHandle ( 0 ));
14152: Func LoadModule( var Module : TModuleHandle; FileName :Str):Bool
14153: Func LoadModuleEx(var Module: TModuleHandle; FileName:Str;Flags:Card):Bool
14154: Proc UnloadModule( var Module : TModuleHandle)
14155: Func GetModuleSymbol( Module : TModuleHandle; SymbolName :Str) : Pointer
14156: Func GetModuleSymbolEx(Module:TModuleHandle;SymbolName:Str;var Accu:Boolean):Pointer
14157: Func ReadModuleData(Module:TModuleHandle;SymbolName:Str;var Buffer,Size:Card):Bool;
14158: Func WriteModuleData(Module:TModuleHandle;SymbolName:Str;var Buffer,Size:Card):Bool;
14159: FindClass('TOBJECT'),'EJclConversionError
14160: Func JStrToBoolean( const S :Str) :Bool
14161: Func JBooleanToStr( B :Bool) :Str
14162: Func JIntToBool( I : Int) :Bool
14163: Func JBoolToInt( B :Bool) : Int
14164: 'ListSeparator','String'; 'ListSeparator1','String':
14165: Proc ListAddItems( var List :Str; const Separator, Items:Str)
14166: Proc ListIncludeItems( var List :Str; const Separator, Items:Str)
14167: Proc ListRemoveItems( var List :Str; const Separator, Items:Str)
14168: Proc ListDelItem( var List :Str; const Separator :Str; const Index : Int)
14169: Func ListItemCount( const List, Separator :Str) : Int
14170: Func ListGetItem( const List, Separator :Str; const Index : Int) :Str
14171: Proc ListSetItem(var List:Str;const Separator:Str;const Index:Int;const Value:Str)
14172: Func ListItemIndex( const List, Separator, Item :Str) : Int
14173: Func SystemTObjectInstance : LongWord
14174: Func IsCompiledWithPackages :Bool
14175: Func JJclGUIDToString( const GUID : TGUID) :Str
14176: Func JJclStringToGUID( const S :Str) : TGUID
14177: SIRegister_TJclIntfCriticalSection(CL);
14178: SIRegister_TJclSimpleLog(CL);
14179: Proc InitSimpleLog( const ALogFileName :Str)
14180: end;
14181:
14182: Proc SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
14183: begin
14184: FindClass('TOBJECT'),'EJclBorRADException
14185: TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
14186: TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
14187: TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
14188: TJclBorRADToolPath', 'string
14189: 'SupportedDelphiVersions','LongInt'( 5 or 6 or 7 or 8 or 9 or 10 or 11);
14190: 'SupportedBCBVersions','LongInt'( 5 or 6 or 10 or 11);
14191: 'SupportedBDSVersions','LongInt'( 1 or 2 or 3 or 4 or 5);
14192: BorRADToolRepositoryPagesSection','String 'Repository Pages
14193: BorRADToolRepositoryDialogsPage','String 'Dialogs
14194: BorRADToolRepositoryFormsPage','String 'Forms
14195: BorRADToolRepositoryProjectsPage','String 'Projects
14196: BorRADToolRepositoryDataModulesPage','String 'Data Modules
14197: BorRADToolRepositoryObjectType','String 'Type
14198: BorRADToolRepositoryFormTemplate','String 'FormTemplate
14199: BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
14200: BorRADToolRepositoryObjectName','String 'Name
14201: BorRADToolRepositoryObjectPage','String 'Page
14202: BorRADToolRepositoryObjectIcon','String 'Icon
14203: BorRADToolRepositoryObjectDescr','String 'Description
14204: BorRADToolRepositoryObjectAuthor','String 'Author
14205: BorRADToolRepositoryObjectAncestor','String 'Ancestor
14206: BorRADToolRepositoryObjectDesigner','String 'Designer
14207: BorRADToolRepositoryDesignerDfm','String 'dfm
14208: BorRADToolRepositoryDesignerXfm','String 'xfm
14209: BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
14210: BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
14211: SourceExtensionDelphiPackage','String '.dpk
14212: SourceExtensionBCBPackage','String '.bpc
14213: SourceExtensionDelphiProject','String '.dpr
14214: SourceExtensionBCBProject','String '.bpr
14215: SourceExtensionBDSProject','String '.bdsproj
14216: SourceExtensionDProject','String '.dproj
14217: BinaryExtensionPackage','String '.bpl
14218: BinaryExtensionLibrary','String '.dll
14219: BinaryExtensionExecutable','String '.exe
14220: CompilerExtensionDCP','String '.dcp
14221: CompilerExtensionBPI','String '.bpi
14222: CompilerExtensionLIB','String '.lib
14223: CompilerExtensionTDS','String '.tds
14224: CompilerExtensionMAP','String '.map
14225: CompilerExtensionDRC','String '.drc
14226: CompilerExtensionDEF','String '.def
14227: SourceExtensionCPP','String '.cpp
14228: SourceExtensionH','String '.h

```



```

14229: SourceExtensionPAS', 'String' '.pas
14230: SourceExtensionDFM', 'String' '.dfm
14231: SourceExtensionXFM', 'String' '.xfm
14232: SourceDescriptionPAS', 'String' 'Pascal source file
14233: SourceDescriptionCPP', 'String' 'C++ source file
14234: DesignerVCL', 'String' 'VCL
14235: DesignerCLX', 'String' 'CLX
14236: ProjectTypePackage', 'String' 'package
14237: ProjectTypeLibrary', 'String' 'library
14238: ProjectTypeProgram', 'String' 'program
14239: Personality32Bit', 'String' '32 bit
14240: Personality64Bit', 'String' '64 bit
14241: PersonalityDelphi', 'String' 'Delphi
14242: PersonalityDelphiDotNet', 'String' 'Delphi.net
14243: PersonalityBCB', 'String' 'C++Builder
14244: PersonalityCSB', 'String' 'C#Builder
14245: PersonalityVB', 'String' 'Visual Basic
14246: PersonalityDesign', 'String' 'Design
14247: PersonalityUnknown', 'String' 'Unknown personality
14248: PersonalityBDS', 'String' 'Borland Developer Studio
14249: DOFDirectoriesSection', 'String' 'Directories
14250: DOFUnitOutputDirKey', 'String' 'UnitOutputDir
14251: DOFSearchPathName', 'String' 'SearchPath
14252: DOFConditionals', 'String' 'Conditionals
14253: DOFLinkerSection', 'String' 'Linker
14254: DOFPackagesKey', 'String' 'Packages
14255: DOFCompilerSection', 'String' 'Compiler
14256: DOFPackageNoLinkKey', 'String' 'PackageNoLink
14257: DOFAdditionalSection', 'String' 'Additional
14258: DOFOptionsKey', 'String' 'Options
14259: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
14260:   + 'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
14261:   + 'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
14262: TJclBorPersonalities', 'set of TJclBorPersonality
14263: TJclBorDesigner', '( bdVCL, bdCLX )
14264: TJclBorDesigners', 'set of TJclBorDesigner
14265: TJclBorPlatform', '( bp32bit, bp64bit )
14266: FindClass('TOBJECT'), 'TJclBorRADToolInstallation
14267: SIRegister_TJclBorRADToolInstallationObject(CL);
14268: SIRegister_TJclBorlandOpenHelp(CL);
14269: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
14270: TJclHelp2Objects', 'set of TJclHelp2Object
14271: SIRegister_TJclHelp2Manager(CL);
14272: SIRegister_TJclBorRADToolIdeTool(CL);
14273: SIRegister_TJclBorRADToolIdePackages(CL);
14274: SIRegister_IJclCommandLineTool(CL);
14275: FindClass('TOBJECT'), 'EJclCommandLineToolError
14276: SIRegister_TJclCommandLineTool(CL);
14277: SIRegister_TJclBorlandCommandLineTool(CL);
14278: SIRegister_TJclBCC32(CL);
14279: SIRegister_TJclDCC32(CL);
14280: TJclDCC', 'TJclDCC32
14281: SIRegister_TJclBpr2Mak(CL);
14282: SIRegister_TJclBorlandMake(CL);
14283: SIRegister_TJclBorRADToolPalette(CL);
14284: SIRegister_TJclBorRADToolRepository(CL);
14285: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
14286: TCommandLineTools', 'set of TCommandLineTool
14287: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
14288: SIRegister_TJclBorRADToolInstallation(CL);
14289: SIRegister_TJclBCBInstallation(CL);
14290: SIRegister_TJclDelphiInstallation(CL);
14291: SIRegister_TJclDCCIL(CL);
14292: SIRegister_TJclBDSInstallation(CL);
14293: TTraverseMethod', 'Func ( Installation: TJclBorRADToolInstallation ) :Bool
14294: SIRegister_TJclBorRADToolInstallations(CL);
14295: Func BPLFileName( const BPLPath, PackageFileName :Str ) :Str
14296: Func BinaryFileName( const OutputPath, ProjectFileName :Str ) :Str
14297: Func IsDelphiPackage( const FileName :Str ) :Bool
14298: Func IsDelphiProject( const FileName :Str ) :Bool
14299: Func IsBCBPackage( const FileName :Str ) :Bool
14300: Func IsBCBProject( const FileName :Str ) :Bool
14301: Proc GetDPRFileInfo(const DPRFileName:str;out BinaryExtensio:str;const LibSuffix:PString);
14302: Proc GetBPRFileInfo(const BPRFileName:str;out BinaryFileName:str;const Descript:PString);
14303: Proc GetDPKFileInfo(const DPKFileName:str;out RunOnly:Bool;const LibSuffix:PString;const Descript:PString);
14304: Proc GetBPKFileInfo(const BPKFileName:str;out RunOnly:Bool;const BinaryFName:PString;const
Descript:PString
14305: Func SamePath(const Path1, Path2:Str):Bool;
14306: end;
14307:
14308: Proc SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
14309: begin
14310:   'ERROR_NO_MORE_FILES', 'LongInt'( 18 );
14311:   //Func stat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Int
14312:   //Func fstat64( FileDes: Int;var StatBuffer : TStatBuf64 ) : Int
14313:   //Func lstat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Int
14314:   'LPathSeparator', 'String' '/'
14315:   'LDirDelimiter', 'String' '/'
14316:   'LDirSeparator', 'String' ':'

```

```

14317: 'JXPathDevicePrefix','String '\\.\
14318: 'JXPathSeparator','String \'
14319: 'JXDirDelimiter','String \'
14320: 'JXDirSeparator','String \'
14321: 'JXPathUncPrefix','String '\\
14322: 'faNormalFile','LongWord') ( $00000080);
14323: //faUnixSpecific',' faSymLink);
14324: JXTCompactPath',' ( cpCenter, cpEnd )
14325: WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
14326: + 'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime : '
14327: + ' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
14328: TWin32FileAttributeData', ' WIN32_FILE_ATTRIBUTE_DATA
14329: WIN32_FILE_ATTRIBUTE_DATA', ' WIN32_FILE_ATTRIBUTE_DATA
14330: Func jxPathAddSeparator( const Path :Str) :Str
14331: Func jxPathAddExtension( const Path, Extension :Str) :Str
14332: Func jxPathAppend( const Path, Append :Str) :Str
14333: Func jxPathBuildRoot( const Drive :Byte) :Str
14334: Func jxPathCanonicalize( const Path :Str) :Str
14335: Func jxPathCommonPrefix( const Path1, Path2 :Str) : Int
14336: Func jxPathCompactPath(const DC:HDC;const Path:str;const Width:Int;CmpFmt:TCompactPath):str
14337: Proc jxPathExtractElements(const Source:str;var Drive,Path,FileName,Ext:str)
14338: Func jxPathExtractFileDirFixed( const S :Str) :Str
14339: Func jxPathExtractFileNameNoExt( const Path :Str) :Str
14340: Func jxPathExtractPathDepth( const Path :Str; Depth : Int) :Str
14341: Func jxPathGetDepth( const Path :Str) : Int
14342: Func jxPathGetLongName( const Path :Str) :Str
14343: Func jxPathGetShortName( const Path :Str) :Str
14344: Func jxPathGetLongName( const Path :Str) :Str
14345: Func jxPathGetShortName( const Path :Str) :Str
14346: Func jxPathGetRelativePath( Origin, Destination :Str) :Str
14347: Func jxPathGetTempPath :Str
14348: Func jxPathIsAbsolute( const Path :Str) :Bool
14349: Func jxPathIsChild( const Path, Base :Str) :Bool
14350: Func jxPathIsDiskDevice( const Path :Str) :Bool
14351: Func jxPathIsUNC( const Path :Str) :Bool
14352: Func jxPathRemoveSeparator( const Path :Str) :Str
14353: Func jxPathRemoveExtension( const Path :Str) :Str
14354: Func jxPathGetPhysicalPath( const LocalizedPath :Str) :Str
14355: Func jxPathGetLocalizedPath( const PhysicalPath :Str) :Str
14356: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
14357: JxTFileListOptions', 'set of TFileListOption
14358: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
14359: TFileHandler', 'Proc ( const FileName :Str)
14360: TFileHandlerEx', 'Proc ( const Directory :Str; const FileInfo : TSearchRec)
14361: Func jxBuildFileList( const Path :Str; const Attr : Int; const List : TStrings) :Bool
14362: Func jxAdvBuildFileList( const Path :Str; const Attr : Int; const Files : TStrings; const
AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:str;const
FileMatchFunc:TFileMatchFunc):Bool;
14363: Func jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int): Bool
14364: Func jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Bool;
14365: Func jxFileAttributesStr( const FileInfo : TSearchRec) :Str
14366: Func jxIsFileNameMatch(FileName:str;const Mask:str;const CaseSensitive:Bool):Bool;
14367: Proc jxEnumFiles(const
Path:str;HandleFile:TFileHandlerEx;RejectedAttributes:Int;RequiredAttributes:Int;Abort:TObject)
14368: Proc jxEnumDirectories(const Root:str;const HandleDirectory:TFileHandler;const
IncludeHiddenDirects:Boolean;const SubDirectoriesMask:str;Abort:TObject;ResolveSymLinks:Bool)
14369: Proc jxCreateEmptyFile( const FileName :Str)
14370: Func jxCloseVolume( var Volume : THandle) :Bool
14371: Func jxDeleteDirectory(const DirectoryName:str; MoveToRecycleBin:Boolean:Boolean)
14372: Func jxCopyDirectory( ExistingDirectoryName, NewDirectoryName :Str) :Bool
14373: Func jxMoveDirectory( ExistingDirectoryName, NewDirectoryName :Str) :Bool
14374: Func jxDelTree( const Path :Str) :Bool
14375: //Func DelTreeEx(const Path:str;AbortOnFailure:Bool; Progress:TDelTreeProgress):Bool
14376: Func jxDiskInDrive( Drive : Char) :Bool
14377: Func jxDirectoryExists( const Name :Str; ResolveSymLinks :Bool) :Bool
14378: Func jxFileCreateTemp( var Prefix :Str) : THandle
14379: Func jxFileBackup( const FileName:Str; Move :Bool) :Bool
14380: Func jxFileCopy(const ExistingFileName,NewFileName:str; ReplaceExisting:Boolean):Bool
14381: Func jxFileDelete( const FileName :Str; MoveToRecycleBin :Bool) :Bool
14382: Func jxFileExists( const FileName :Str) :Bool
14383: Func jxFileMove(const ExistingFileName,NewFileName:Str;ReplaceExisting:Bool) :Boolean
14384: Func jxFileRestore( const FileName :Str) :Bool
14385: Func jxGetBackupFileName( const FileName :Str) :Str
14386: Func jxIsBackupFileName( const FileName :Str) :Bool
14387: Func jxFileGetDisplayName( const FileName :Str) :Str
14388: Func jxFileGetGroupName( const FileName :Str; ResolveSymLinks :Bool) :Str
14389: Func jxFileGetOwnerName( const FileName :Str; ResolveSymLinks :Bool) :Str
14390: Func jxFileGetSize( const FileName :Str) : Int64
14391: Func jxFileGetTempName( const Prefix :Str) :Str
14392: Func jxFileGetTypeNames( const FileName :Str) :Str
14393: Func jxFindUnusedFileName(FileName:str;const FileExt:str;NumberPrefix:str):str
14394: Func jxForceDirectories( Name :Str) :Bool
14395: Func jxGetDirectorySize( const Path :Str) : Int64
14396: Func jxGetDriveTypeStr( const Drive : Char) :Str
14397: Func jxGetFileAgeCoherence( const FileName :Str) :Bool
14398: Proc jxGetFileAttributeList( const Items : TStrings; const Attr : Int)
14399: Proc jxGetFileAttributeListEx( const Items : TStrings; const Attr : Int)
14400: Func jxGetFileInformation( const FileName :Str; out FileInfo : TSearchRec) :Bool;
14401: Func jxGetFileInformation1( const FileName :Str) : TSearchRec;

```

```

14402: //Func GetFileStatus(const FileName:str;out StatBuf:TStatBuf64;const ResolveSymLinks:Boolean):Int
14403: Func jxGetFileLastWrite( const FName :Str) : TFileTime;
14404: Func jxGetFileLastWrite( const FName :Str; out LocalTime : TDateTime) :Bool;
14405: Func jxGetFileLastAccess( const FName :Str) : TFileTime;
14406: Func jxGetFileLastAccess( const FName:str;out LocalTime : TDateTime) :Bool;
14407: Func jxGetFileCreation( const FName :Str) : TFileTime;
14408: Func jxGetFileCreation( const FName :Str; out LocalTime : TDateTime) :Bool;
14409: Func jxGetFileLastWrite( const FName:str;out TimeStamp:Int;ResolveSymLinks:Bool):Bool;
14410: Func jxGetFileLastWrite( const FName:str;out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool
14411: Func jxGetFileLastWrite2( const FName :Str; ResolveSymLinks :Bool) : Int;
14412: Func jxGetFileLastAccess( const FName:str;out TimeStamp:Int;ResolveSymLinks: Bool) : Bool;
14413: Func jxGetFileLastAccess( const FName:str;out LocalTime:TDateTime;ResolveSymLinks:Bool) :Bool
14414: Func jxGetFileLastAccess2( const FName:str; ResolveSymLinks:Boolean) : Int;
14415: Func jxGetFileLastAttrChange( const FName:str;out TimeStamp:Int;ResolveSymLinks:Bool):Bool;
14416: Func jxGetFileLastAttrChange( const FName:str;out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
14417: Func jxGetFileLastAttrChange2( const FName :Str; ResolveSymLinks:Boolean) : Int;
14418: Func jxGetModulePath( const Module : HMODULE) :Str
14419: Func jxGetSizeOfFile( const FileName :Str) : Int64;
14420: Func jxGetSizeOfFile( const FileInfo : TSearchRec) : Int64;
14421: Func jxGetSizeOfFile2( Handle : THandle) : Int64;
14422: Func jxGetStandardFileInfo( const FileName :Str) : TWin32FileAttributeData
14423: Func jxIsDirectory( const FileName :Str; ResolveSymLinks :Bool) :Bool
14424: Func jxIsRootDirectory( const CanonicFileName :Str) :Bool
14425: Func jxLockVolume( const Volume :Str; var Handle : THandle) :Bool
14426: Func jxOpenVolume( const Drive : Char) : THandle
14427: Func jxSetDirLastWrite( const DirName :Str; const DateTime : TDateTime) :Bool
14428: Func jxSetDirLastAccess( const DirName :Str; const DateTime : TDateTime) :Bool
14429: Func jxSetDirCreation( const DirName :Str; const DateTime : TDateTime) :Bool
14430: Func jxSetFileLastWrite( const FileName :Str; const DateTime : TDateTime) :Bool
14431: Func jxSetFileLastAccess( const FileName :Str; const DateTime : TDateTime) :Bool
14432: Func jxSetFileCreation( const FileName :Str; const DateTime : TDateTime) :Bool
14433: Proc jxShredFile( const FileName :Str; Times : Int)
14434: Func jxUnlockVolume( var Handle : THandle) :Bool
14435: Func jxCreateSymbolicLink( const Name, Target :Str) :Bool
14436: Func jxSymbolicLinkTarget( const Name :Str) :Str
14437: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
14438: SIRegister_TJclCustomFileAttrMask(CL);
14439: SIRegister_TJclFileAttributeMask(CL);
14440: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
14441: + 'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
14442: TFileSearchOptions', 'set of TFileSearchOption
14443: TFileSearchTaskID', 'Int
14444: TFileSearchTerminationEvent', 'Procedure(const ID:TFileSearchTaskID;const Aborted:Boolean)
14445: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
14446: SIRegister_IJclFileEnumerator(CL);
14447: SIRegister_TJclFileEnumerator(CL);
14448: Func JxFileSearch : IJclFileEnumerator
14449: JxTFileFlag', '(ffDebug,ffInfoInferred,ffPatched,ffPreRelease,ffPrivateBuild,ffSpecialBuild )
14450: JxTFileFlags', 'set of TFileFlag
14451: FindClass('TOBJECT'),'EJclFileVersionInfoError
14452: SIRegister_TJclFileVersionInfo(CL);
14453: Func jxOSIdentToString( const OSIdent : DWORD) :Str
14454: Func jxOSFileTypeToString(const OSFileType: DWORD; const OSFileSubType : DWORD) :Str
14455: Func jxVersionResourceAvailable( const FileName :Str) :Bool
14456: TFileVersionFormat', '( vfMajorMinor, vfFull )
14457: Func jxFormatVersionString( const HiV, LoV : Word) :Str;
14458: Func jxFormatVersionString1( const Major, Minor, Build, Revision : Word) :Str;
14459: //Func FormatVersionString2(const FixedInfo:TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
14460: //Proc VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
14461: //Proc VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
14462: //Func VersionFixedFileInfo(const FileName:str;var FixedInfo:TVSFixedFileInfo):Boolean
14463: Func jxVersionFixedFileInfoString(const FileName:str;VersionFormat:TFileVersionFormat;const
NotAvailableText :str):str
14464: SIRegister_TJclTempFileStream(CL);
14465: FindClass('TOBJECT'),'TJclCustomFileMapping
14466: SIRegister_TJclFileMapView(CL);
14467: TJclFileMappingRoundOffset', '( rvDown, rvUp )
14468: SIRegister_TJclCustomFileMapping(CL);
14469: SIRegister_TJclFileMapping(CL);
14470: SIRegister_TJclSwapFileMapping(CL);
14471: SIRegister_TJclFileMappingStream(CL);
14472: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
14473: //PPCharArray', '^TPCharArray // will not work
14474: SIRegister_TJclMappedTextReader(CL);
14475: SIRegister_TJclFileMaskComparator(CL);
14476: FindClass('TOBJECT'),'EJclPathError
14477: FindClass('TOBJECT'),'EJclFileUtilsError
14478: FindClass('TOBJECT'),'EJclTempFileStreamError
14479: FindClass('TOBJECT'),'EJclTempFileStreamError
14480: FindClass('TOBJECT'),'EJclFileMappingError
14481: FindClass('TOBJECT'),'EJclFileMapViewError
14482: Func jxPathGetLongName2( const Path :Str) :Str
14483: Func jxWin32DeleteFile( const FileName :Str; MoveToRecycleBin :Bool) :Bool
14484: Func jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName :Str) :Bool
14485: Func jxWin32BackupFile( const FileName :Str; Move :Bool) :Bool
14486: Func jxWin32RestoreFile( const FileName :Str) :Bool
14487: Func jxSamePath( const Path1, Path2 :Str) :Bool
14488: Proc jxPathListAddItems( var List :Str; const Items :Str)
14489: Proc jxPathListIncludeItems( var List :Str; const Items :Str)

```

```

14490: Proc jxPathListDelItems( var List :Str; const Items :Str)
14491: Proc jxPathListDelItem( var List :Str; const Index : Int)
14492: Func jxPathListItemCount( const List :Str) : Int
14493: Func jxPathListGetItem( const List :Str; const Index : Int) :Str
14494: Proc jxPathListSetItem( var List :Str; const Index : Int; const Value :Str)
14495: Func jxPathListItemIndex( const List, Item :Str) : Int
14496: Func jxParamName(Idx:Int;const Separator:str;const AllowedPrefixChars:str;TrimName:Bool):str;
14497: Func jxParamValue(Index:Int; const Separator :Str; TrimValue :Bool) : Str;
14498: Func jxParamValue1(const SearchName:str; const Separator :Str; CaseSensitive :Bool;
14499: const AllowedPrefixCharacters :Str; TrimValue :Bool) :Str;
14500: Func jxParamPos(const SearchName:Str;const Separator:str;CaseSensitive:Boolean;const
AllowedPrefixCharacters string): Int
14501: end;
14502:
14503: Proc SIRegister_FileUtil(CL: TPSPascalCompiler);
14504: begin
14505: 'UTF8FileHeader','String #Sef#Sbb#Sbf);
14506: Func lCompareFileNames( const Filename1, Filename2 :Str) : Int
14507: Func lCompareFileNamesIgnoreCase( const Filename1, Filename2 :Str) : Int
14508: Func lCompareFileNames(const Filename1,Filename2 :Str; ResolveLinks :Bool): Int
14509: Func lCompareFileNames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLiks:bool):int;
14510: Func lFilenameIsAbsolute( const TheFilename :Str) :Bool
14511: Func lFilenameIsWinAbsolute( const TheFilename :Str) :Bool
14512: Func lFilenameIsUnixAbsolute( const TheFilename :Str) :Bool
14513: Proc lCheckIfFileIsExecutable( const AFilename :Str)
14514: Proc lCheckIfFileIsSymlink( const AFilename :Str)
14515: Func lFileIsReadable( const AFilename :Str) :Bool
14516: Func lFileIsWritable( const AFilename :Str) :Bool
14517: Func lFileIsText( const AFilename :Str) :Bool
14518: Func lFileIsText( const AFilename :Str; out FileReadable :Bool) :Bool
14519: Func lFileIsExecutable( const AFilename :Str) :Bool
14520: Func lFileIsSymlink( const AFilename :Str) :Bool
14521: Func lFileIsHardLink( const AFilename :Str) :Bool
14522: Func lFileSize( const Filename :Str) : int64;
14523: Func lGetFileDescription( const AFilename :Str) :Str
14524: Func lReadAllLinks( const Filename :Str; ExceptionOnError :Bool) :Str
14525: Func lTryReadAllLinks( const Filename :Str) :Str
14526: Func lDirPathExists( const FileName :Str) :Bool
14527: Func lForceDirectory( DirectoryName :Str) :Bool
14528: Func lDeleteDirectory(const DirectoryName:Str; OnlyChildren:Bool) :Bool
14529: Func lProgramDirectory :Str
14530: Func lDirectoryIsWritable( const DirectoryName :Str) :Bool
14531: Func lExtractFileNameOnly( const AFilename :Str) :Str
14532: Func lExtractFileNameWithoutExt( const AFilename :Str) :Str
14533: Func lCompareFileExt( const Filename, Ext :Str; CaseSensitive :Bool) : Int;
14534: Func lCompareFileExt( const Filename, Ext :Str) : Int;
14535: Func lFilenameIsPascalUnit( const Filename :Str) :Bool
14536: Func lAppendPathDelim( const Path :Str) :Str
14537: Func lChompPathDelim( const Path :Str) :Str
14538: Func lTrimFilename( const AFilename :Str) :Str
14539: Func lCleanAndExpandFilename( const Filename :Str) :Str
14540: Func lCleanAndExpandDirectory( const Filename :Str) :Str
14541: Func lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory :Str) :Str
14542: Func lCreateRelativePath(const Filename,BaseDirectory
string;UsePointDirectory:boolean;AlwaysRequireSharedBaseFolder:Bool):str
14543: Func lCreateAbsolutePath( const Filename, BaseDirectory :Str) :Str
14544: Func lFileIsInPath( const Filename, Path :Str) :Bool
14545: Func lFileIsInDirectory( const Filename, Directory :Str) :Bool
14546: TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
14547: TSearchFileInPathFlags', 'set of TSearchFileInPathFlag AllDirectoryEntriesMask','String '*
14548: Func lGetAllFilesMask :Str
14549: Func lGetExeExt :Str
14550: Func lSearchFileInPath(const Filename,BasePath,SearchPath,Delimiter:Str;Flags:TSearchFileInPathFlags):str
14551: Func lSearchAllFilesInPath(const Filename,BasePath,SearchPath,Delimiter:str;Flags:
TSearchFileInPathFlags):TStrings
14552: Func lFindDiskFilename( const Filename :Str) :Str
14553: Func lFindDiskFileCaseInsensitive( const Filename :Str) :Str
14554: Func lFindDefaultExecutablePath( const Executable :Str; const BaseDir:Str):str
14555: Func lGetDarwinSystemFilename( Filename :Str) :Str
14556: SIRegister_TFileIterator(CL);
14557: TFileFoundEvent', 'Proc ( FileIterator : TFileIterator)
14558: TDirectoryFoundEvent', 'Proc ( FileIterator : TFileIterator)
14559: TDirectoryEnterEvent', 'Proc ( FileIterator : TFileIterator)
14560: SIRegister_TFileSearcher(CL);
14561: Func lFindAllFiles(const SearchPath:str;SearchMsk:str;SearchSubDirs:Bool):TStringList
14562: Func lFindAllDirectories(const SearchPath:Str; SearchSubDirs:Bool): TStringList
14563: // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
14564: // TCopyFileFlags', 'set of TCopyFileFlag
14565: Func lCopyFile(const SrcFilename, DestFilename:Str; Flags : TCopyFileFlags):Bool
14566: Func lCopyFile(const SrcFilename, DestFilename:Str; PreserveTime :Bool):Bool
14567: Func lCopyDirTree( const SourceDir, TargetDir :Str; Flags : TCopyFileFlags):Bool
14568: Func lReadFileToString( const Filename :Str) :Str
14569: Func lGetTempFilename( const Directory, Prefix :Str) :Str
14570: {Func NeedRTLAnsi :Bool
14571: Proc SetNeedRTLAnsi( NewValue :Bool)
14572: Func UTF8ToSys( const s :Str) :Str
14573: Func SysToUTF8( const s :Str) :Str
14574: Func ConsoleToUTF8( const s :Str) :Str
14575: Func UTF8ToConsole( const s :Str) :Str}

```

```

14576: Func FileExistsUTF8( const Filename :Str) :Bool
14577: Func FileAgeUTF8( const FileName :Str) : Longint
14578: Func DirectoryExistsUTF8( const Directory :Str) :Bool
14579: Func ExpandFileNameUTF8( const FileName :Str) :Str
14580: Func ExpandUNCFileNameUTF8( const FileName :Str) :Str
14581: Func ExtractShortPathNameUTF8( const FileName :Str) :Str
14582: Func FindFirstUTF8(const Path:Str; Attr : Longint; out Rslt : TSearchRec) : Longint
14583: Func FindNextUTF8( var Rslt : TSearchRec) : Longint
14584: Proc FindCloseUTF8( var F : TSearchrec)
14585: Func FileSetDateUTF8( const FileName :Str; Age : Longint) : Longint
14586: Func FileGetAttrUTF8( const FileName :Str) : Longint
14587: Func FileSetAttrUTF8( const Filename :Str; Attr : longint) : Longint
14588: Func DeleteFileUTF8( const FileName :Str) :Bool
14589: Func RenameFileUTF8( const OldName, NewName :Str) :Bool
14590: Func FileSearchUTF8(const Name, DirList :Str; ImplicitCurrentDir :Bool) :Str
14591: Func FileIsReadOnlyUTF8( const FileName :Str) :Bool
14592: Func GetCurrentDirUTF8 :Str
14593: Func SetCurrentDirUTF8( const NewDir :Str) :Bool
14594: Func CreateDirUTF8( const NewDir :Str) :Bool
14595: Func RemoveDirUTF8( const Dir :Str) :Bool
14596: Func ForceDirectoriesUTF8( const Dir :Str) :Bool
14597: Func FileOpenUTF8( const FileName :Str; Mode : Int) : THandle
14598: Func FileCreateUTF8( const FileName :Str) : THandle;
14599: Func FileCreateUTF81( const FileName :Str; Rights :Card) : THandle;
14600: Func ParamStrUTF8( Param : Int) :Str
14601: Func GetEnvironmentStringUTF8( Index : Int) :Str
14602: Func GetEnvironmentVariableUTF8( const EnvVar :Str) :Str
14603: Func GetAppConfigDirUTF8( Global :Bool; Create :Bool) :Str
14604: Func GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir :Bool) :Str
14605: Func SysErrorMessageUTF8( ErrorCode : Int) :Str
14606: end;
14607:
14608: Proc SIRegister_Keyboard(CL: TPSPascalCompiler);
14609: begin
14610:   //VK_F23 = 134;
14611:   //{$EXTERNALSYM VK_F24}
14612:   //VK_F24 = 135;
14613:   TVirtualKeyCode, 'Int
14614:   'VK_MOUSEWHEELUP','Int'(134);
14615:   'VK_MOUSEWHEELDOWN','Int'(135);
14616:   Func glIsKeyDown( c : Char) :Bool;
14617:   Func glIsKeyDown1( vk : TVirtualKeyCode) :Bool;
14618:   Func glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
14619:   Func glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) :Str
14620:   Func glKeyNameToVirtualKeyCode( const keyName :Str) : TVirtualKeyCode
14621:   Func glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
14622:   Proc glKeyboardNotifyWheelMoved( wheelDelta : Int)
14623: end;
14624:
14625: Proc SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14626: begin
14627:   TGLPoint, 'TPoint
14628:   //PGLPoint, '^TGLPoint // will not work
14629:   TGLRect, 'TRect
14630:   //PGLRect, '^TGLRect // will not work
14631:   TDelphiColor, 'TColor
14632:   TGLPicture, 'TPicture
14633:   TGLGraphic, 'TGraphic
14634:   TGLBitmap, 'TBitmap
14635:   //TGraphicClass, 'class of TGraphic
14636:   TGLTextLayout, '( tlTop, tlCenter, tlBottom )
14637:   TGLMouseButton, '( mbLeft, mbRight, mbMiddle )
14638:   TGLMouseEvent, 'Proc ( Sender : TObject; Button : TGLMouse'
14639:   +'Button; Shift : TShiftState; X, Y : Int)
14640:   TGLMouseMoveEvent, 'TMouseMoveEvent
14641:   TGLKeyEvent, 'TKeyEvent
14642:   TGLKeyPressEvent, 'TKeyPressEvent
14643:   EGLOSError, 'EWin32Error
14644:   EGLOSError, 'EWin32Error
14645:   EGLOSError, 'EOSError
14646:   'glsAllFilter','string'All // sAllFilter
14647:   Func GLPoint( const x, y : Int) : TGLPoint
14648:   Func GLRGB( const r, g, b : Byte) : TColor
14649:   Func GLColorToRGB( color : TColor) : TColor
14650:   Func GLGetRValue( rgb : DWORD) : Byte
14651:   Func GLGetGValue( rgb : DWORD) : Byte
14652:   Func GLGetBValue( rgb : DWORD) : Byte
14653:   Proc GLInitWinColors
14654:   Func GLRect( const aLeft, aTop, aRight, aBottom : Int) : TGLRect
14655:   Proc GLInflateGLRect( var aRect : TGLRect; dx, dy : Int)
14656:   Proc GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14657:   Proc GLInformationDlg( const msg :Str)
14658:   Func GLQuestionDlg( const msg :Str) :Bool
14659:   Func GLInputDialog( const aCaption, aPrompt, aDefault :Str) :Str
14660:   Func GLSavePictureDialog( var aFileName :Str; const aTitle :Str) :Bool
14661:   Func GLOpenPictureDialog( var aFileName :Str; const aTitle :Str) :Bool
14662:   Func GLApplicationTerminated :Bool
14663:   Proc GLRaiseLastOSError
14664:   Proc GLFreeAndNil( var anObject: TObject)

```



```

14665: Func GLGetDeviceLogicalPixelsX( device :Card) : Int
14666: Func GLGetCurrentColorDepth : Int
14667: Func GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Int
14668: Func GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Int) : Pointer
14669: Proc GLSleep( length :Card)
14670: Proc GLQueryPerformanceCounter( var val : Int64)
14671: Func GLQueryPerformanceFrequency( var val : Int64) :Bool
14672: Func GLStartPrecisionTimer : Int64
14673: Func GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14674: Func GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14675: Func GLRDTSC : Int64
14676: Proc GLLoadBitmapFromInstance( ABitmap : TBitmap; AName :Str)
14677: Func GLOKMessageBox( const Text, Caption :Str) : Int
14678: Proc GLShowHTMLUrl( Url :Str)
14679: Proc GLShowCursor( AShow :Bool)
14680: Proc GLSetCursorPos( AScreenX, AScreenY : Int)
14681: Proc GLGetCursorPos( var point : TGLPoint)
14682: Func GLGetScreenWidth : Int
14683: Func GLGetScreenHeight : Int
14684: Func GLGetTickCount : int64
14685: Func RemoveSpaces(const str :Str) :Str;
14686: TNormalMapSpace', '( nmsObject, nmsTangent )
14687: Proc CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14688: Proc SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14689: Proc CalcTangentSpaceLightVectors(Light:TAffineVector;Vertices,Normals,Tangents,
BiNormals:TAffineVectorList;Colors:TVectorList)
14690: Func CreateObjectSpaceNormalMap(Width,Height:Int;HiNormals,HiTexCoords:TAffineVectorList):TGLBitmap
14691: Func CreateTangentSpaceNormalMap(Width, Height:Int;HiNormals, HiTexCoords,LoNormals,LoTexCoords,Tangents,
14692: BiNormals : TAffineVectorList) : TGLBitmap
14693: end;
14694:
14695: Proc SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14696: begin
14697:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14698:   // PGLStarRecord', '^TGLStarRecord // will not work
14699:   Func StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14700:   Func StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14701:   Func StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14702: end;
14703:
14704: Proc SIRegister_GeometryBB(CL: TPSPascalCompiler);
14705: begin
14706:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14707:   //PAABB', '^TAABB // will not work
14708:   TBSphere', 'record Center : TAffineVector; Radius : single; end
14709:   TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14710:   TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14711:   Func AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14712:   Proc AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14713:   Proc SetBB( var c : THmgBoundingBox; const v : TVector)
14714:   Proc SetAABB( var bb : TAABB; const v : TVector)
14715:   Proc BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14716:   Proc AABBBTransform( var bb : TAABB; const m : TMatrix)
14717:   Proc AABBScale( var bb : TAABB; const v : TAffineVector)
14718:   Func BBMinX( const c : THmgBoundingBox) : Single
14719:   Func BBMaxX( const c : THmgBoundingBox) : Single
14720:   Func BBMinY( const c : THmgBoundingBox) : Single
14721:   Func BBMaxY( const c : THmgBoundingBox) : Single
14722:   Func BBMinZ( const c : THmgBoundingBox) : Single
14723:   Func BBMaxZ( const c : THmgBoundingBox) : Single
14724:   Proc AABBIInclude( var bb : TAABB; const p : TAffineVector)
14725:   Proc AABBFFromSweep(var SweepAABB: TAABB; const Start, Dest:TVector; const Radius:Single)
14726:   Func AABBIIntersection( const aabb1, aabb2 : TAABB) : TAABB
14727:   Func BBToAABB( const aabb : THmgBoundingBox) : TAABB
14728:   Func AABBTToBB( const anAABB : TAABB) : THmgBoundingBox;
14729:   Func AABBTToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14730:   Proc OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14731:   Proc OffsetAABB1( var aabb : TAABB; const delta : TVector);
14732:   Func IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) :Bool;
14733:   Func IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) :Bool
14734:   Func IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) :Bool
14735:   Func IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) :Bool
14736:   Func AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) :Bool
14737:   Func PointInAABB( const p : TAffineVector; const aabb : TAABB) :Bool;
14738:   Func PointInAABB1( const p : TVector; const aabb : TAABB) :Bool;
14739:   Func PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) :Bool
14740:   Func TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) :Bool
14741:   Proc ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14742:   Proc AABBTToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14743:   Proc BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14744:   Func BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14745:   Func BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14746:   Func AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14747:   Func BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14748:   Func BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14749:   Func AABBContainsBSphere(const mainAABB:TAABB;const testBSphere:TBSphere): TSpaceContains
14750:   Func PlaneContainsBSphere(const Location,Normal:TAffineVector;const testBSphere:TBSphere):TSpaceContains
14751:   Func FrustumContainsBSphere(const Frustum:TFrustum;const testBSphere:TBSphere): TSpaceContains
14752:   Func FrustumContainsAABB( const Frustum:TFrustum;const testAABB:TAABB) : TSpaceContains

```

```

14753: Func ClipToAABB( const v : TAffineVector; const AABB : TAABB ) : TAffineVector
14754: Func BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere ) : Bool
14755: Proc IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14756: Func AABBTToClipRect( const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
viewportSizeY:Int);TClipRect
14757: end;
14758:
14759: Proc SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14760: begin
14761: Proc Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14762: Proc Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14763: Proc Cylindrical_Cartesian2(const r,theta,z1: single; var x,y,z : single; var ierr : Int);
14764: Proc Cylindrical_Cartesian3(const r,theta,z1: double; var x,y,z: double; var ierr : Int);
14765: Proc Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14766: Proc Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14767: Proc Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14768: Proc Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14769: Proc Spherical_Cartesian2(const r,theta, phi: single;var x,y,z:single;var ierr:Int);
14770: Proc Spherical_Cartesian3(const r,theta, phi:double;var x,y,z:double;var ierr:Int);
14771: Proc Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14772: Proc Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14773: Proc Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14774: Proc ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14775: Proc ProlateSpheroidal_Cartesian1(const xi,eta,phi,a: double;var x,y, z : double);
14776: Proc ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:singl;var x,y,z:single;var ierr:Int);
14777: Proc ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:doubl;var x,y,z:double;var ierr:Int);
14778: Proc OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14779: Proc OblateSpheroidal_Cartesian1(const xi, eta, phi, a : double; var x, y, z : double);
14780: Proc OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr:Int);
14781: Proc OblateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr:Int);
14782: Proc BipolarCylindrical_Cartesian(const u, v, z1, a: single; var x, y, z : single);
14783: Proc BipolarCylindrical_Cartesian1(const u, v, z1, a: double; var x, y, z : double);
14784: Proc BipolarCylindrical_Cartesian2(const u,v,z1,a:single;var x,y,z:single;var ierr:Int);
14785: Proc BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double;var ierr:Int);
14786: end;
14787:
14788: Proc SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14789: begin
14790: 'EPSILON','Single').setExtended( 1e-40);
14791: 'EPSILON2','Single').setExtended( 1e-30); }
14792: TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14793: +irection : TVector; viewPortRadius:Single;farClippingDistance:Single;frustum:TFrustum;end
14794: THmgPlane', 'TVector
14795: TDoubleHmgPlane', 'THomogeneousDblVector
14796: {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14797: + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14798: +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW })
14799: TSingleArray', 'array of Single
14800: TTransformations','array [0..15] of Single)
14801: TPackedRotationMatrix','array [0..2] of Smallint)
14802: TVertex', 'TAffineVector
14803: //TVectorGL', 'THomogeneousFltVector
14804: //TMatrixGL', 'THomogeneousFltMatrix
14805: // TPackedRotationMatrix = array [0..2] of SmallInt;
14806: Func glTexPointMake( const s, t : Single ) : TTexPoint
14807: Func glAffineVectorMake( const x, y, z : Single ) : TAffineVector;
14808: Func glAffineVectorMakel( const v : TVectorGL ) : TAffineVector;
14809: Proc glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14810: Proc glSetVector( var v : TAffineVector; const x, y, z : Single);
14811: Proc glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14812: Proc glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14813: Proc glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14814: Proc glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14815: Func glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14816: Func glVectorMakel( const x, y, z : Single; w : Single) : TVectorGL;
14817: Func glPointMake( const x, y, z : Single) : TVectorGL;
14818: Func glPointMakel( const v : TAffineVector) : TVectorGL;
14819: Func glPointMake2( const v : TVectorGL) : TVectorGL;
14820: Proc glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14821: Proc glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14822: Proc glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14823: Proc glMakePoint( var v : TVectorGL; const x, y, z : Single);
14824: Proc glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14825: Proc glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14826: Proc glMakeVector( var v : TAffineVector; const x, y, z : Single);
14827: Proc glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14828: Proc glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14829: Proc glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14830: Proc glRstVector( var v : TAffineVector);
14831: Proc glRstVector1( var v : TVectorGL);
14832: Func glVectorAdd( const v1, v2 : TAffineVector ) : TAffineVector;
14833: Proc glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14834: //Proc VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14835: Func glVectorAdd3( const v1, v2 : TVectorGL ) : TVectorGL;
14836: Proc glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14837: Func glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14838: Func glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14839: Proc glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14840: Proc glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);

```

```

14841: Proc glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14842: Proc glAddVector10( var v : TAffineVector; const f : Single);
14843: Proc glAddVector11( var v : TVectorGL; const f : Single);
14844: //Proc TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Int;dest:PTexPointArray);
14845: //Proc TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const nb:Int;const scale:
TTexPoint; dest : PTexPointArray);
14846: //Proc VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Int;dest:
PAffineVectorArray);
14847: Func glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14848: Proc glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14849: Proc glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14850: Proc glVectorSubtract3(const v1: TVectorGL; v2: TAffineVector; var result : TVectorGL);
14851: Func glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14852: Proc glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14853: Proc glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14854: Func glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14855: Func glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14856: Proc glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14857: Proc glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14858: Proc glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14859: //Proc CombineVector1(var vr: TAffineVector; const v : TAffineVector; pf : PFloat);
14860: Func glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14861: Func glVectorCombine2(const V1,V2: TAffineVector; const F1, F2 : Single) : TAffineVector;
14862: Func glVectorCombine3( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14863: Proc glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14864: Proc glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14865: Proc glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14866: Func glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14867: Func glVectorCombine8(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single): TVectorGL;
14868: Proc glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14869: Proc glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14870: Proc glVectorCombine11(const V1,V2 : TVectorGL; const F2 : Single; var vr: TVectorGL);
14871: Func glVectorCombine3(const V1,V2,V3: TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14872: Proc glVectorCombine31(const V1, V2, V3: TVectorGL; const F1,F2, F3:Single; var vr : TVectorGL);
14873: Func glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14874: Func glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14875: Func glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14876: Func glPointProject( const p, origin, direction : TAffineVector) : Single;
14877: Func glPointProject1( const p, origin, direction : TVectorGL) : Single;
14878: Func glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14879: Func glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14880: Proc glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14881: Proc glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14882: Proc glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14883: Proc glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14884: Func glLerp( const start, stop, t : Single) : Single
14885: Func glAngleLerp( start, stop, t : Single) : Single
14886: Func glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14887: Func glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14888: Func glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14889: Proc glVectorLerp1(const v1,v2: TAffineVector; t : Single; var vr : TAffineVector);
14890: Func glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14891: Proc glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14892: Func glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14893: Func glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14894: //Proc VectorArrayLerp(const src1,src2:PVectorArray; t:Single; n:Int; dest:PVectorArray);
14895: //Proc VectorArrayLerp1(const src1,src2:PAffineVectorArray; t:Single;n:Int;dest: PAffineVectorArray);
14896: Func glVectorLength( const x, y : Single) : Single;
14897: Func glVectorLength1( const x, y, z : Single) : Single;
14898: Func glVectorLength2( const v : TAffineVector) : Single;
14899: Func glVectorLength3( const v : TVectorGL) : Single;
14900: Func glVectorLength4( const v : array of Single) : Single;
14901: Func glVectorNorm( const x, y : Single) : Single;
14902: Func glVectorNorm1( const v : TAffineVector) : Single;
14903: Func glVectorNorm2( const v : TVectorGL) : Single;
14904: Func glVectorNorm3( var V : array of Single) : Single;
14905: Proc glNormalizeVector( var v : TVectorGL);
14906: Proc glNormalizeVector1( var v : TVectorGL);
14907: Func glVectorNormalize( const v : TAffineVector) : TAffineVector;
14908: Func glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14909: // Proc NormalizeVectorArray( list : PAffineVectorArray; n : Int);
14910: Func glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14911: Func glVectorNegate( const v : TAffineVector) : TAffineVector;
14912: Func glVectorNegate1( const v : TVectorGL) : TVectorGL;
14913: Proc glNegateVector( var V : TAffineVector);
14914: Proc glNegateVector2( var V : TVectorGL);
14915: Proc glNegateVector3( var V : array of Single);
14916: Proc glScaleVector( var v : TAffineVector; factor : Single);
14917: Proc glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14918: Proc glScaleVector2( var v : TVectorGL; factor : Single);
14919: Proc glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14920: Func glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14921: Proc glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14922: Func glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14923: Proc glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14924: Proc glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14925: Proc glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14926: Func glVectorEquals( const V1, V2 : TVectorGL) : Bool;
14927: Func glVectorEquals1( const V1, V2 : TAffineVector) : Bool;

```

```

14928: Func glAffineVectorEquals( const V1, V2 : TVectorGL) :Bool;
14929: Func glVectorIsNull( const v : TVectorGL) :Bool;
14930: Func glVectorIsNull1( const v : TAffineVector) :Bool;
14931: Func glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14932: Func glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14933: Func glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14934: Func glVectorDistance( const v1, v2 : TAffineVector) : Single;
14935: Func glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14936: Func glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14937: Func glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14938: Func glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14939: Func glVectorReflect( const V, N : TAffineVector) : TAffineVector
14940: Proc glRotateVector( var vector:TVectorGL; const axis: TAffineVector; angle : Single);
14941: Proc glRotateVector1( var vector:TVectorGL; const axis : TVectorGL; angle : Single);
14942: Proc glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14943: Func glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14944: Func glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14945: Proc glVectorRotateAroundY1( const v:TAffineVector; alpha:Single; var vr : TAffineVector);
14946: Func glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14947: Proc glAbsVector( var v : TVectorGL);
14948: Proc glAbsVector1( var v : TAffineVector);
14949: Func glVectorAbs( const v : TVectorGL) : TVectorGL;
14950: Func glVectorAbs1( const v : TAffineVector) : TAffineVector;
14951: Proc glSetMatrix( var dest : THomogeneousDb1Matrix; const src : TMatrixGL);
14952: Proc glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14953: Proc glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14954: Proc glSetMatrixRow( var dest : TMatrixGL; rowNb : Int; const aRow : TVectorGL);
14955: Func glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14956: Func glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14957: Func glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14958: Func glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14959: Func glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14960: Func glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14961: Func glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14962: Func glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14963: Func glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14964: Func glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14965: Func glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14966: Func glCreateRotationMatrix( const anAxis:TAffineVector; angle: Single) : TMatrixGL;
14967: Func glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14968: Func glCreateAffineRotationMatrix( const anAxis:TAffineVector; angle:Single):TAffineMatrix
14969: Func glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14970: Func glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14971: Proc glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14972: Func glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14973: Func glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14974: Func glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14975: Func glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix):TAffineVector;
14976: Func glMatrixDeterminant( const M : TAffineMatrix) : Single;
14977: Func glMatrixDeterminant1( const M : TMatrixGL) : Single;
14978: Proc glAdjointMatrix( var M : TMatrixGL);
14979: Proc glAdjointMatrix1( var M : TAffineMatrix);
14980: Proc glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14981: Proc glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14982: Proc glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14983: Proc glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14984: Proc glNormalizeMatrix( var M : TMatrixGL);
14985: Proc glTransposeMatrix( var M : TAffineMatrix);
14986: Proc glTransposeMatrix1( var M : TMatrixGL);
14987: Proc glInvertMatrix( var M : TMatrixGL);
14988: Proc glInvertMatrix1( var M : TAffineMatrix);
14989: Func glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14990: Func glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) :Bool
14991: Func glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14992: Func glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14993: Func glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14994: Func glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14995: Proc glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14996: Proc glNormalizePlane( var plane : THmgPlane)
14997: Func glPlaneEvaluatePoint( const plane:THmgPlane; const point: TAffineVector): Single;
14998: Func glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14999: Func glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
15000: Proc glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
15001: Proc glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
15002: Func glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) :Bool;
15003: Func glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) :Bool;
15004: Func glPointPlaneDistance( const point, planePoint, planeNormal: TVectorGL) : Single;
15005: Func glPointPlaneDistance1( const point, planePoint, planeNormal: TAffineVector) : Single;
15006: Func glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
15007: Func glPointSegmentDistance( const point, segmentStart, segmentStop:TAffineVector): single
15008: Func glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
15009: Func glPointLineDistance( const point, linePoint, lineDirection:TAffineVector) : Single
15010: Proc glSegmentSegmentClosestPoint( const S0Start,S0Stop,S1Start,S1Stop TAffineVector;
15011: var Segment0Closest,Segment1Closest: TAffineVector)
15012: Func glSegmentSegmentDistance( const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
15013: TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
15014: Func glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
15015: Func glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
15016: Func glQuaternionMagnitude( const Q : TQuaternion) : Single

```



```

15017: Proc glNormalizeQuaternion( var Q : TQuaternion)
15018: Func glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
15019: Proc glQuaternionToPoints(const Q: TQuaternion; var ArcFrom,ArcTo : TAffineVector)
15020: Func glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
15021: Func glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
15022: Func glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
15023: Func glQuaternionFromAngleAxis(const angle:Single;const axis:TAffineVector) : TQuaternion
15024: Func glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
15025: Func glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
15026: Func glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
15027: Func glQuaternionSlerp(const QStart,QEnd:TQuaternion;Spin:Int; t: Single): TQuaternion;
15028: Func glQuaternionSlerp1(const source, dest:TQuaternion; const t: Single) : TQuaternion;
15029: Func glLnXP1( X : Extended) : Extended
15030: Func glLog10( X : Extended) : Extended
15031: Func glLog2( X : Extended) : Extended;
15032: Func glLog21( X : Single) : Single;
15033: Func glLogN( Base, X : Extended) : Extended
15034: Func glIntPower( Base : Extended; Exponent : Int) : Extended
15035: Func glPower( const Base, Exponent : Single) : Single;
15036: Func glPower1( Base : Single; Exponent : Int) : Single;
15037: Func glDegToRad( const Degrees : Extended) : Extended;
15038: Func glDegToRad1( const Degrees : Single) : Single;
15039: Func glRadToDeg( const Radians : Extended) : Extended;
15040: Func glRadToDeg1( const Radians : Single) : Single;
15041: Func glNormalizeAngle( angle : Single) : Single
15042: Func glNormalizeDegAngle( angle : Single) : Single
15043: Proc glSinCos( const Theta : Extended; var Sin, Cos : Extended);
15044: Proc glSinCos11( const Theta : Double; var Sin, Cos : Double);
15045: Proc glSinCos0( const Theta : Single; var Sin, Cos : Single);
15046: Proc glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
15047: Proc glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
15048: Proc glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
15049: Proc glPrepareSinCosCache(var s,c: array of Single; startAngle,stopAngle : Single)
15050: Func glArcCos( const X : Extended) : Extended;
15051: Func glArcCos1( const x : Single) : Single;
15052: Func glArcSin( const X : Extended) : Extended;
15053: Func glArcSin1( const X : Single) : Single;
15054: Func glArcTan21( const Y, X : Extended) : Extended;
15055: Func glArcTan21( const Y, X : Single) : Single;
15056: Func glFastArcTan2( y, x : Single) : Single
15057: Func glTan( const X : Extended) : Extended;
15058: Func glTan1( const X : Single) : Single;
15059: Func glCoTan( const X : Extended) : Extended;
15060: Func glCoTan1( const X : Single) : Single;
15061: Func glSinh( const x : Single) : Single;
15062: Func glSinh1( const x : Double) : Double;
15063: Func glCosh( const x : Single) : Single;
15064: Func glCosh1( const x : Double) : Double;
15065: Func glRSqrt( v : Single) : Single
15066: Func glRLength( x, y : Single) : Single
15067: Func glISqrt( i : Int) : Int
15068: Func glLLength( x, y : Int) : Int;
15069: Func glLLength1( x, y, z : Int) : Int;
15070: Proc glRegisterBasedExp
15071: Proc glRandomPointOnSphere( var p : TAffineVector)
15072: Func glRoundInt( v : Single) : Single;
15073: Func glRoundInt1( v : Extended) : Extended;
15074: Func glTrunc( v : Single) : Int;
15075: Func glTrunc64( v : Extended) : Int64;
15076: Func glInt( v : Single) : Single;
15077: Func glInt1( v : Extended) : Extended;
15078: Func glFrac( v : Single) : Single;
15079: Func glFrac1( v : Extended) : Extended;
15080: Func glRound( v : Single) : Int;
15081: Func glRound64( v : Single) : Int64;
15082: Func glRound641( v : Extended) : Int64;
15083: Func glTrunc( X : Extended) : Int64
15084: Func glRound( X : Extended) : Int64
15085: Func glFrac( X : Extended) : Extended
15086: Func glCeil( v : Single) : Int;
15087: Func glCeil64( v : Extended) : Int64;
15088: Func glFloor( v : Single) : Int;
15089: Func glFloor64( v : Extended) : Int64;
15090: Func glScaleAndRound( i : Int; var s : Single) : Int
15091: Func glSign( x : Single) : Int
15092: Func glIsInRange( const x, a, b : Single) :Bool;
15093: Func glIsInRangel( const x, a, b : Double) :Bool;
15094: Func glIsInCube( const p, d : TAffineVector) :Bool;
15095: Func glIsInCube1( const p, d : TVectorGL) :Bool;
15096: //Func MinFloat( values : PSingleArray; nbItems : Int) : Single;
15097: //Func MinFloat1( values : PDoubleArray; nbItems : Int) : Double;
15098: //Func MinFloat2( values : PExtendedArray; nbItems : Int) : Extended;
15099: Func glMinFloat3( const v1, v2 : Single) : Single;
15100: Func glMinFloat4( const v : array of Single) : Single;
15101: Func glMinFloat5( const v1, v2 : Double) : Double;
15102: Func glMinFloat6( const v1, v2 : Extended) : Extended;
15103: Func glMinFloat7( const v1, v2, v3 : Single) : Single;
15104: Func glMinFloat8( const v1, v2, v3 : Double) : Double;
15105: Func glMinFloat9( const v1, v2, v3 : Extended) : Extended;

```



```

15106: //Func MaxFloat10( values : PSingleArray; nbItems : Int) : Single;
15107: //Func MaxFloat( values : PDoubleArray; nbItems : Int) : Double;
15108: //Func MaxFloat1( values : PExtendedArray; nbItems : Int) : Extended;
15109: Func glMaxFloat2( const v : array of Single) : Single;
15110: Func glMaxFloat3( const v1, v2 : Single) : Single;
15111: Func glMaxFloat4( const v1, v2 : Double) : Double;
15112: Func glMaxFloat5( const v1, v2 : Extended) : Extended;
15113: Func glMaxFloat6( const v1, v2, v3 : Single) : Single;
15114: Func glMaxFloat7( const v1, v2, v3 : Double) : Double;
15115: Func glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
15116: Func glMinInt9( const v1, v2 : Int) : Int;
15117: Func glMinInt( const v1, v2 : Card) : Card;
15118: Func glMaxInt( const v1, v2 : Int) : Int;
15119: Func glMaxInt1( const v1, v2 : Card) : Card;
15120: Func glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
15121: //Func PolygonArea( const p : PAffineVectorArray; nSides : Int) : Single;
15122: Func glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
15123: //Func PolygonSignedArea( const p : PAffineVectorArray; nSides : Int) : Single;
15124: //Proc ScaleFloatArray( values : PSingleArray; nb : Int; var factor : Single);
15125: Proc glScaleFloatArray( var values : TSingleArray; factor : Single);
15126: //Proc OffsetFloatArray( values : PSingleArray; nb : Int; var delta : Single);
15127: Proc glOffsetFloatArray1( var values : array of Single; delta : Single);
15128: //Proc OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Int);
15129: Func glMaxXYZComponent( const v : TVectorGL) : Single;
15130: Func glMaxXYZComponent1( const v : TAffineVector) : single;
15131: Func glMinXYZComponent( const v : TVectorGL) : Single;
15132: Func glMinXYZComponent1( const v : TAffineVector) : single;
15133: Func glMaxAbsXYZComponent( v : TVectorGL) : Single
15134: Func glMinAbsXYZComponent( v : TVectorGL) : Single
15135: Proc glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
15136: Proc glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
15137: Proc glMinVector( var v : TVectorGL; const v1 : TVectorGL);
15138: Proc glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
15139: Proc glSortArrayAscending( var a : array of Extended)
15140: Func glClampValue( const aValue, aMin, aMax : Single) : Single;
15141: Func glClampValue1( const aValue, aMin : Single) : Single;
15142: Func glGeometryOptimizationMode :Str
15143: Proc glBeginFPUOnlySection
15144: Proc glEndFPUOnlySection
15145: Func glConvertRotation( const Angles : TAffineVector) : TVectorGL
15146: Func glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
15147: Func glMakeDblVector( var v : array of Double) : THomogeneousDblVector
15148: Func glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
15149: Func glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
15150: Func glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
15151: Func glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
15152: Func glPointInPolygon( var xp, yp : array of Single; x, y : Single) :Bool
15153: Proc glDivMod( Dividend : Int; Divisor : Word; var Result, Remainder : Word)
15154: Func glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
15155: Func glTurn1( const Matrix: TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
15156: Func glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
15157: Func glPitch1( const Matrix: TMatrixGL; const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
15158: Func glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
15159: Func glRoll1( const Matrix: TMatrixGL; const MasterDirection:TAffineVector;Angle:Single): TMatrixGL;
15160: Func glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL; const point: TVectorGL) : Single
15161: Func glRayCastIntersectsSphere( const rayStart, rayVector: TVectorGL; const sphereCenter: TVectorGL; const
sphereRadius: Single) : Bool;
15162: Func glRayCastSphereIntersect( const rayStart, rayVector: TVectorGL; const sphereCenter: TVectorGL; const
sphereRadius: Single; var i1, i2: TVectorGL) : Int;
15163: Func glSphereVisibleRadius( distance, radius : Single) : Single
15164: Func glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
15165: Func glIsVolumeClipped( const objPos: TVectorGL; const objRadius: Single; const
rccl: TRenderContextClippingInfo) : Bool;
15166: Func glIsVolumeClipped1( const objPos: TAffineVector; const objRadius: Single; const
rccl: TRenderContextClippingInfo) : Bool;
15167: Func glIsVolumeClipped2( const min, max: TAffineVector; const rccl: TRenderContextClippingInfo) : Bool;
15168: Func glIsVolumeClipped3( const objPos: TAffineVector; const objRadius: Single; const Frustum: TFrustum) : Bool;
15169: Func glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
15170: Func glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
15171: Func glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
15172: Func glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
15173: Func glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
15174: 'cPI', 'Single').setExtended( 3.141592654);
15175: 'cPIdiv180', 'Single').setExtended( 0.017453292);
15176: 'c180divPI', 'Single').setExtended( 57.29577951);
15177: 'c2PI', 'Single').setExtended( 6.283185307);
15178: 'cPIdiv2', 'Single').setExtended( 1.570796326);
15179: 'cPIdiv4', 'Single').setExtended( 0.785398163);
15180: 'c3PIdiv4', 'Single').setExtended( 2.35619449);
15181: 'cInv2PI', 'Single').setExtended( 1 / 6.283185307);
15182: 'cInv360', 'Single').setExtended( 1 / 360);
15183: 'c180', 'Single').setExtended( 180);
15184: 'c360', 'Single').setExtended( 360);
15185: 'cOneHalf', 'Single').setExtended( 0.5);
15186: 'cLn10', 'Single').setExtended( 2.302585093);
15187: {'MinSingle', 'Extended').setExtended( 1.5e-45);
15188: 'MaxSingle', 'Extended').setExtended( 3.4e+38);
15189: 'MinDouble', 'Extended').setExtended( 5.0e-324);
15190: 'MaxDouble', 'Extended').setExtended( 1.7e+308);

```

```

15191: 'MinExtended','Extended').setExtended( 3.4e-4932);
15192: 'MaxExtended','Extended').setExtended( 1.1e+4932);
15193: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
15194: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
15195: end;
15196:
15197: Proc SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
15198: begin
15199:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
15200:   (FindClass('TOBJECT'),'TFaceGroups
15201:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
15202:   TMeshAutoCenterings', 'set of TMeshAutoCentering
15203:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
15204:   SIRegister_TBaseMeshObject(CL);
15205:   (FindClass('TOBJECT'),'TSkeletonFrameList
15206:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
15207:   SIRegister_TSkeletonFrame(CL);
15208:   SIRegister_TSkeletonFrameList(CL);
15209:   (FindClass('TOBJECT'),'TSkeleton
15210:   (FindClass('TOBJECT'),'TSkeletonBone
15211:   SIRegister_TSkeletonBoneList(CL);
15212:   SIRegister_TSkeletonRootBoneList(CL);
15213:   SIRegister_TSkeletonBone(CL);
15214:   (FindClass('TOBJECT'),'TSkeletonColliderList
15215:   SIRegister_TSkeletonCollider(CL);
15216:   SIRegister_TSkeletonColliderList(CL);
15217:   (FindClass('TOBJECT'),'TGLBaseMesh
15218:   TBlendedLerpInfo', 'record frameIndex1 : Int; frameIndex2 : '
15219:   + 'Int; lerpFactor : Single; weight : Single; externalPositions : TAffine'
15220:   + 'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
15221:   + 'QuaternionList; end
15222:   SIRegister_TSkeleton(CL);
15223:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
15224:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
15225:   SIRegister_TMeshObject(CL);
15226:   SIRegister_TMeshObjectList(CL);
15227:   //TMeshObjectListClass', 'class of TMeshObjectList
15228:   (FindClass('TOBJECT'),'TMeshMorphTargetList
15229:   SIRegister_TMeshMorphTarget(CL);
15230:   SIRegister_TMeshMorphTargetList(CL);
15231:   SIRegister_TMorphableMeshObject(CL);
15232:   TVertexBoneWeight', 'record BoneID : Int; Weight : Single; end
15233:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not work
15234:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
15235:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
15236:   SIRegister_TSkeletonMeshObject(CL);
15237:   SIRegister_TFaceGroup(CL);
15238:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl
15239:   + 'atTriangles, fgmmTriangleFan, fgmmQuads )
15240:   SIRegister_TFGVertexIndexList(CL);
15241:   SIRegister_TFGVertexNormalTexIndexList(CL);
15242:   SIRegister_TFGIndexTexCoordList(CL);
15243:   SIRegister_TFaceGroups(CL);
15244:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
15245:   SIRegister_TVectorFile(CL);
15246:   //TVectorFileClass', 'class of TVectorFile
15247:   SIRegister_TGLGLSMVectorFile(CL);
15248:   SIRegister_TGLBaseMesh(CL);
15249:   SIRegister_TGLFreeForm(CL);
15250:   TGLActorOption', '( aoSkeletonNormalizeNormals )
15251:   TGLActorOptions', 'set of TGLActorOption
15252:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
15253:   (FindClass('TOBJECT'),'TGLActor
15254:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
15255:   SIRegister_TActorAnimation(CL);
15256:   TActorAnimationName', 'String
15257:   SIRegister_TActorAnimations(CL);
15258:   SIRegister_TGLBaseAnimationController(CL);
15259:   SIRegister_TGLAnimationController(CL);
15260:   TActorFrameInterpolation', '( afpNone, afpLinear )
15261:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc
15262:   + 'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
15263:   SIRegister_TGLActor(CL);
15264:   SIRegister_TVectorFileFormat(CL);
15265:   SIRegister_TVectorFileFormatsList(CL);
15266:   (FindClass('TOBJECT'),'EInvalidVectorFile
15267:   Func GetVectorFileFormats : TVectorFileFormatsList
15268:   Func VectorFileFormatsFilter : Str
15269:   Func VectorFileFormatsSaveFilter : Str
15270:   Func VectorFileFormatExtensionByIndex( index : Int) : Str
15271:   Proc RegisterVectorFileFormat(const aExtension,aDescription:str;aClass: TVectorFileClass)
15272:   Proc UnregisterVectorFileClass( aClass : TVectorFileClass)
15273: end;
15274:
15275: Proc SIRegister_AxCtrls(CL: TPSPascalCompiler);
15276: begin
15277:   'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
15278:   'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
15279:   'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}

```

```

15280: 'Class_DStringPropPage', 'TGUID ' {F42D677E-754B-11D0-BDFB-00A024D1875C}
15281: SIRegister_TOLEStream(CL);
15282: (FindClass('TOBJECT'), 'TConnectionPoints
15283: TConnectionKind', ' ( ckSingle, ckMulti )
15284: SIRegister_TConnectionPoint(CL);
15285: SIRegister_TConnectionPoints(CL);
15286: TDefinePropertyPage', 'Proc ( const GUID : TGUID)
15287: (FindClass('TOBJECT'), 'TActiveXControlFactory
15288: SIRegister_TActiveXControl(CL);
15289: //TActiveXControlClass', 'class of TActiveXControl
15290: SIRegister_TActiveXControlFactory(CL);
15291: SIRegister_TActiveFormControl(CL);
15292: SIRegister_TActiveForm(CL);
15293: //TActiveFormClass', 'class of TActiveForm
15294: SIRegister_TActiveFormFactory(CL);
15295: (FindClass('TOBJECT'), 'TPropertyPageImpl
15296: SIRegister_TPropertyPage(CL);
15297: //TPropertyPageClass', 'class of TPropertyPage
15298: SIRegister_TPropertyPageImpl(CL);
15299: SIRegister_TActiveXPropertyPage(CL);
15300: SIRegister_TActiveXPropertyPageFactory(CL);
15301: SIRegister_TCustomAdapter(CL);
15302: SIRegister_TAdapterNotifier(CL);
15303: SIRegister_IFontAccess(CL);
15304: SIRegister_TFontAdapter(CL);
15305: SIRegister_IPictureAccess(CL);
15306: SIRegister_TPictureAdapter(CL);
15307: SIRegister_TOleGraphic(CL);
15308: SIRegister_TStringsAdapter(CL);
15309: SIRegister_TReflectorWindow(CL);
15310: Proc EnumDispatchProperties (Dispatch:IDispatch; PropType:TGUID; VTCode: Int; PropList:TStrings);
15311: Proc GetOleFont( Font : TFont; var OleFont : IFontDisp)
15312: Proc SetOleFont( Font : TFont; OleFont : IFontDisp)
15313: Proc GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
15314: Proc SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
15315: Proc GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
15316: Proc SetOleStrings( Strings : TStrings; OleStrings : IStrings)
15317: Func ParkingWindow : HWND
15318: end;
15319:
15320: Proc SIRegister_synaip(CL: TPSPascalCompiler);
15321: begin
15322: // Tip6Bytes = array [0..15] of Byte;
15323: {binary form of IPv6 address (for string conversion routines)}
15324: // Tip6Words = array [0..7] of Word;
15325: AddTypeS('Tip6Bytes', 'array [0..15] of Byte;;
15326: AddTypeS('Tip6Words', 'array [0..7] of Word;;
15327: AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end;
15328: Func synaIsIP6( const Value :Str) :Bool;
15329: Func synaIsIP6( const Value :Str) :Bool;
15330: Func synaIPToID( Host :Str) : Ansistr;
15331: Func synaStrToIP6( value :Str) : Tip6Bytes;
15332: Func synaIP6ToStr( value : Tip6Bytes) :Str;
15333: Func synaStrToIP( value :Str) : Int;
15334: Func synaIPToStr( value : Int) :Str;
15335: Func synaReverseIP( Value : Ansistr) : Ansistr;
15336: Func synaReverseIP6( Value : Ansistr) : Ansistr;
15337: Func synaExpandIP6( Value : Ansistr) : Ansistr;
15338: Func xStrToIP( const Value :Str) : TIPAdr;
15339: Func xIPToStr( const Adresse : TIPAdr) :Str;
15340: Func IPToCardinal( const Adresse : TIPAdr) :Card;
15341: Func CardinalToIP( const Value :Card) : TIPAdr;
15342: Func ReverseIP( Value : Ansistr) : Ansistr';
15343: Func ReverseIP6( Value : Ansistr) : Ansistr';
15344: end;
15345:
15346: Proc SIRegister_synacode(CL: TPSPascalCompiler);
15347: begin
15348: AddTypeS('TSpecials', 'set of Char;
15349: Const('SpecialChar', 'TSpecials').SetSet( '='[<>:;,&/?\"_];
15350: Const('URLEFullSpecialChar', 'TSpecials').SetSet( ' ;/?:@=&#+;
15351: Const('TableBase64'ABCDEFHGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=;
15352: Const('TableBase64mod'ABCDEFHGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,,=;
15353: Const('TableUU'('!'#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_
15354: Const('TableXX'(+0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;
15355: Func DecodeTriplet( const Value : Ansistr; Delimiter : Char) : Ansistr;
15356: Func DecodeQuotedPrintable( const Value : Ansistr) : Ansistr;
15357: Func DecodeURL( const Value : Ansistr) : Ansistr;
15358: Func EncodeTriplet( const Value:Ansistr;Delimiter:Char;Specials:TSpecials):Ansistr);
15359: Func EncodeQuotedPrintable( const Value : Ansistr) : Ansistr;
15360: Func EncodeSafeQuotedPrintable( const Value : Ansistr) : Ansistr;
15361: Func EncodeURLElement( const Value : Ansistr) : Ansistr;
15362: Func EncodeURL( const Value : Ansistr) : Ansistr;
15363: Func Decode4to3( const Value, Table : Ansistr) : Ansistr;
15364: Func Decode4to3Ex( const Value, Table : Ansistr) : Ansistr;
15365: Func Encode3to4( const Value, Table : Ansistr) : Ansistr;
15366: Func synDecodeBase64( const Value : Ansistr) : Ansistr;
15367: Func synEncodeBase64( const Value : Ansistr) : Ansistr;
15368: Func DecodeBase64mod( const Value : Ansistr) : Ansistr;

```

```

15369: Func EncodeBase64mod( const Value : Ansistr ) : Ansistr;
15370: Func DecodeUU( const Value : Ansistr ) : Ansistr;
15371: Func EncodeUU( const Value : Ansistr ) : Ansistr;
15372: Func DecodeXX( const Value : Ansistr ) : Ansistr;
15373: Func DecodeYEnc( const Value : Ansistr ) : Ansistr;
15374: Func UpdateCrc32( Value : Byte; Crc32 : Int ) : Int;
15375: Func synCrc32( const Value : Ansistr ) : Int;
15376: Func UpdateCrc16( Value : Byte; Crc16 : Word ) : Word;
15377: Func Crc16( const Value : Ansistr ) : Word;
15378: Func synMD5( const Value : Ansistr ) : Ansistr;
15379: Func HMAC_MD5( Text, Key : Ansistr ) : Ansistr;
15380: Func MD5LongHash( const Value : Ansistr; Len : Int ) : Ansistr;
15381: Func synSHA1( const Value : Ansistr ) : Ansistr;
15382: Func HMAC_SHA1( Text, Key : Ansistr ) : Ansistr;
15383: Func SHA1LongHash( const Value : Ansistr; Len : Int ) : Ansistr;
15384: Func synMD4( const Value : Ansistr ) : Ansistr;
15385: end;
15386:
15387: Proc SIRegister_synachar(CL: TPSPascalCompiler);
15388: begin
15389:   AddTypes('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
15390:     + 'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
15391:     + 'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
15392:     + '4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
15393:     + '8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE, '
15394:     + 'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
15395:     + ', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
15396:     + ', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
15397:     + 'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
15398:     + '1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
15399:     + '2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN,'
15400:     + 'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
15401:     + 'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
15402:     + ', CP864, CP865, CP869, CP1125 );
15403:   AddTypes('TMimeSetChar', 'set of TMimeChar;
15404:   Func CharSetConversion(const Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiStr;
15405:   Func CharSetConversionEx(const Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeChar;const
TransformTable:array of Word):AnsiStr;
15406:   Func CharSetConversionTrans(Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeChar;const TransformTable:array
of Word;Translit:Boolean):AnsiStr;
15407:   Func GetCurCP : TMimeChar;
15408:   Func GetCurOEMCP : TMimeChar;
15409:   Func GetCPFromID( Value : Ansistr ) : TMimeChar;
15410:   Func GetIDFromCP( Value : TMimeChar ) : Ansistr;
15411:   Func NeedCharSetConversion( const Value : Ansistr ) : Bool;
15412:   Func IdealCharSetCoding(const Value:AnsiStr;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
15413:   Func GetBOM( Value : TMimeChar ) : Ansistr;
15414:   Func StringToWide( const Value : Ansistr ) : WideString;
15415:   Func WideToString( const Value : WideString ) : Ansistr;
15416: end;
15417:
15418: Proc SIRegister_synamisc(CL: TPSPascalCompiler);
15419: begin
15420:   AddTypes('TProxySetting', 'record Host :Str; Port :Str; Bypass :Str; end;
15421:   Proc WakeOnLan( MAC, IP :Str);
15422:   Func GetDNS :Str;
15423:   Func GetIEProxy( protocol :Str ) : TProxySetting;
15424:   Func GetLocalIPs :Str;
15425: end;
15426:
15427: Proc SIRegister_synaser(CL: TPSPascalCompiler);
15428: begin
15429:   AddConstantN('synCR','Char #$0d);
15430:   Const('synLF','Char #$0a);
15431:   Const('cSerialChunk','LongInt'( 8192);
15432:   Const('LockfileDirectory','String'('/var/lock);
15433:   Const('PortIsClosed','LongInt'( - 1);
15434:   Const('ErrAlreadyOwned','LongInt'( 9991);
15435:   Const('ErrAlreadyInUse','LongInt'( 9992);
15436:   Const('ErrWrongParameter','LongInt'( 9993);
15437:   Const('ErrPortNotOpen','LongInt'( 9994);
15438:   Const('ErrNoDeviceAnswer','LongInt'( 9995);
15439:   Const('ErrMaxBuffer','LongInt'( 9996);
15440:   Const('ErrTimeout','LongInt'( 9997);
15441:   Const('ErrNotRead','LongInt'( 9998);
15442:   Const('ErrFrame','LongInt'( 9999);
15443:   Const('ErrOverrun','LongInt'( 10000);
15444:   Const('ErrRxOver','LongInt'( 10001);
15445:   Const('ErrRxParity','LongInt'( 10002);
15446:   Const('ErrTxFull','LongInt'( 10003);
15447:   Const('dcb_Binary','LongWord')( $00000001);
15448:   Const('dcb_ParityCheck','LongWord')( $00000002);
15449:   Const('dcb_OutxCtsFlow','LongWord')( $00000004);
15450:   Const('dcb_OutxDsrFlow','LongWord')( $00000008);
15451:   Const('dcb_DtrControlMask','LongWord')( $00000030);
15452:   Const('dcb_DtrControlDisable','LongWord')( $00000000);
15453:   Const('dcb_DtrControlEnable','LongWord')( $00000010);
15454:   Const('dcb_DtrControlHandshake','LongWord')( $00000020);
15455:   Const('dcb_DsrSensitivity','LongWord')( $00000040);

```

```

15456: Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
15457: Const('dcb_OutX','LongWord')( $00000100);
15458: Const('dcb_InX','LongWord')( $00000200);
15459: Const('dcb_ErrorChar','LongWord')( $00000400);
15460: Const('dcb_NullStrip','LongWord')( $00000800);
15461: Const('dcb_RtsControlMask','LongWord')( $00003000);
15462: Const('dcb_RtsControlDisable','LongWord')( $00000000);
15463: Const('dcb_RtsControlEnable','LongWord')( $00001000);
15464: Const('dcb_RtsControlHandshake','LongWord')( $00002000);
15465: Const('dcb_RtsControlToggle','LongWord')( $00003000);
15466: Const('dcb_AbortOnError','LongWord')( $00004000);
15467: Const('dcb_Reserveds','LongWord')( $FFFF8000);
15468: Const('synSB1','LongInt'( 0);
15469: Const('SBlandHalf','LongInt'( 1);
15470: Const('synSB2','LongInt'( 2);
15471: Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
15472: Const('CS7fix','LongWord')( $0000020);
15473: AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
15474:   + 'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
15475:   + 'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
15476:   + 'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end;
15477: //AddTypeS('PDCB', '^TDCB // will not work;
15478: //Const('MaxRates','LongInt'( 18);
15479: //Const('MaxRates','LongInt'( 30);
15480: //Const('MaxRates','LongInt'( 19);
15481: Const('O_SYNC','LongWord')( $0080);
15482: Const('synOK','LongInt'( 0);
15483: Const('synErr','LongInt'( Int ( - 1 ));
15484: AddTypeS('THookSerialReason', '(HR_SerialClose,HR_Connect,HR_CanRead,HR_CanWrite,HR_ReadCount,
HR_WriteCount,HR_Wait);
15485: Type('THookSerialStatus', Procedure (Sender:TObject;Reason:THookSerialReason;const Value:str);
15486: SIRegister_ESynaSerError(CL);
15487: SIRegister_TBlockSerial(CL);
15488: Func GetSerialPortNames :Str;
15489: end;
15490:
15491: Proc SIRegister_synaicnv(CL: TPSPascalCompiler);
15492: begin
15493:   Const('DLLIconvName','String 'libiconv.so;
15494:   Const('DLLIconvName','String 'iconv.dll;
15495:   AddTypeS('size_t', 'Cardinal;
15496:   AddTypeS('iconv_t', 'Int;
15497:   //AddTypeS('iconv_t', 'Pointer;
15498:   AddTypeS('argptr', 'iconv_t;
15499:   Func SynaIconvOpen( const tocode, fromcode : Ansistr) : iconv_t;
15500:   Func SynaIconvOpenTranslit( const tocode, fromcode : Ansistr) : iconv_t;
15501:   Func SynaIconvOpenIgnore( const tocode, fromcode : Ansistr) : iconv_t;
15502:   Func SynaIconv( cd : iconv_t; inbuf : Ansistr; var outbuf : Ansistr) : Int;
15503:   Func SynaIconvClose( var cd : iconv_t) : Int;
15504:   Func SynaIconvCtl( cd : iconv_t; request : Int; argument : argptr) : Int;
15505:   Func IsIconvloaded :Bool;
15506:   Func InitIconvInterface :Bool;
15507:   Func DestroyIconvInterface :Bool;
15508:   Const('ICONV_TRIVIALP','LongInt'( 0);
15509:   Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
15510:   Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
15511:   Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
15512:   Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
15513: end;
15514:
15515: Proc SIRegister_pingsend(CL: TPSPascalCompiler);
15516: begin
15517:   Const('ICMP_ECHO','LongInt'( 8);
15518:   Const('ICMP_ECHOREPLY','LongInt'( 0);
15519:   Const('ICMP_UNREACH','LongInt'( 3);
15520:   Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
15521:   Const('ICMP6_ECHO','LongInt'( 128);
15522:   Const('ICMP6_ECHOREPLY','LongInt'( 129);
15523:   Const('ICMP6_UNREACH','LongInt'( 1);
15524:   Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
15525:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
15526:   + 'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort );
15527:   SIRegister_TPINGSend(CL);
15528:   Func PingHost( const Host :Str) : Int;
15529:   Func TraceRouteHost( const Host :Str) :Str;
15530: end;
15531:
15532: Proc SIRegister_asnutil(CL: TPSPascalCompiler);
15533: begin
15534:   AddConstantN('synASN1_BOOL','LongWord')( $01);
15535:   Const('synASN1_INT','LongWord')( $02);
15536:   Const('synASN1_OCTSTR','LongWord')( $04);
15537:   Const('synASN1_NULL','LongWord')( $05);
15538:   Const('synASN1_OBJID','LongWord')( $06);
15539:   Const('synASN1_ENUM','LongWord')( $0a);
15540:   Const('synASN1_SEQ','LongWord')( $30);
15541:   Const('synASN1_SETOF','LongWord')( $31);
15542:   Const('synASN1_IPADDR','LongWord')( $40);
15543:   Const('synASN1_COUNTER','LongWord')( $41);

```



```

15544: Const('synASN1_GAUGE','LongWord')($42);
15545: Const('synASN1_TIMETICKS','LongWord')($43);
15546: Const('synASN1_OPAQUE','LongWord')($44);
15547: Func synASNEncOIDItem( Value : Int) : Ansistr;
15548: Func synASNDecOIDItem( var Start : Int; const Buffer : Ansistr) : Int;
15549: Func synASNEncLen( Len : Int) : Ansistr;
15550: Func synASNDecLen( var Start : Int; const Buffer : Ansistr) : Int;
15551: Func synASNEncInt( Value : Int) : Ansistr;
15552: Func synASNEncUInt( Value : Int) : Ansistr;
15553: Func synASNObject( const Data : Ansistr; ASNType : Int) : Ansistr;
15554: Func synASNItem( var Start: Int; const Buffer: Ansistr; var Value: Int): Ansistr;
15555: Func synMibToId( Mib : Str) : Ansistr;
15556: Func synIdToMib( const Id : Ansistr) : Str;
15557: Func synIntMibToStr( const Value : Ansistr) : Ansistr;
15558: Func ASNDump( const Value : Ansistr) : Ansistr;
15559: Func GetMailServers( const DNSHost, Domain : Ansistr; const Servers: TStrings): Bool;
15560: Func LDAPResultDump( const Value : TLDAPEResultList) : Ansistr;
15561: end;
15562:
15563: Proc SIRegister_ldapsend(CL: TPSPascalCompiler);
15564: begin
15565: Const('cLDAPProtocol','String '389;
15566: Const('LDAP_ASN1_BIND_REQUEST','LongWord')($60);
15567: Const('LDAP_ASN1_BIND_RESPONSE','LongWord')($61);
15568: Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')($42);
15569: Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')($63);
15570: Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')($64);
15571: Const('LDAP_ASN1_SEARCH_DONE','LongWord')($65);
15572: Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')($73);
15573: Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')($66);
15574: Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')($67);
15575: Const('LDAP_ASN1_ADD_REQUEST','LongWord')($68);
15576: Const('LDAP_ASN1_ADD_RESPONSE','LongWord')($69);
15577: Const('LDAP_ASN1_DEL_REQUEST','LongWord')($4A);
15578: Const('LDAP_ASN1_DEL_RESPONSE','LongWord')($6B);
15579: Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')($6C);
15580: Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')($6D);
15581: Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')($6E);
15582: Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')($6F);
15583: Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')($70);
15584: Const('LDAP_ASN1_EXT_REQUEST','LongWord')($77);
15585: Const('LDAP_ASN1_EXT_RESPONSE','LongWord')($78);
15586: SIRegister_TLDAPAttribute(CL);
15587: SIRegister_TLDAPAttributeList(CL);
15588: SIRegister_TLDAPResult(CL);
15589: SIRegister_TLDAPResultList(CL);
15590: AddTypeS('TLDAPEModifyOp', '( MO_Add, MO_Delete, MO_Replace );
15591: AddTypeS('TLDAPESearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree );
15592: AddTypeS('TLDAPESearchAliases', (SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always);
15593: SIRegister_TLDAPSend(CL);
15594: Func LDAPResultDump( const Value : TLDAPEResultList) : Ansistr;
15595: end;
15596:
15597:
15598: Proc SIRegister_slogsend(CL: TPSPascalCompiler);
15599: begin
15600: Const('cSysLogProtocol','String '514;
15601: Const('FCL_Kernel','LongInt'( 0);
15602: Const('FCL_UserLevel','LongInt'( 1);
15603: Const('FCL_MailSystem','LongInt'( 2);
15604: Const('FCL_System','LongInt'( 3);
15605: Const('FCL_Security','LongInt'( 4);
15606: Const('FCL_Syslogd','LongInt'( 5);
15607: Const('FCL_Printer','LongInt'( 6);
15608: Const('FCL_News','LongInt'( 7);
15609: Const('FCL_UUCP','LongInt'( 8);
15610: Const('FCL_Clock','LongInt'( 9);
15611: Const('FCL_Authorization','LongInt'( 10);
15612: Const('FCL_FTP','LongInt'( 11);
15613: Const('FCL_NTP','LongInt'( 12);
15614: Const('FCL_LogAudit','LongInt'( 13);
15615: Const('FCL_LogAlert','LongInt'( 14);
15616: Const('FCL_Time','LongInt'( 15);
15617: Const('FCL_Local0','LongInt'( 16);
15618: Const('FCL_Local1','LongInt'( 17);
15619: Const('FCL_Local2','LongInt'( 18);
15620: Const('FCL_Local3','LongInt'( 19);
15621: Const('FCL_Local4','LongInt'( 20);
15622: Const('FCL_Local5','LongInt'( 21);
15623: Const('FCL_Local6','LongInt'( 22);
15624: Const('FCL_Local7','LongInt'( 23);
15625: Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning, Notice, Info, Debug);
15626: SIRegister_TSyslogMessage(CL);
15627: SIRegister_TSyslogSend(CL);
15628: Func ToSysLog(const SyslogServer: str; Facil: Byte; Sever: TSyslogSeverity; const Content: str): Bool;
15629: end;
15630:
15631: Proc SIRegister_mimemess(CL: TPSPascalCompiler);
15632: begin

```

```

15633:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high );
15634:   SIRegister_TMessHeader(CL);
15635:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader;
15636:   SIRegister_TMimeMess(CL);
15637: end;
15638:
15639: Proc SIRegister_mimepart(CL: TPSPascalCompiler);
15640: begin
15641:   (FindClass('TOBJECT'), 'TmimeType');
15642:   AddTypeS('THookWalkPart', 'Proc ( const Sender : TmimeType);
15643:   AddTypeS('TmimeTypePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY );
15644:   AddTypeS('TmimeTypeEncoding', ( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX );
15645:   SIRegister_TmimeType(CL);
15646:   Const('MaxMimeType', 'LongInt' ( 25);
15647:   Func GenerateBoundary :Str;
15648: end;
15649:
15650: Proc SIRegister_mimeinln(CL: TPSPascalCompiler);
15651: begin
15652:   Func InlineDecode( const Value :Str; CP : TmimeTypeChar) :Str;
15653:   Func InlineEncode( const Value :Str; CP, MimeType : TmimeTypeChar) :Str;
15654:   Func NeedInline( const Value : Ansistr) :Bool;
15655:   Func InlineCodeEx( const Value :Str; FromCP : TmimeTypeChar) :Str;
15656:   Func InlineCode( const Value :Str) :Str;
15657:   Func InlineEmailEx( const Value :Str; FromCP : TmimeTypeChar) :Str;
15658:   Func InlineEmail( const Value :Str) :Str;
15659: end;
15660:
15661: Proc SIRegister_ftpsend(CL: TPSPascalCompiler);
15662: begin
15663:   Const('cFtpProtocol', 'String '21;
15664:   Const('cFtpDataProtocol', 'String '20;
15665:   Const('FTP_OK', 'LongInt' ( 255);
15666:   Const('FTP_ERR', 'LongInt' ( 254);
15667:   AddTypeS('TFTPStatus', 'Procedure(Sender:TObject; Response:Boolean; const Value :Str);
15668:   SIRegister_TFTPListRec(CL);
15669:   SIRegister_TFTPList(CL);
15670:   SIRegister_TFTPListSend(CL);
15671:   Func FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass :Str) :Bool;
15672:   Func FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass :Str) :Bool;
15673:   Func FtpInterServerTransfer(const FromIP, FromPort, FromFile, FromUser, FromPass:str; const ToIP, ToPort, ToFile,
   ToUser, ToPass:str):Bool;
15674: end;
15675:
15676: Proc SIRegister_httpsend(CL: TPSPascalCompiler);
15677: begin
15678:   Const('cHttpProtocol', 'String '80;
15679:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED );
15680:   SIRegister_THTTPEnd(CL);
15681:   Func HttpGetText( const URL :Str; const Response : TStrings) :Bool;
15682:   Func HttpGetBinary( const URL :Str; const Response : TStream) :Bool;
15683:   Func HttpPostBinary( const URL :Str; const Data : TStream) :Bool;
15684:   Func HttpPostURL( const URL, URLData :Str; const Data : TStream) :Bool;
15685:   Func HttpPostFile(const URL, FileName, FileName:str; const Data:TStream; const ResultData:TStrings):Bool;
15686: end;
15687:
15688: Proc SIRegister_smtpsend(CL: TPSPascalCompiler);
15689: begin
15690:   Const('cSmtpProtocol', 'String '25;
15691:   SIRegister_TSMTPSend(CL);
15692:   Func SendToRaw(const MailFr, MailTo, SMTPHost:str; const MailData:TStrings; const Username, Passw:str):Bool;
15693:   Func SendTo(const MailFrom, MailTo, Subject, SMTPHost:str; const MailData:TStrings):Bool;
15694:   Func SendToEx(const MailFrom, MailTo, Subject, SMTPHost:str; const MailData:TStrings; const Username,
   Password :Str):Boolean;
15695: end;
15696:
15697: Proc SIRegister_snmpsend(CL: TPSPascalCompiler);
15698: begin
15699:   Const('cSnmpProtocol', 'String '161;
15700:   Const('cSnmpTrapProtocol', 'String '162;
15701:   Const('SNMP_V1', 'LongInt' ( 0);
15702:   Const('SNMP_V2C', 'LongInt' ( 1);
15703:   Const('SNMP_V3', 'LongInt' ( 3);
15704:   Const('PDUGetRequest', 'LongWord' ( $A0);
15705:   Const('PDUGetNextRequest', 'LongWord' ( $A1);
15706:   Const('PDUGetResponse', 'LongWord' ( $A2);
15707:   Const('PDUSetRequest', 'LongWord' ( $A3);
15708:   Const('PDUTrap', 'LongWord' ( $A4);
15709:   Const('PDUGetBulkRequest', 'LongWord' ( $A5);
15710:   Const('PDUInformRequest', 'LongWord' ( $A6);
15711:   Const('PDUTrapV2', 'LongWord' ( $A7);
15712:   Const('PDUReport', 'LongWord' ( $A8);
15713:   Const('ENoError', LongInt 0;
15714:   Const('ETooBig', 'LongInt' ( 1);
15715:   Const('ENoSuchName', 'LongInt' ( 2);
15716:   Const('EBadValue', 'LongInt' ( 3);
15717:   Const('EReadOnly', 'LongInt' ( 4);
15718:   Const('EGenErr', 'LongInt' ( 5);
15719:   Const('ENoAccess', 'LongInt' ( 6);

```

```

15720: Const('EWrongType','LongInt'( 7);
15721: Const('EWrongLength','LongInt'( 8);
15722: Const('EWrongEncoding','LongInt'( 9);
15723: Const('EWrongValue','LongInt'( 10);
15724: Const('ENoCreation','LongInt'( 11);
15725: Const('EInconsistentValue','LongInt'( 12);
15726: Const('EResourceUnavailable','LongInt'( 13);
15727: Const('ECommitFailed','LongInt'( 14);
15728: Const('EUndoFailed','LongInt'( 15);
15729: Const('EAuthorizationError','LongInt'( 16);
15730: Const('ENotWritable','LongInt'( 17);
15731: Const('EInconsistentName','LongInt'( 18);
15732: AddTypeS('TV3Flags','( NoAuthNoPriv, AuthNoPriv, AuthPriv );
15733: AddTypeS('TV3Auth','( AuthMD5, AuthSHAL );
15734: AddTypeS('TV3Priv','( PrivDES, Priv3DES, PrivAES );
15735: SIRegister_TSNMPMib(CL);
15736: AddTypeS(TV3Sync',' record EngineID : Ansistr; EngineBoots : Int; EngineTime : Int; EngineStamp :Card;
end;
15737: SIRegister_TSNMPRec(CL);
15738: SIRegister_TSNMPSend(CL);
15739: Func SNMPGet(const OID,Community,SNMPHost:Ansistr; var Value: Ansistr) :Bool;
15740: Func SNMPSet(const OID,Community,SNMPHost,Value:Ansistr;ValueType:Int):Bool;
15741: Func SNMPGetNext(var OID:Ansistr;const Community,SNMPHost:Ansistr;var Value:Ansistr):Bool;
15742: Func SNMPGetTable(const BaseOID,Community,SNMPHost:Ansistr;const Value:TStrings):Bool;
15743: Func SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15744: Func SendTrap(const Dest,Source,Enterprise,Community : Ansistr; Generic,Specific,Seconds : Int; const
MIBName,MIBValue : Ansistr; MIBtype : Int) : Int;
15745: Func RecvTrap(var Dest,Source,Enterprise,Community:Ansistr;var Generic,Specific,Seconds Int; const
MIBName,MIBValue : TStringList): Int;
15746: end;
15747:
15748: Proc SIRegister_NetWork(CL: TPSPascalCompiler);
15749: begin
15750: Func GetDomainName2: Ansistr;
15751: Func GetDomainController( Domain : Ansistr) : Ansistr;
15752: Func GetDomainUsers( Controller : Ansistr) : Ansistr;
15753: Func GetDomainGroups( Controller : Ansistr) : Ansistr;
15754: Func GetDateTime( Controller : Ansistr) : TDateTime;
15755: Func GetFullName2( Controller, UserID : Ansistr) : Ansistr;
15756: end;
15757:
15758: Proc SIRegister_wwSystem(CL: TPSPascalCompiler);
15759: begin
15760: AddTypeS('TwvDateOrder','( doMDY, doDMY, doYMD );
15761: TwvDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15762: Func wwStrToDate( const S :Str) :Bool;
15763: Func wwStrToTime( const S :Str) :Bool;
15764: Func wwStrToDateTime( const S :Str) :Bool;
15765: Func wwStrToTimeVal( const S :Str) : TDateTime;
15766: Func wwStrToDateVal( const S :Str) : TDateTime;
15767: Func wwStrToDateTimeVal( const S :Str) : TDateTime;
15768: Func wwStrToInt( const S :Str) :Bool;
15769: Func wwStrToFloat( const S :Str) :Bool;
15770: Func wwGetDateOrder( const DateFormat :Str) : TwvDateOrder;
15771: Func wwNextDay( Year, Month, Day : Word) : Int;
15772: Func wwPriorDay( Year, Month, Day : Word) : Int;
15773: Func wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) :Bool;
15774: Func wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) :Bool;
15775: Func wwGetDateTimeCursorPosition(SelStart:int;Text:str;TimeOnly:Bool):TwvDateTimeSelection;
15776: Func wwGetTimeCursorPosition( SelStart : Int; Text :Str) : TwvDateTimeSelection;
15777: Func wwScanDate( const S :Str; var Date : TDateTime) :Bool;
15778: Func wwScanDateEpoch(const S:Str; var Date : TDateTime; Epoch : Int):Bool;
15779: Proc wwSetDateTimeCursorPosition(dateCursor:TwvDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15780: Func wwStrToFloat2(const S:Str; var FloatValue:Extended;DisplayFormat:str):bool;
15781: end;
15782:
15783: unit uPSI_Themes;
15784: Func ThemeServices : TThemeServices;
15785: Func ThemeControl( AControl : TControl) :Bool;
15786: Func UnthemedDesigner( AControl : TControl) :Bool;
15787: Proc SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15788: begin
15789: Func GetBindingkeyAccessPoint(const Operator:Str; const key:Str):Str;
15790: end;
15791: Unit uPSC_menus;
15792: Func StripHotkey( const Text :Str) :Str;
15793: Func GetHotkey( const Text :Str) :Str;
15794: Func AnsiSameCaption( const Text1, Text2 :Str) :Bool;
15795: Func IsAltGRPressed :Bool;
15796:
15797: Proc SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15798: begin
15799: TCommandEvent,Procedure(Thread:TIdPeerThread; const Tag,CmdStr:str;var Handled:Bool);
15800: SIRegister_TIdIMAP4Server(CL);
15801: end;
15802:
15803: Proc SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15804: begin
15805: 'HASH_SIZE','LongInt'( 256);

```

```

15806: CL.FindClass('TOBJECT'),'EVariantSymbolTable;
15807: TypeS('TSymbolType','( stInt, stFloat, stDate, stString );
15808: //TypeS('PSymbol','^TSymbol // will not work;
15809: TypeS('TSymbol','record Name :Str; BlockLevel : Int; HashValue'
15810: +': Int; Value : Variant; end;
15811: //TypeS('PSymbolArray','^TSymbolArray // will not work;
15812: SIRegister_TVVariantSymbolTable(CL);
15813: end;
15814:
15815: Proc SIRegister_udf_glob(CL: TPSPascalCompiler);
15816: begin
15817:   SIRegister_TThreadLocalVariables(CL);
15818:   Func MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar;
15819:   //Func MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD;
15820:   Func ThreadLocals : TThreadLocalVariables;
15821:   Proc WriteDebug( sz :Str);
15822:   ConstantN('UDF_SUCCESS','LongInt'( 0);
15823:   'UDF_FAILURE','LongInt'( 1);
15824:   'cSignificantlyLarger','LongInt'( 1024 * 4);
15825:   TypeS('mTByteArray','array of byte;;
15826:   Func ChangeOEPFFromBytes(bFile:mTByteArray):Boolean;
15827:   Func ChangeOEPFFromFile(sFile:str; sDestFile:str):Boolean;
15828:   Proc CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget:Str);
15829:   Func IsNetworkConnected:Bool;
15830:   Func IsInternetConnected:Bool;
15831:   Func IsCOMConnected:Bool;
15832:   Func IsNetworkOn:Bool;
15833:   Func IsInternetOn:Bool;
15834:   Func IsCOMOn:Bool;
15835:   Func SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15836:   TmrProc, 'Proc TmrProc(hWnd: HWND; uMsg: Int; idEvent: Int; dwTime: Int);;
15837:   Func SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT;
15838:   Func KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL;
15839:   Func wIsWindowUnicode( hWnd : HWND) : BOOL;
15840:   Func wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL;
15841:   Func wIsWindowEnabled( hWnd : HWND) : BOOL;
15842:   Func GetMenu( hWnd : HWND) : HMENU;
15843:   Func SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL;
15844: end;
15845:
15846: Proc SIRegister_SockTransport(CL: TPSPascalCompiler);
15847: begin
15848:   SIRegister_IDataBlock(CL);
15849:   SIRegister_ISendDataBlock(CL);
15850:   SIRegister_ITransport(CL);
15851:   SIRegister_TDataBlock(CL);
15852:   //TypeS('PIntArray','^TIntArray // will not work;
15853:   //TypeS('PVariantArray','^TVariantArray // will not work;
15854:   TypeS('TVarFlag','( vfByRef, vfVariant );
15855:   TypeS('TVarFlags','set of TVarFlag;
15856:   SIRegister_TCustomDataBlockInterpreter(CL);
15857:   SIRegister_TSendDataBlock(CL);
15858:   'CallSig','LongWord')( $D800);
15859:   'ResultSig','LongWord')( $D400);
15860:   'asMask','LongWord')( $00FF);
15861:   ClassN(CL.FindClass('TOBJECT'),'EInterpreterError;
15862:   ClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError;
15863:   Proc CheckSignature( Sig : Int);
15864: end;
15865:
15866: Proc SIRegister_WinInet(CL: TPSPascalCompiler);
15867: begin
15868:   //TypeS('HINTERNET','__Pointer;
15869:   TypeS('HINTERNET1','THANDLE;
15870:   TypeS('HINTERNET','Int;
15871:   TypeS('HINTERNET2','__Pointer;
15872:   //TypeS('PHINTERNET','^HINTERNET // will not work;
15873:   //TypeS('LPHINTERNET','PHINTERNET;
15874:   TypeS('INTERNET_PORT','Word;
15875:   //TypeS('PINTERNET_PORT','^INTERNET_PORT // will not work;
15876:   //TypeS('LPINTERNET_PORT','PINTERNET_PORT;
15877:   Func InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15878:   'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15879:   'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15880:   Func InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
15881:   Func InternetCreateUrl(var lpUrlCompons:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
dwUrlLength:DWORD):BOOL;
15882:   Func InternetCloseHandle(hInet : HINTERNET) : BOOL;
15883:   Func
InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;lpszPassword:PChar;d
15884:   Func InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
;dwContext:DWORD):HINTERNET;
15885:   Func InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15887:   Func InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumOfBytesAvail:DWORD;dwFlags,
dwContext:DWORD):BOOL;
15888:   Func InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL;
15889:   Func
InternetDial(hwndParent:HWND;lpszConnect:PChar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;

```

```

15890: Func InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD;
15891: Func InternetGoOnline( lpszURL : pchar; hwnDParent : HWND; dwFlags : DWORD) : BOOL;
15892: Func InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL;
15893: Func InternetAutodialHangup( dwReserved : DWORD) : BOOL;
15894: Func InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL;
15895: Func InternetCanonicalizeUrl( lpszUrl:PChar;lpszBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15896: Func InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl: PChar; lpszBuffer: PChar; var lpdwBufferLength: Func
InternetCloseHandle( hInet:HINTERNET):BOOL;
15897: Func InternetConnect( hInet:HINTERNET; lpszServerName:PChar; nServerPort:INTERNET_PORT; lpszUsername:PChar
; lpszPassword:PChar; dwService:DWORD; dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15899: Func InternetQueryDataAvailable( hFile:HINTERNET; var lpdwNumOfBytesAvail:DWORD; dwFlgs,
dwContext:DWORD):BOOL;
15900: Func FtpFindFirstFile( hConnect:HINTERNET; lpszSearchFile:PChar; var lpFindFileData:TWin32FindData; dwFlags:
DWORD; dwContext:DWORD): HINTERNET;
15901: Func
WFTpGetFile( hConnect:HINTERNET; lpszRemoteFile:PChar; lpszNewFile:PChar; fFailIfExists:BOOL; dwFlagsAndAttributes:DWORD; dw
15902: Func WFTpPutFile( hConct:HINTERNET; lpszLocFile:PChar; lpszNewRemFile:PChar; dwFlags:DWORD; dwCotx:DWORD):BOOL;
15903: Func FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL;
15904: Func FtpRenameFile( hConnect:HINTERNET; lpszExisting:PChar; lpszNew:PChar):BOOL;
15905: Func FtpOpenFile( hConnect:HINTER; lpszFileName:PChar; dwAccess:DWORD; dwFlags:DWORD; dwContext:DWORD):HINTER;
15906: Func FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL;
15907: Func FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL;
15908: Func FtpSetCurrentDirectory( hConnect:HINTERNET; lpszDirectory : PChar) : BOOL;
15909: Func FtpGetCurrentDirectory( hConnect:HINTER; lpszCurrentDir:PChar; var lpdwCurrentDir:DWORD):BOOL;
15910: Func FtpCommand( hConnect:HINTER; fExpectResponse:BOOL; dwFlags:DWORD; lpszCommnd:PChar; dwContxt:DWORD):BOOL;
15911: Func IS_GOPHER_FILE( GopherType : DWORD) : BOOL;
15912: Func IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL;
15913: Func IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL;
15914: Func IS_GOPHER_ERROR( GopherType : DWORD) : BOOL;
15915: Func IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL;
15916: Func IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL;
15917: Func IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL;
15918: Func IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL;
15919: Func IS_GOPHER_ASK( GopherType : DWORD) : BOOL;
15920: Func IS_GOPHER_PLUS( GopherType : DWORD) : BOOL;
15921: Func IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL;
15922: Func
GopherCreateLocator( lpszHost:PChar; nServerPort:INTERNET_PORT; lpszDisplayString:PChar; lpszSelectorStr:PChar; dwGopherType
lpdwBufferLength:DWORD):BOOL;
15923: Func GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL;
15924: Func
GopherOpenFile( hConect:HINTERNET; lpszLocat:PChar; lpszView:PChar; dwFlgs:DWORD; dwContext:DWORD):HINTERNET;
15925: Func
HttpOpenRequest( hConnect:HINTERNET; lpszVerb:PChar; lpszObjectName:PChar; lpszVersion:PChar; lpszReferrer:
PChar; lplpszAcceptTypes:PLPSTR; dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15926: Func HttpAddRequestHeaders( hReq:HINTERNET; lpszHeaders:PChar; dwHeadersLength:DWORD; dwModifiers:DWORD):BOOL;
15927: Func HttpSendRequest( hRequest:HINTERNET; lpszHeaders:PChar; dwHeadersLength:DWORD; lpOptional:Tobject;
dwOptionalLength:DWORD):BOOL;
15928: Func InternetGetCookie( lpszUrl, lpszCookName, lpszCookieData:PChar; var lpdwSize:DWORD):BOOL;
15929: Func InternetAttemptConnect( dwReserved : DWORD) : DWORD;
15930: Func InternetCheckConnection( lpszUrl:PChar; dwFlags:DWORD; dwReserved:DWORD) : BOOL;
15931: Func InternetErrorDlg( hWnd:HWND; hRequest:HINTERNET; dwError, dwFlags:DWORD; var lppvData:TObject):DWORD;
15932: Func InternetConfirmZoneCrossing( hWnd:HWND; szUrlPrev, szUrlNew:LPSTR; bPost:BOOL) : DWORD;
15933: Func CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64;
15934: Func DeleteUrlCacheGroup( GroupId: Int64; dwFlags : DWORD; lpReserved : TObject) : Bool;
15935: Func FindFirstUrlCacheEntry( lpszUrlSearchPattern PChar; var
lpFirstCacheEntryInfo:TInternetCacheEntryInfo; var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15936: Func FindNextUrlCacheEntry( hEnumHandle:THandle; var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
lpdwNextCacheEntryInfoBufferSize:DWORD):BOOL;
15937: Func FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15938: Func DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15939: Func
InternetDial( hwnDParent:HWND; lpszConnect:PChar; dwFlgs:DWORD; lpdwConnect:DWORD; dwReserved:DWORD):DWORD;
15940: Func InternetSetDialState( lpszConnectoid:PChar; dwState:DWORD; dwReserved: DWORD) : BOOL;
15941: end;
15942:
15943: Proc SIRegister_Wwstr( CL: TPSPascalCompiler);
15944: begin
15945: AddTypeS('strCharSet', 'set of char;
15946: TwwGetWordOption', 'wwgwSkipLeadingBlanks, wwgwQuotesAsWords, wwgwStripQuotes , wwgwSpacesInWords);
15947: AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption;
15948: Proc strBreakApart( s :Str; delimiter :Str; parts : TStrings);
15949: Func strGetToken( s :Str; delimiter :Str; var APos : Int) :Str;
15950: Proc strStripPreceding( var s :Str; delimiter : strCharSet);
15951: Proc strStripTrailing( var s :Str; delimiter : strCharSet);
15952: Proc strStripWhiteSpace( var s :Str);
15953: Func strRemoveChar( str :Str; removeChar : char) :Str;
15954: Func wwstrReplaceChar( str :Str; removeChar, replaceChar : char) :Str;
15955: Func strReplaceCharWithStr( str:str; removeChar:char; replaceStr:Str) :Str;
15956: Func wwEqualStr( s1, s2 :Str) :Bool;
15957: Func strCount( s :Str; delimiter : char) : Int;
15958: Func strWhiteSpace : strCharSet;
15959: Func wwExtractFileNameOnly( const FileName :Str) :Str;
15960: Func wwGetWord( s:str; var APos:Int; Options:TwwGetWordOptions; DelimSet:strCharSet):str;
15961: Func strTrailing( s :Str; delimiter : char) :Str;
15962: Func strPreceding( s :Str; delimiter : char) :Str;
15963: Func wwstrReplace( s, Find, Replace :Str) :Str;
15964: end;

```



```

15965:
15966: Ex.:tex2:= 'Finally, we recognize that the 8% is calculated over all states of the world.'
15967: delims:= [' ','?',';','.',':','-','=', '{','}','[',']', '(',')','>','<','%'];
15968: while length(tex2)>0 do begin
15969:   k:=1;
15970:   s:= wwGetWord(tex2, k, [wwgwStripQuotes, wwgwSpacesInWords], delims);
15971:   delete(tex2,1,k);
15972:   write(' '+s)
15973: end;
15974:
15975: Proc SIRegister_DataBkr(CL: TPSPascalCompiler);
15976: begin
15977:   SIRegister_TRemoteDataModule(CL);
15978:   SIRegister_TCRemoteDataModule(CL);
15979:   Proc RegisterPooled(const ClassID:str; Max,Timeout : Int; Singleton :Bool);
15980:   Proc UnregisterPooled( const ClassID :Str);
15981:   Proc EnableSocketTransport( const ClassID :Str);
15982:   Proc DisableSocketTransport( const ClassID :Str);
15983:   Proc EnableWebTransport( const ClassID :Str);
15984:   Proc DisableWebTransport( const ClassID :Str);
15985: end;
15986:
15987: Proc SIRegister_Mathbox(CL: TPSPascalCompiler);
15988: begin
15989:   Func mxArcCos( x : Real) : Real;
15990:   Func mxArcSin( x : Real) : Real;
15991:   Func Comp2Str( N : Comp) :Str;
15992:   Func Int2StrPad0( N : LongInt; Len : Int) :Str;
15993:   Func Int2Str( N : LongInt) :Str;
15994:   Func mxIsEqual( R1, R2 : Double) :Bool;
15995:   Func LogXY( x, y : Real) : Real;
15996:   Func Pennies2Dollars( C : Comp) :Str;
15997:   Func mxPower( X : Int; Y : Int) : Real;
15998:   Func Real2Str( N : Real; Width, Places : Int) :Str;
15999:   Func mxStr2Comp( MyString :Str) : Comp;
16000:   Func mxStr2Pennies( S :Str) : Comp;
16001:   Func Str2Real( MyString :Str) : Real;
16002:   Func XToTheY( x, y : Real) : Real;
16003: end;
16004:
16005: //*****Cindy Functions!*****
16006: Proc SIRegister_cyIndy(CL: TPSPascalCompiler);
16007: begin
16008:   TypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
16009:   + 'Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
16010:   + 'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification);
16011:   MessagePlainText','String 'text/plain;
16012:   ConstantN('MessagePlainText_Attach','String 'multipart/mixed;
16013:   MessageAlterText_Html','String 'multipart/alternative;
16014:   MessageHtml_Attach','String 'multipart/mixed;
16015:   MessageHtml_RelatedAttach','String 'multipart/related; type="text/html";
16016:   MessageAlterText_Html_Attach','String 'multipart/mixed;
16017:   MessageAlterText_Html_RelatedAttach,String)('multipart/related;type="multipart/alternative";
16018:   MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed;
16019:   MessageReadNotificatin(String).('multipart/report;report-type="disposition-notification";
16020:   Func ForceDecodeHeader( aHeader :Str) :Str;
16021:   Func Base64_EncodeString( Value :Str; const aEncoding : TEncoding) :Str;
16022:   Func Base64_DecodeToString(Value:Str; const aBytesEncoding:TEncoding) :Str;;
16023:   Func Base64_DecodeToBytes( Value :Str) : TBytes;;
16024:   Func IdHttp_DownloadFile(aSrcUrlFile,aDestFile:str;const OnWorkEvent:TWorkEvent):Bool;
16025:   Func Get_MD5( const aFileName :Str) :Str;
16026:   Func Get_MD5FromString( const aString :Str) :Str;
16027: end;
16028:
16029: Proc SIRegister_cySysUtils(CL: TPSPascalCompiler);
16030: begin
16031:   Func IsFolder( SRec : TSearchrec) :Bool;
16032:   Func IsFolderReadOnly( Directory :Str) :Bool;
16033:   Func DirectoryIsEmpty( Directory :Str) :Bool;
16034:   Func DirectoryWithSubDir( Directory :Str) :Bool;
16035:   Proc GetSubDirs( FromDirectory :Str; aList : TStrings);
16036:   Func DiskFreeBytes( Drv : Char) : Int64;
16037:   Func DiskBytes( Drv : Char) : Int64;
16038:   Func GetFileBytes( Filename :Str) : Int64;
16039:   Func GetFilesBytes( Directory, Filter :Str) : Int64;
16040:   SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege;
16041:   SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege;
16042:   SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege;
16043:   SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege;
16044:   SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege;
16045:   SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege;
16046:   SE_TCB_NAME','String 'SeTcbPrivilege;
16047:   SE_SECURITY_NAME','String 'SeSecurityPrivilege;
16048:   SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege;
16049:   SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege;
16050:   SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege;
16051:   SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege;
16052:   SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege;
16053:   SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege;

```

```

16054: SE_CREATE_PAGEFILE_NAME', 'String 'SeCreatePagefilePrivilege;
16055: SE_CREATE_PERMANENT_NAME', 'String 'SeCreatePermanentPrivilege;
16056: SE_BACKUP_NAME', 'String 'SeBackupPrivilege;
16057: SE_RESTORE_NAME', 'String 'SeRestorePrivilege;
16058: SE_SHUTDOWN_NAME', 'String 'SeShutdownPrivilege;
16059: SE_DEBUG_NAME', 'String 'SeDebugPrivilege;
16060: SE_AUDIT_NAME', 'String 'SeAuditPrivilege;
16061: SE_SYSTEM_ENVIRONMENT_NAME', 'String 'SeSystemEnvironmentPrivilege;
16062: SE_CHANGE_NOTIFY_NAME', 'String 'SeChangeNotifyPrivilege;
16063: SE_REMOTE_SHUTDOWN_NAME', 'String 'SeRemoteShutdownPrivilege;
16064: SE_UNDOCK_NAME', 'String 'SeUndockPrivilege;
16065: SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege;
16066: SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege;
16067: SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege;
16068: end;
16069:
16070: Proc SIRegister_cyWinUtils (CL: TPSPascalCompiler);
16071: begin
16072:   Type (TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
16073:     +Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8 wvWin8_Or_Upper ) );
16074:   Func ShellGetExtensionName ( FileName : Str ) : Str;
16075:   Func ShellGetIconIndex ( FileName : Str ) : Int;
16076:   Func ShellGetIconHandle ( FileName : Str ) : HIcon;
16077:   Proc ShellThreadCopy ( App_Handle : THandle; fromFile : Str; toFile : Str);
16078:   Proc ShellThreadMove ( App_Handle : THandle; fromFile : Str; toFile : Str);
16079:   Proc ShellRenameDir ( DirFrom, DirTo:Str);
16080:   Func cyShellExecute (Operation, FileName, Parameters, Directory:Str; ShowCmd: Int) : Card;
16081:   Proc cyShellExecutel (ExeFilename, Params, ApplicName, ApplicationClass:Str; Restore:Bool);
16082:   Proc ShellExecuteAsModal ( ExeFilename, ApplicationName, Directory : Str);
16083:   Proc ShellExecuteExAsAdmin ( hWnd : HWND; Filename : Str; Parameters : Str);
16084:   Func ShellExecuteEx (aFileName:Str; const Parameters:Str; const Directory:Str; const
WaitCloseCompletion:bool) : Bool;
16085:   Proc RestoreAndSetForegroundWindow ( Hnd : Int);
16086:   Func RemoveDuplicatedPathDelimiter ( Str : Str ) : Str;
16087:   Func cyFileTimeToDateTime ( _FT : TFileTime ) : TDateTime;
16088:   Func GetModificationDate ( Filename : Str ) : TDateTime;
16089:   Func GetCreationDate ( Filename : Str ) : TDateTime;
16090:   Func GetLastAccessDate ( Filename : Str ) : TDateTime;
16091:   Func FileDelete ( Filename : Str ) : Bool;
16092:   Func FileIsOpen ( Filename : Str ) : Bool;
16093:   Proc FilesDelete ( FromDirectory : Str; Filter : ShortString);
16094:   Func DirectoryDelete ( Directory : Str ) : Bool;
16095:   Func GetPrinters ( PrintersList : TStrings ) : Int;
16096:   Proc SetDefaultPrinter ( PrinterName : Str);
16097:   Proc ShowDefaultPrinterWindowProperties ( FormParent_Handle : Int);
16098:   Func WinToDosPath ( WinPathName : Str ) : Str;
16099:   Func DosToWinPath ( DosPathName : Str ) : Str;
16100:   Func cyGetWindowsVersion : TWindowsVersion;
16101:   Func NTSetPrivilege ( sPrivilege : Str; bEnabled : Bool ) : Bool;
16102:   Proc WindowsShutDown ( Restart : Bool);
16103:   Proc CreateShortcut (FileSrc, Parametres, FileLnk, Descript, DossierDeTravail, FileIcon:Str; NumIcône: Int);
16104:   Proc GetWindowsFonts ( FontsList : TStrings);
16105:   Func GetAvailableFilename ( DesiredFileName : Str): Str;
16106: end;
16107:
16108: Proc SIRegister_cyStrUtils (CL: TPSPascalCompiler);
16109: begin
16110:   Type (TStrLocateOption', '( strloCaseInsensitive, strloPartialKey );
16111:   Type (TStrLocateOptions', 'set of TStrLocateOption;
16112:   Type (TStringRead', '( srFromLeft, srFromRight );
16113:   Type (TStringReads', 'set of TStringRead;
16114:   Type (TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive );
16115:   Type (TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar );
16116:   Type (TWordsOptions', 'set of TWordsOption;
16117:   Type (TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther );
16118:   Type (TCarTypes', 'set of TCarType;
16119:   //TypeS('TUnicodeCategories', 'set of TUnicodeCategory;
16120:   CarTypeAlphabetic', 'LongInt') := ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
16121:   Func Char_GetType ( aChar : Char ) : TCarType;
16122:   Func SubString_Count ( Str : Str; Separator : Char ) : Int;
16123:   Func SubString_AtPos ( Str : Str; Separator : Char; SubStringIndex : Word ) : Int;
16124:   Func SubString_Get ( Str : Str; Separator : Char; SubStringIndex : Word ) : Str;
16125:   Func SubString_Length ( Str : Str; Separator : Char; SubStringIndex : Word ) : Int;
16126:   Proc SubString_Add ( var Str : Str; Separator : Char; Value : Str);
16127:   Proc SubString_Insert (var Str:Str; Separator:Char; SubStrIndex:Word; Value:Str);
16128:   Proc SubString_Edit (var Str:Str; Separator:Char; SubStringIndex:Word; NewValue:Str);
16129:   Func SubString_Remove (var Str:Str; Separator:Char; SubStringIndex : Word) : Bool;
16130:   Func SubString_Locate (Str:Str; Separator:Char; SubString:Str; Options:TStrLocateOptions): Int;
16131:   Func SubString_Ribbon (Str:Str; Separator:Char; Current: Word; MoveBy: Int): Int;;
16132:   Func SubString_Ribbon1 (Str:Str; Separator:Char; Current:Str; MoveBy: Int): Str;;
16133:   Func String_Quote ( Str : Str ) : Str;
16134:   Func String_GetCar (Str:Str; Position : Word; ReturnCarIfNotExists : Char) : Char;
16135:   Func String_ExtractCars (fromStr:Str; CarTypes:TCarTypes; IncludeCars, ExcludeCars:Str): Str;
16136:   Func String_GetWord ( Str : Str; StringRead : TStringRead ) : Str;
16137:   Func String_GetInt ( Str : Str; StringRead : TStringRead ) : Str;
16138:   Func String_ToInt ( Str : Str ) : Int;
16139:   Func String_Uppercase ( Str : Str; Options : TWordsOptions ) : Str;
16140:   Func String_Lowercase ( Str : Str; Options : TWordsOptions ) : Str;
16141:   Func String_Reverse ( Str : Str ) : Str;

```

```

16142: Func String_Pos(SubStr:str;Str:str;fromPos:Int;CaseSensitive:TCaseSensitiv)Int;
16143: Func
String_Pos1(SubStr:str;Str:str;StringRead:TStringRead;Occurrence:Word;CaseSensitive:TCaseSensitive):Int
16144: Func String_Copy( Str :Str; fromIndex : Int; toIndex : Int) :Str;;
16145: Func String_Copy1(Str:str;StringRead:TStringRead;UntilFind:Str;_Inclusive:Bool):str;
16146: Func String_Copy2(Str:str;Between1:str;Between1MustExist:Boolean;Between2:str;
Between2MustExist:Boolean;CaseSensitive:TCaseSensitive):str;
16147: Func String_Delete( Str :Str; fromIndex : Int; toIndex : Int) :Str;;
16148: Func String_Delete1(Str:str; delStr:str; CaseSensitive:TCaseSensitive):str;;
16149: Func String_BoundsCut( Str :Str; CutCar : Char; Bounds : TStringReads) :Str;
16150: Func String_BoundsAdd( Str :Str; AddCar : Char; ReturnLength : Int) :Str;
16151: Func String_Add(Str:str;StringRead:TStringRead;aCar:Char;ReturnLength:Int):str;
16152: Func String_End( Str :Str; Cars : Word) :Str;
16153: Func String_Subst(OldStr:str;NewStr:str;Str:str;
CaseSensitive:TCaseSensitive;AlwaysFindFromBeginning:Bool):Str;
16154: Func String_SubstCar( Str :Str; Old, New : Char) :Str;
16155: Func String_Count(Str:Str; SubStr:Str; CaseSenSitive : TCaseSensitive) : Int;
16156: Func String_SameCars(Str1,Str2:str;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Int;
16157: Func String_IsNumbers( Str :Str) :Bool;
16158: Func SearchPos( SubStr :Str; Str :Str; MaxErrors : Int) : Int;
16159: Func StringToCsvCell( aStr :Str) :Str;
16160: end;
16161:
16162: Proc SIRegister_cyDateUtils(CL: TPSPascalCompiler);
16163: begin
16164: Func LongDayName( aDate : TDate) :Str;
16165: Func LongMonthName( aDate : TDate) :Str;
16166: Func ShortYearOf( aDate : TDate) : byte;
16167: Func DateToStrYYYYMMDD( aDate : TDate) :Str;
16168: Func StrYYYYMMDDToDate( aStr:Str) : TDate;
16169: Func SecondsToMinutes( Seconds : Int) : Double;
16170: Func MinutesToSeconds( Minutes : Double) : Int;
16171: Func MinutesToHours( Minutes : Int) : Double;
16172: Func HoursToMinutes( Hours : Double) : Int;
16173: Func ShortTimeStringToTime(ShortTimeStr:str;const ShortTimeFormat:str): TDateTime;
16174: Proc cyAddMonths( var aMonth, aYear : Word; Months : Int);
16175: Func MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime;
16176: Func GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;;
16177: Func GetMinutesBetween1(From ShortTimeStr,To ShortTimeStr:str;const ShortTimeFormat:str):Int64;
16178: Func GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;;
16179: Func IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime;var
RsltBegin:TDateTime;RsltEnd:TDateTime):Bool;
16180: Func IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
16181: Func TryToEncodeDate( Year, Month, Day : Int; var RsltDate : TDateTime) :Bool;
16182: end;
16183:
16184: Proc SIRegister_cyObjUtils(CL: TPSPascalCompiler);
16185: begin
16186: Type(TStringsSortType','( stNone, stStringSensitive, stStringInsensitive, stExtended) );
16187: Type(TStringsValueKind','( skStringSensitive, skStringInsensitive, skExtended) );
16188: Type(TcyLocateOption','( lCaseInsensitive, lPartialKey) );
16189: Type(TcyLocateOptions','set of TcyLocateOption;
16190: Func StringsLocate(aList: TStrings; Value:str; Options:TcyLocateOptions): Int;;
16191: Func StringsLocate1(aList:TStrings; Value:Str; ValueKind:TStringsValueKind):Int;;
16192: Func StringsAdd(aList:TStrings;Value:str;Unique:Boolean;SortType:TStringsSortType): Int;
16193: Proc StringsReplace(aList:TStrings;OldStr:str; NewStr:str;ValueKind : TStringsValueKind);
16194: Proc StringsSort( aList : TStrings; SortType : TStringsSortType);
16195: Func TreeNodeLocate( ParentNode : TTreeNode; Value :Str) : TTreeNode;
16196: Func TreeNodeLocateOnLevel(TreeView:TTreeView;OnLevel:Int;Value:str): TTreeNode;
16197: Func TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Int):TTreeNode;
16198: Func TreeNodeGetParentOnLevel(ChildNode: TTreeNode; ParentLevel : Int) : TTreeNode;
16199: Proc TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
CopySubChildren:Bool;
16200: Proc RichEditSetStr( aRichEdit : TRichEdit; FormatedString :Str);
16201: Proc RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:str;Flags:TReplaceFlags);
16202: Func GetTopMostControlAtPos(FromControl:TWinControl; aControlPoint : TPoint): TControl;
16203: Proc cyCenterControl( aControl : TControl);
16204: Func GetLastParent( aControl : TControl) : TWinControl;
16205: Func GetControlBitmap( aControl : TWinControl) : TBitmap;
16206: Func GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap;
16207: end;
16208:
16209: Proc SIRegister_cyBDE(CL: TPSPascalCompiler);
16210: begin
16211: Func TablePackTable( Tab : TTable) :Bool;
16212: Func TableRegenIndexes( Tab : TTable) :Bool;
16213: Func TableShowDeletedRecords( Tab : TTable; Show :Bool) :Bool;
16214: Func TableUndeleteRecord( Tab : TTable) :Bool;
16215: Func TableAddIndex(Tab:TTable;FieldName Str;FieldExpression:Str;IOpt:TIndexOptions):Bool;
16216: Func TableDeleteIndex( Tab : TTable; IndexFieldName :Str) :Bool;
16217: Func TableEmptyTable( Tab : TTable) :Bool;
16218: Func TableFindKey( aTable : TTable; Value :Str) :Bool;
16219: Proc TableFindNearest( aTable : TTable; Value :Str);
16220: Func
TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:str;IdxName:ShortString;ReadOnly:Bool):TTable;
16221: Func
TableOpen(Tab:TTable;FileName:str;IndexFieldName:str;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
16222: Func DateToBDESQLDate( aDate : TDate; const DateFormat :Str) :Str;
16223: end;

```

```

16224:
16225: Proc SIRegister_cyClasses(CL: TPSPascalCompiler);
16226: begin
16227:   SIRegister_TcyRunTimeDesign(CL);
16228:   SIRegister_TcyShadowText(CL);
16229:   SIRegister_TcyBgPicture(CL);
16230:   SIRegister_TcyGradient(CL);
16231:   SIRegister_tcyBevel(CL);
16232:   //TypeS('TcyBevelClass', 'class of tcyBevel;
16233:   SIRegister_tcyBevels(CL);
16234:   SIRegister_TcyImagelistOptions(CL);
16235: Proc cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture);
16236: end;
16237:
16238: Proc SIRegister_cyGraphics(CL: TPSPascalCompiler);
16239: begin
16240: Proc cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; adgradOrientation :
TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode; Maxdegrade : Byte;'+
16241: SpeedPercent : Int; const AngleClipRect :Bool; const AngleBuffer : TBitmap);
16242: Proc cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrade:byte);
16243: Proc cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromCol,toColor:TColor;MaxDegrade:byte);
16244: Proc cyGradientFillShape(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrade:
Byte;toRect:TRect;OrientationShape:TDgradOrientationShape);
16245: Proc cyGradientFillAngle(aCanvas:TCanvas;aRect TRect;fromColor,toColor:TColor;MaxDegrade:
Byte;AngleDegree:Word;const ClipRect:Boolean;const Buffer:TBitmap);
16246: Proc DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Int);
16247: Proc cyFrame(aCanvas:TCanvas; var InsideRect:TRect;Color:TColor;const Width:Int);
16248: Proc cyFrame1(Canvas:TCanvas;var InsideRect:TRect;LeftColor,TopColor,RightColor,BottomColor:TColor;const
Width:Int;const RoundRect:bool);
16249: 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;
16250: Proc cyDrawButtonFace(Canvas:TCanvas; var Rect:Rect;GradientColor1,
GradientColor2:TColor;aState:TButtonState;Focused,Hot:Bool);
16251: Proc cyDrawButton(Canvas:TCanvas;Caption:str;ARect:TRect;radientColor1,GradientColor2:TColor;aState
TButtonState;Focused,Hot : Bool);
16252: Proc cyDrawSpeedButtonFace(Canvas:TCanvas;var Rect:TRect;GradientColor1,
GradientColor2:TColor;aState:TButtonState;Focused,Hot:Bool);
16253: Proc cyDrawSpeedButton(Canvas:TCanvas;Caption:str;ARect:TRect;GradientColor1,
GradientColor2:TColor;aState: TButtonState;Focused,Hot:oolean);
16254: Proc cyDrawCheckBox(Canvas:TCanvas;IsChecked:Boolean;ARect:TRect;const BgColor:TColor;const
DarkFrameColor:TColor;const LightFrameColor:TColor;const MarkColor:TColor);
16255: Proc
cyDrawSingleLineText(Canvas:TCanvas;Text:str;ARect:TRect;Alignment:TAlignment;TextLayout:TTextLayout; const
IndentX:Int;const IndentY:Int);
16256: Func DrawTextFormatFlags(aTextFormat:LongInt;Alignment
TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
16257: Func DrawTextFormatFlags1(aTextFormat:LongInt;Alignment:TAlignment;Layout:TTextLayout;
WordWrap:Boolean;CaptionRender:TCaptionRender):LongInt;
16258: Proc cyDrawText(Canvas:Canvas;Card;Text:str;var Rect:TRect;TextFormat:LongInt);
16259: Func cyCreateFontIndirect(fromFont: TFont; Angle : Double) : TFont;
16260: Func cyCreateFontIndirect1(fromFont: TFont;CaptionOrientation:TCaptionOrientation):TFont;
16261: Proc cyDrawVerticalText(Canvas : TCanvas; Text :Str; var Rect : TRect; TextFormat : Longint;
CaptionOrientation:TCaptionOrientation;Alignment:TAlignment;Layout: TextLayout);
16262: Func DrawLeftTurnPageEffect(Canvas:TCanvas;PageColor:TColor;PageRect:TRect;PercentDone:Int; const
OnlyCalcFoldLine:Boolean):TLineCoord;
16263: Func DrawRightTurnPageEffect(Canvas:TCanvas;PageColor:TColor;PageRect:TRect;PercentDone:Int;const
OnlyCalcFoldLine:Boolean):TLineCoord;
16264: Func PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean;
16265: Func IconIsTransparentAtPos(aIcon : TIcon; aPoint : TPoint) :Bool;
16266: Func MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) :Bool;
16267: Func PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) :Bool;
16268: Proc DrawCanvas(Destination:TCanvas;DestRect:TRect;SrcRect:TRect;Source:TCanvas;SourceRect:TRect);
16269: 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);
16270: 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;
16271: 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);
16272: Proc DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap);
16273: Func ValidGraphic( aGraphic : TGraphic) :Bool;
16274: Func ColorSetPercentBrightness( Color : TColor; PercentLight : Int) : TColor;
16275: Func ColorModify( Color : TColor; incR, incG, incB : Int) : TColor;
16276: Func ColorSetPercentContrast( Color : TColor; IncPercent : Int) : TColor;
16277: Func ColorSetPercentPale( Color : TColor; IncPercent : Int) : TColor;
16278: Func MediumColor( Color1, Color2 : TColor) : TColor;
16279: Func ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect;
16280: Func ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect;
16281: Func CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect;
16282: Proc InflateRectPercent( var aRect : TRect; withPercent : Double);
16283: Func GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect;
16284: Func GetProportionalRect( fromRect, InsideRect : TRect) : TRect;
16285: Func PointInRect( const aPt : TPoint; const aRect : TRect) :Bool;
16286: Func PointInEllipse( const aPt : TPoint; const aRect : TRect) :Bool;
16287: Func CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText :Str) : Int;

```

```

16288: end;
16289:
16290: Proc SIRegister_cyTypes (CL: TPSPascalCompiler);
16291: begin
16292:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight );
16293:   Type(TGlyphLayout', '( glTop, glCenter, glBottom );
16294:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome );
16295:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis );
16296:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed );
16297:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,
16298:     + ' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft);
16299:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional );
16300:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext );
16301:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle );
16302:   Type(TDgradOrientationShape', '( osRadial, osRectangle );
16303:   Type(TDgradBalanceMode', (bmNormal, bmMirror, bmReverse, bmReverseFromColor, bmInvertReverse,
16304:     bmInvertReverseFromColor);
16305:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft, rjResizeTopLeft,
16306:     rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight );
16307:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end;
16308:   cCaptionOrientationWarning, String (Note text orientation doesnt work with all fonts!;
16309: end;
16310:
16311: Proc SIRegister_WinSvc (CL: TPSPascalCompiler);
16312: begin
16313:   Const SERVICES_ACTIVE_DATABASE', 'String 'ServicesActive;
16314:   SERVICES_ACTIVE_DATABASE', 'String') 'ServicesActive;
16315:   Const SERVICES_FAILED_DATABASE', 'String') 'SERVICES_ACTIVE_DATABASE;
16316:   Const SERVICES_FAILED_DATABASE', 'String') 'ServicesFailed;
16317:   Const SERVICES_FAILED_DATABASE', 'String') 'SERVICES_FAILED_DATABASE;
16318:   Const SC_GROUP_IDENTIFIER', 'String') . '+';
16319:   Const SC_GROUP_IDENTIFIER', 'String') '+';
16320:   Const SC_GROUP_IDENTIFIER', 'string 'SC_GROUP_IDENTIFIER;
16321:   Const SERVICE_NO_CHANGE', 'LongWord $FFFFFFF);
16322:   Const SERVICE_ACTIVE', 'LongWord') ( $00000001);
16323:   Const SERVICE_INACTIVE', 'LongWord $00000002);
16324:   Const SERVICE_CONTROL_STOP', 'LongWord $00000001);
16325:   Const SERVICE_CONTROL_PAUSE', 'LongWord $00000002);
16326:   Const SERVICE_CONTROL_CONTINUE', 'LongWord $00000003);
16327:   Const SERVICE_CONTROL_INTERROGATE', 'LongWord $00000004);
16328:   Const SERVICE_CONTROL_SHUTDOWN', 'LongWord $00000005);
16329:   Const SERVICE_STOPPED', 'LongWord $00000001);
16330:   Const SERVICE_START_PENDING', 'LongWord $00000002);
16331:   Const SERVICE_STOP_PENDING', 'LongWord $00000003);
16332:   Const SERVICE_RUNNING', 'LongWord $00000004);
16333:   Const SERVICE_CONTINUE_PENDING', 'LongWord $00000005);
16334:   Const SERVICE_PAUSE_PENDING', 'LongWord $00000006);
16335:   Const SERVICE_PAUSED', 'LongWord $00000007);
16336:   Const SERVICE_ACCEPT_STOP', 'LongWord $00000001);
16337:   Const SERVICE_ACCEPT_PAUSE_CONTINUE', 'LongWord $00000002);
16338:   Const SERVICE_ACCEPT_SHUTDOWN', 'LongWord $00000004);
16339:   Const SC_MANAGER_CONNECT', 'LongWord $0001);
16340:   Const SC_MANAGER_CREATE_SERVICE', 'LongWord $0002);
16341:   Const SC_MANAGER_ENUMERATE_SERVICE', 'LongWord $0004);
16342:   Const SC_MANAGER_LOCK', 'LongWord $0008);
16343:   Const SC_MANAGER_QUERY_LOCK_STATUS', 'LongWord $0010);
16344:   Const SC_MANAGER_MODIFY_BOOT_CONFIG', 'LongWord $0020);
16345:   Const SERVICE_QUERY_CONFIG', 'LongWord $0001);
16346:   Const SERVICE_CHANGE_CONFIG', 'LongWord $0002);
16347:   Const SERVICE_QUERY_STATUS', 'LongWord $0004);
16348:   Const SERVICE_ENUMERATE_DEPENDENTS', 'LongWord $0008);
16349:   Const SERVICE_START', 'LongWord $0010);
16350:   Const SERVICE_STOP', 'LongWord $0020);
16351:   Const SERVICE_PAUSE_CONTINUE', 'LongWord $0040);
16352:   Const SERVICE_INTERROGATE', 'LongWord $0080);
16353:   Const SERVICE_USER_DEFINED_CONTROL', 'LongWord $0100);
16354:   Const SERVICE_KERNEL_DRIVER', 'LongWord $00000001);
16355:   Const SERVICE_FILE_SYSTEM_DRIVER', 'LongWord $00000002);
16356:   Const SERVICE_ADAPTER', 'LongWord $00000004);
16357:   Const SERVICE_RECOGNIZER_DRIVER', 'LongWord $00000008);
16358:   Const SERVICE_WIN32_OWN_PROCESS', 'LongWord $00000010);
16359:   Const SERVICE_WIN32_SHARE_PROCESS', 'LongWord $00000020);
16360:   Const SERVICE_INTERACTIVE_PROCESS', 'LongWord $00000100);
16361:   Const SERVICE_BOOT_START', 'LongWord $00000000);
16362:   Const SERVICE_SYSTEM_START', 'LongWord $00000001);
16363:   Const SERVICE_AUTO_START', 'LongWord $00000002);
16364:   Const SERVICE_DEMAND_START', 'LongWord $00000003);
16365:   Const SERVICE_DISABLED', 'LongWord $00000004);
16366:   Const SERVICE_ERROR_IGNORE', 'LongWord $00000000);
16367:   Const SERVICE_ERROR_NORMAL', 'LongWord $00000001);
16368:   Const SERVICE_ERROR_SEVERE', 'LongWord $00000002);
16369:   Const SERVICE_ERROR_CRITICAL', 'LongWord $00000003);
16370:   Types('SC_HANDLE', 'THandle;
16371:   //TypeS('LPSC_HANDLE', '^SC_HANDLE // will not work;
16372:   TypeS('SERVICE_STATUS_HANDLE', 'DWORD;
16373:   Const SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
16374:     + ': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
16375:     + 'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end;

```



```

16375:  Const SERVICE_STATUS', '_SERVICE_STATUS;
16376:  Const TServiceStatus', '_SERVICE_STATUS;
16377:  TypeS(' _ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
16378:  + 'playName : PChar; ServiceStatus : TServiceStatus; end;
16379:  ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA;
16380:  _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA;
16381:  TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA;
16382:  TEnumServiceStatus', 'TEnumServiceStatusA;
16383:  SC_LOCK', '_Pointer;
16384:  _QUERY_SERVICE_LOCK_STATUSA, record fIsLocked:DWORD;lpLockOwner:PChar;dwLockDuration:DWORD;end;
16385:  _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA;
16386:  QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA;
16387:  QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA;
16388:  TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA;
16389:  //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW;
16390:  TQueryServiceLockStatus', 'TQueryServiceLockStatusA;
16391:  _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
16392:  + 'ype : DWORD; dwErrorControl:DWORD;lpBinaryPathName: PChar; lpLoadO'
16393:  + 'rderGroup:PChar;dwTagId:DWORD;lpDependencies:PChar;lpServiceStartName:PChar;lpDisplayName:PChar;end;
16394:  _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA;
16395:  QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA;
16396:  QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA;
16397:  TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA;
16398:  TQueryServiceConfig', 'TQueryServiceConfigA;
16399:  Func CloseServiceHandle( hSCObject : SC_HANDLE ) : BOOL;
16400:  Func ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
16401:  Func CreateService(hSCManager:SC_HANDLE;lpServiceName,lpDisplayName:PChar;dwDesiredAccess,dwServiceType,
dwStartType,dwErrorControl:DWORD;lpBinaryPathName,lpLoadOrderGroup:PChar;
16402:  + ' lpdwTagId: DWORD;lpDependencies,lpServiceStartName, lpPassword : PChar): SC_HANDLE;
16403:  Func CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
16404:  + 'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE;
16405:  Func DeleteService( hService : SC_HANDLE) : BOOL;
16406:  Func EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD; var
lpServices:TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned: DWORD) : BOOL;
16407:  Func EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState:DWORD;var lpServices:
TEnumServiceStatus;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,lpResumeHandle:DWORD) : BOOL;
16408:  Func GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var lpccchBuffer:DWORD) : BOOL;
16409:  Func GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceName,lpDisplayName:PChar;var
lpccchBuffer:DWORD) : BOOL;
16410:  Func LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK;
16411:  Func NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL;
16412:  Func OpenSCManager(lpMachineName,lpDatabaseName:PChar;dwDesiredAccess:DWORD):SC_HANDLE;
16413:  Func OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE;
16414:  Func QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL;
16415:  Func QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
16416:  Func SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
16417:  Func StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
16418:  Func UnlockServiceDatabase(SCLock : SC_LOCK) : BOOL;
16419:  end;
16420:
16421: Proc SIRegister_JvPickDate(CL: TPSPascalCompiler);
16422: begin
16423:  Func SelectDate(Sender:TWinControl; var Date:TDateTime; const DlgCaption:TCaption; AStartOfWeek:
TDayOfWeekName;AWeekends:TDaysOfWeek;AWeekendColor:TColor;BtnHints:TStrings;MinDate:TDateTime;MaxDate:TDateTime):Bool;
16424:  Func SelectDateStr(Sender:TWinControl;var StrDate:str;const
DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDate
16425:  Func PopupDate(var Date:TDateTime;Edit:TWinControl;MinDate:TDateTime;MaxDate:TDateTime):Bool;
16426:  Func
CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):TWinControl;
16427:  Proc SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
16428:  Func CreateNotifyThread(const FolderName:str;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
16429:  end;
16430:
16431: Proc SIRegister_JclNTFS2(CL: TPSPascalCompiler);
16432: begin
16433:  ClassN(CL.FindClass('TObject'),'EJclNTfsError;
16434:  TypeS(TFileCompressionState,(fcNoCompression,fcDefaultCompression,fcLZNT1Compression);
16435:  Func NtfsGetCompression2( const FileName : TFileName; var State : Short) :Bool;;
16436:  Func NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;;
16437:  Func NtfsSetCompression2( const FileName : TFileName; const State : Short) :Bool;
16438:  Proc NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState);
16439:  Proc NtfsSetDirectoryTreeCompression2(const Directy:str;const State:TFileCompressionState);
16440:  Proc NtfsSetDefaultFileCompression2(const Directory:Str;const State:TFileCompressionState);
16441:  Proc NtfsSetPathCompression2(const Path:str;const State:TFileCompressState;Recurve:Bool);
16442:  Func NtfsSetSparse2( const FileName :Str) :Bool;
16443:  Func NtfsZeroDataByHandle2(const Handle: THandle; const First, Last : Int64) :Bool;
16444:  Func NtfsZeroDataByName2(const FileName:Str;const First,Last: Int64) :Bool;
16445:  Func NtfsSparseStreamsSupported2( const Volume :Str) :Bool;
16446:  Func NtfsGetSparse2( const FileName :Str) :Bool;
16447:  Func NtfsDeleteReparsePoint2( const FileName :Str; ReparseTag : DWORD) :Bool;
16448:  Func NtfsSetReparsePoint2(const FileName:str;var ReparseData,Size : Longword) : Bool;
16449:  Func NtfsGetReparseTag2( const Path :Str; var Tag : DWORD) :Bool;
16450:  Func NtfsReparsePointsSupported2( const Volume :Str) :Bool;
16451:  Func NtfsFileHasReparsePoint2( const Path :Str) :Bool;
16452:  Func NtfsIsFolderMountPoint2( const Path :Str) :Bool;
16453:  Func NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) :Bool;

```

```

16454: Func NtfsMountVolume2(const Volume:WideChar;const MountPoint:WideString) :Bool;
16455:   TypeS('TopLock', '( olExclusive, olReadOnly, olBatch, olFilter );
16456: Func NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped ) :Bool;
16457: Func NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped ) :Bool;
16458: Func NtfsOpLockBreakAcknowledge2(Handle:THandle; Overlapped: TOverlapped) :Bool;
16459: Func NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped ) :Bool;
16460: Func NtfsRequestOpLock2(Handle:THandle; Kind:TopLock; Overlapped: TOverlapped):Boolean;
16461: Func NtfsCreateJunctionPoint2( const Source, Destination :Str) :Bool;
16462: Func NtfsDeleteJunctionPoint2( const Source :Str) :Bool;
16463: Func NtfsGetJunctionPointDestination2(const Source:str;var Destination:Str):Bool;
16464:   TypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
16465:     +urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile);
16466:   TypeS('TStreamIds', 'set of TStreamId;
16467:   TInternalFindStreamData', 'record FileHandle:THandle;Context:TObject; StreamIds:TStreamIds;end;
16468:   TypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
16469:     +tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end;
16470: Func NtfsFindFirstStream2(const FileName:str;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
16471: Func NtfsFindNextStream2( var Data : TFindStreamData) :Bool;
16472: Func NtfsFindStreamClose2( var Data : TFindStreamData) :Bool;
16473: Func NtfsCreateHardLink2( const LinkFileName, ExistingFileName :Str) :Bool;
16474: Func NtfsCreateHardLinkA2(const LinkFileName, ExistingFileName:Ansistr):Bool;
16475: Func NtfsCreateHardLinkW2(const LinkFileName, ExistingFileName: WideString):Bool;
16476:   TypeS('TNtfsHardLinkInfo', 'record LinkCount :Card; FileIndex : Int64; end;
16477: Func NtfsGetHardLinkInfo2(const FileName:str;var Info:TNtfsHardLinkInfo):Boolean;
16478: Func NtfsFindHardLinks2(const Path:str;const FileIdxHigh, FIdxLow:Card;const List:TStrings):Bool
16479: Func NtfsDeleteHardLinks2( const FileName :Str) :Bool;
16480: FindClass('TOBJECT'), 'EJclFileSummaryError;
16481: TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite );
16482: TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll );
16483: TJclFileSummaryPropSetCallback', 'Func ( const FMTID : TGUID ) :Bool;
16484: ClassN(CL.FindClass('TOBJECT'), 'TJclFileSummary;
16485: SIRegister_TJclFilePropertySet(CL);
16486: //TypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet;
16487: SIRegister_TJclFileSummary(CL);
16488: SIRegister_TJclFileSummaryInformation(CL);
16489: SIRegister_TJclDocSummaryInformation(CL);
16490: SIRegister_TJclMediaFileSummaryInformation(CL);
16491: SIRegister_TJclMSISummaryInformation(CL);
16492: SIRegister_TJclShellSummaryInformation(CL);
16493: SIRegister_TJclStorageSummaryInformation(CL);
16494: SIRegister_TJclImageSummaryInformation(CL);
16495: SIRegister_TJclDisplacedSummaryInformation(CL);
16496: SIRegister_TJclBriefCaseSummaryInformation(CL);
16497: SIRegister_TJclMiscSummaryInformation(CL);
16498: SIRegister_TJclWebViewSummaryInformation(CL);
16499: SIRegister_TJclMusicSummaryInformation(CL);
16500: SIRegister_TJclDRMSummaryInformation(CL);
16501: SIRegister_TJclVideoSummaryInformation(CL);
16502: SIRegister_TJclAudioSummaryInformation(CL);
16503: SIRegister_TJclControlPanelSummaryInformation(CL);
16504: SIRegister_TJclVolumeSummaryInformation(CL);
16505: SIRegister_TJclShareSummaryInformation(CL);
16506: SIRegister_TJclLinkSummaryInformation(CL);
16507: SIRegister_TJclQuerySummaryInformation(CL);
16508: SIRegister_TJclImageInformation(CL);
16509: SIRegister_TJclJpegSummaryInformation(CL);
16510: end;
16511:
16512: Proc SIRegister_Jcl8087(CL: TPSPascalCompiler);
16513: begin
16514:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended );
16515:   T8087Rounding', '( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate );
16516:   T8087Infinity', '( icProjective, icAffine );
16517:   T8087Exception', '( emInvalidOp, emDenormalizedOperand, emZeroDivide, emOverflow, emUnderflow, emPrecision;
16518:   TypeS('T8087Exceptions', set of T8087Exception;
16519:   Func Get8087ControlWord : Word;
16520:   Func Get8087Infinity : T8087Infinity;
16521:   Func Get8087Precision : T8087Precision;
16522:   Func Get8087Rounding : T8087Rounding;
16523:   Func Get8087StatusWord( ClearExceptions :Bool) : Word;
16524:   Func Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity;
16525:   Func Set8087Precision( const Precision : T8087Precision) : T8087Precision;
16526:   Func Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding;
16527:   Func Set8087ControlWord( const Control : Word) : Word;
16528:   Func ClearPending8087Exceptions : T8087Exceptions;
16529:   Func GetPending8087Exceptions : T8087Exceptions;
16530:   Func GetMasked8087Exceptions : T8087Exceptions;
16531:   Func SetMasked8087Exceptions( Exceptions:T8087Exceptions;ClearBefore:Bool):T8087Exceptions;
16532:   Func Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions;
16533:   Func Unmask8087Exceptions( Exceptions:T8087Exceptions;ClearBefore:Bool):T8087Exceptions;
16534: end;
16535:
16536: Proc SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
16537: begin
16538:   Proc BoxMoveSelectedItems( SrcList, DstList : TWinControl);
16539:   Proc BoxMoveAllItems( SrcList, DstList : TWinControl);
16540:   Proc BoxDragOver(List:TWinControl;Sorce:TObject;X,Y:Int;State:TDragState;var Accept:Bool;Sorted:Bool;
16541:   Proc BoxMoveFocusedItem( List : TWinControl; DstIndex : Int);
16542:   Proc BoxMoveSelected( List : TWinControl; Items : TStrings);

```

```

16543: Proc BoxSetItem( List : TWinControl; Index : Int);
16544: Func BoxGetFirstSelection( List : TWinControl) : Int;
16545: Func BoxCanDropItem(List:TWinControl; X,Y:Int; var DragIndex : Int) :Bool;
16546: end;
16547:
16548: Proc SIRegister_UrlMon(CL: TPSPascalCompiler);
16549: begin
16550:   //ConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context;
16551:   //ConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee;
16552:   ConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
16553:   type ULONG, 'Cardinal; LPCWSTR, 'PChar; TypeS('LPWSTR', 'PChar;
16554:   LPSTR, 'PChar;
16555:   TBindVerb, 'ULONG;
16556:   TBindInfoF, 'ULONG;
16557:   TBindF, 'ULONG;
16558:   TBSCF, 'ULONG;
16559:   TBindStatus, 'ULONG;
16560:   TCIPStatus, 'ULONG;
16561:   TBindString, 'ULONG;
16562:   TPiFlags, 'ULONG;
16563:   TOIBdgFlags, 'ULONG;
16564:   TParseAction, 'ULONG;
16565:   TPSUAction, 'ULONG;
16566:   TQueryOption, 'ULONG;
16567:   TPUAF, 'ULONG;
16568:   TSZMFlags, 'ULONG;
16569:   TUrlZone, 'ULONG;
16570:   TUrlTemplate, 'ULONG;
16571:   TZAFlags, 'ULONG;
16572:   TUrlZoneReg, 'ULONG;
16573:   'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
16574:   ConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
16575:   const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
16576:   const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
16577:   const 'CF_NULL','LongInt').SetInt( 0);
16578:   const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
16579:   const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain;
16580:   const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext;
16581:   const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap;
16582:   const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript;
16583:   const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff;
16584:   const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic;
16585:   const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav;
16586:   const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav;
16587:   const 'CFSTR_MIME_GIF','String').SetString( 'image/gif;
16588:   const 'CFSTR_MIME_PJPEG','String').SetString( 'image/jpeg;
16589:   const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg;
16590:   const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff;
16591:   const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png;
16592:   const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp;
16593:   const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg;
16594:   const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf;
16595:   const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf;
16596:   const 'CFSTR_MIME_AVI','String').SetString( 'video/avi;
16597:   const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg;
16598:   const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals;
16599:   const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream;
16600:   const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream;
16601:   const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf;
16602:   const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff;
16603:   const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio;
16604:   const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm;
16605:   const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime;
16606:   const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo;
16607:   const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie;
16608:   const 'CFSTR_MIME_HTML','String').SetString( 'text/html;
16609:   const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
16610:   const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
16611:   const 'E_PENDING','LongWord').SetUInt( $8000000A);
16612:   Interface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding;
16613:   SIRegister_IPersistMoniker(CL);
16614:   SIRegister_IBindProtocol(CL);
16615:   SIRegister_IBinding(CL);
16616:   const BINDVERB_GET','LongWord').SetUInt( $00000000);
16617:   const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
16618:   const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
16619:   const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
16620:   const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
16621:   const 'BINDINFOF_URLENCODEDEXTRAIINFO','LongWord').SetUInt( $00000002);
16622:   const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
16623:   const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
16624:   const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
16625:   const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
16626:   const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
16627:   const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
16628:   const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16629:   const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16630:   const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16631:   const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);

```

```

16632: const 'BINDF_HYPERLINK', 'LongWord').SetUInt( $00000400);
16633: const 'BINDF_NO_UI', 'LongWord').SetUInt( $00000800);
16634: const 'BINDF_SILENTOPERATION', 'LongWord').SetUInt( $00001000);
16635: const 'BINDF_PRAGMA_NO_CACHE', 'LongWord').SetUInt( $00002000);
16636: const 'BINDF_FREE_THREADED', 'LongWord').SetUInt( $00010000);
16637: const 'BINDF_DIRECT_READ', 'LongWord').SetUInt( $00020000);
16638: const 'BINDF_FORMS_SUBMIT', 'LongWord').SetUInt( $00040000);
16639: const 'BINDF_GETFROMCACHE_IF_NET_FAIL', 'LongWord').SetUInt( $00080000);
16640: //const 'BINDF_DONTUSECACHE', '').SetString( BINDF_GETNEWESTVERSION);
16641: //const 'BINDF_DONTPUTINCACHE', '').SetString( BINDF_NOWRITECACHE);
16642: //const 'BINDF_NOCOPYDATA', '').SetString( BINDF_PULLDATA);
16643: const 'BSCF_FIRSTDATANOTIFICATION', 'LongWord').SetUInt( $00000001);
16644: const 'BSCF_INTERMEDIATEDATANOTIFICATION', 'LongWord').SetUInt( $00000002);
16645: const 'BSCF_LASTDATANOTIFICATION', 'LongWord').SetUInt( $00000004);
16646: const 'BSCF_DATAFULLYAVAILABLE', 'LongWord').SetUInt( $00000008);
16647: const 'BSCF_AVAILABLEDATASIZEUNKNOWN', 'LongWord').SetUInt( $00000010);
16648: const 'BINDSTATUS_FINDINGRESOURCE', 'LongInt').SetInt( 1);
16649: const 'BINDSTATUS_CONNECTING', 'LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16650: const 'BINDSTATUS_REDIRECTING', 'LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16651: const 'BINDSTATUS_BEGINDOWNLOADDATA', 'LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16652: const 'BINDSTATUS_DOWNLOADINGDATA', 'LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16653: const 'BINDSTATUS_ENDDOWNLOADDATA', 'LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16654: const BINDSTATUS_BEGINDOWNLOADCMPONENTS, LongInt).SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16655: const 'BINDSTATUS_INSTALLINGCOMPONENTS', LongInt).SetInt( BINDSTATUS_BEGINDOWNLOADCMPONENTS+1);
16656: const BINDSTATUS_ENDDOWNLOADCMPONENTS, LongInt).SetInt( BINDSTATUS_INSTALLINGCOMPONENTS+1);
16657: const BINDSTATUS_USINGCACHEDCOPY, LongInt).SetInt( BINDSTATUS_ENDDOWNLOADCMPONENTS + 1);
16658: const 'BINDSTATUS_SENDINGREQUEST', 'LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16659: const 'BINDSTATUS_CLASSIDAVAILABLE', 'LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16660: const 'BINDSTATUS_MIMETYPEAVAILABLE', 'LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16661: const BINDSTATUS_CACHEFILENAMEAVAILABLE, LongInt).SetInt( BINDSTATUS_MIMETYPEAVAILABLE+1);
16662: const 'BINDSTATUS_BEGINSYNCOOPERATION', 'LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE+1);
16663: const BINDSTATUS_ENDSYNCOOPERATION, LongInt).SetInt( BINDSTATUS_BEGINSYNCOOPERATION + 1);
16664: const 'BINDSTATUS_BEGINUPLOADDATA', 'LongInt').SetInt( BINDSTATUS_ENDSYNCOOPERATION + 1);
16665: const 'BINDSTATUS_UPLOADINGDATA', 'LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16666: const 'BINDSTATUS_ENDUPLOADDATA', 'LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16667: const 'BINDSTATUS_PROTOCOLCLASSID', 'LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16668: const 'BINDSTATUS_ENCODING', 'LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16669: const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE', 'LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16670: const BINDSTATUS_CLASSINSTALLLOCATION, LongInt).SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE+1);
16671: const 'BINDSTATUS_DECODING', 'LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16672: const 'BINDSTATUS_LOADINGMIMEHANDLER', 'LongInt').SetInt( BINDSTATUS_DECODING + 1);
16673: const BINDSTATUS_CONTENTDISPOSITIONATTACH, LongInt).SetInt( BINDSTATUS_LOADINGMIMEHANDLER+1);
16674: const 'BINDSTATUS_FILTERREPORTMIMETYPE', LongInt).SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH+1);
16675: const BINDSTATUS_CLSIDCANINstantiate, LongInt).SetInt( BINDSTATUS_FILTERREPORTMIMETYPE+ 1);
16676: const BINDSTATUS_IUNKNOWNAVAILABLE, LongInt).SetInt( BINDSTATUS_CLSIDCANINstantiate + 1);
16677: const 'BINDSTATUS_DIRECTBIND', 'LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16678: const 'BINDSTATUS_RAWMIMETYPE', 'LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16679: const 'BINDSTATUS_PROXYDETECTING', 'LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16680: const 'BINDSTATUS_ACCEPTRANGES', 'LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16681: const 'BINDSTATUS_COOKIE_SENT', 'LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16682: const 'BINDSTATUS_COMPACT_POLICY_RECEIVED', 'LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16683: const BINDSTATUS_COOKIE_SUPPRESSED, LongInt).SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED+1);
16684: const BINDSTATUS_COOKIE_STATE_UNKNOWN, LongInt).SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16685: const BINDSTATUS_COOKIE_STATE_ACCEPT, LongInt).SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN+ 1);
16686: const BINDSTATUS_COOKIE_STATE_REJECT, LongInt).SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16687: const BINDSTATUS_COOKIE_STATE_PROMPT, LongInt).SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16688: const BINDSTATUS_COOKIE_STATE_LEASH, LongInt).SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16689: const BINDSTATUS_COOKIE_STATE_DOWNGRADE, LongInt).SetInt( BINDSTATUS_COOKIE_STATE_LEASH+1);
16690: const 'BINDSTATUS_POLICY_HREF', 'LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16691: const 'BINDSTATUS_P3P_HEADER', 'LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16692: const 'BINDSTATUS_SESSION_COOKIE_RECEIVED', 'LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16693: const BINDSTATUS_PERSISTENT_COOKIE_RECEIVED, LongInt).SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED+1);
16694: const BINDSTATUS_SESSION_COOKIES_ALLOWED, LongInt).SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED+1);
16695: const BINDSTATUS_CACHECONTROL, LongInt).SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16696: const BINDSTATUS_CONTENTDISPOSITIONFILENAME, LongInt).SetInt( BINDSTATUS_CACHECONTROL + 1);
16697: const BINDSTATUS_MIMETEXTPLAINMISMATCH, 'LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME+1);
16698: const BINDSTATUS_PUBLISHERAVAILABLE, LongInt).SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH+1);
16699: const BINDSTATUS_DISPLAYNAMEAVAILABLE, LongInt).SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16700: // PBindInfo, 'TBindInfo // will not work;
16701: { tagBINDINFO, 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16702: + 'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16703: + 'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16704: + 'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16705: + 'UID; pUnk : IUnknown; dwReserved : DWORD; end;
16706: TBindInfo, 'tagBINDINFO;
16707: BINDINFO, 'tagBINDINFO;
16708: REMSECURITY_ATTRIBUTES, 'record nLength : DWORD; lpSecurityDes'
16709: + 'cripto : DWORD; bInheritHandle : BOOL; end;
16710: TRemSecurityAttributes, ' REMSECURITY_ATTRIBUTES;
16711: REMSECURITY_ATTRIBUTES, ' REMSECURITY_ATTRIBUTES;
16712: //PRemBindInfo, 'TRemBindInfo // will not work;
16713: { tagRemBINDINFO, 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16714: + 'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16715: + 'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16716: + ' ; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16717: + 'n; dwReserved : DWORD; end;
16718: TRemBindInfo, 'tagRemBINDINFO;
16719: RemBINDINFO, 'tagRemBINDINFO;
16720: //PRemFormatEtc, 'TRemFormatEtc // will not work;

```



```

16721: tagRemFORMATETC', 'record cffFormat:DWORD; ptd:DWORD; dwAspect:DWORD; lindex:Longint; tmed:DWORD; end;
16722: TRemFormatEtc', 'tagRemFORMATETC;
16723: RemFORMATETC', 'tagRemFORMATETC;
16724: SIRegister_IBindStatusCallback(CL);
16725: SIRegister_IAuthenticate(CL);
16726: SIRegister_IHttpNegotiate(CL);
16727: SIRegister_IWindowForBindingUI(CL);
16728: const 'CIP_DISK_FULL', 'LongInt').SetInt( 0);
16729: const 'CIP_ACCESS_DENIED', 'LongInt').SetInt( CIP_DISK_FULL + 1);
16730: const 'CIP_NEWER_VERSION_EXISTS', 'LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16731: const 'CIP_OLDER_VERSION_EXISTS', 'LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16732: const 'CIP_NAME_CONFLICT', 'LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16733: const CIP_TRUST_VERIFICATION_COMPONENT_MISSING, LongInt).SetInt( CIP_NAME_CONFLICT + 1);
16734: const CIP_EXE_SELF_REGISTRATION_TIMEOUT, LongInt).SetInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING+1);
16735: const 'CIP_UNSAFE_TO_ABORT', 'LongInt').SetInt( CIP_EXE_SELF_REGISTRATION_TIMEOUT + 1);
16736: const 'CIP_NEED_REBOOT', 'LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16737: const 'CIP_NEED_REBOOT_UI_PERMISSION', 'LongInt').SetInt( CIP_NEED_REBOOT + 1);
16738: SIRegister_ICodeInstall(CL);
16739: SIRegister_IWinInetInfo(CL);
16740: const 'WININETINFO_OPTION_LOCK_HANDLE', 'LongInt').SetInt( 65534);
16741: SIRegister_IHttpSecurity(CL);
16742: SIRegister_IWinInetHttpInfo(CL);
16743: SIRegister_IBindHost(CL);
16744: const 'URLOSTRM_USECACHEDCOPY_ONLY', 'LongWord').SetUInt( $00000001);
16745: const 'URLOSTRM_USECACHEDCOPY', 'LongWord').SetUInt( $00000002);
16746: const 'URLOSTRM_GETNEWESTVERSION', 'LongWord').SetUInt( $00000003);
16747: Func URLOpenStream(p1:IUnknown; p2:PChar; p3:DWORD; p4:IBindStatusCallback): HRESULT;
16748: Func URLOpenPullStream(p1:IUnknown; p2:PChar; p3:DWORD; BSC:IBindStatusCallback): HRESULT;
16749: Func
URLDownloadToFile( Caller:IUnknown; URL:PChar; FileName:PChar; Reserved:DWORD; StatusCB:IBindStatusCallback ):
HRESULT;
16750: Func URLDownloadToCacheFile( p1:IUnknown; p2:PChar; p3:PChar; p4:DWORD; p5:DWORD; p6:IBindStatusCallback ):
HRESULT;
16751: Func URLOpenBlockingStream( p1:IUnknown; p2:PChar; out p3:IStream; p4:DWORD; p5:IBindStatusCallback ): HRESULT;
16752: Func HlinkGoBack( unk : IUnknown ) : HRESULT;
16753: Func HlinkGoForward( unk : IUnknown ) : HRESULT;
16754: Func HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR ) : HRESULT;
16755: // Func HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker ) : HRESULT;
16756: SIRegister_IInternet(CL);
16757: const 'BINDSTRING_HEADERS', 'LongInt').SetInt( 1);
16758: const 'BINDSTRING_ACCEPT_MIMES', 'LongInt').SetInt( BINDSTRING_HEADERS + 1);
16759: const 'BINDSTRING_EXTRA_URL', 'LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16760: const 'BINDSTRING_LANGUAGE', 'LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16761: const 'BINDSTRING_USERNAME', 'LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16762: const 'BINDSTRING_PASSWORD', 'LongInt').SetInt( BINDSTRING_USERNAME + 1);
16763: const 'BINDSTRING_UA_PIXELS', 'LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16764: const 'BINDSTRING_UA_COLOR', 'LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16765: const 'BINDSTRING_OS', 'LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16766: const 'BINDSTRING_USER_AGENT', 'LongInt').SetInt( BINDSTRING_OS + 1);
16767: const 'BINDSTRING_ACCEPT_ENCODINGS', 'LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16768: const 'BINDSTRING_POST_COOKIE', 'LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16769: const 'BINDSTRING_POST_DATA_MIME', 'LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16770: const 'BINDSTRING_URL', 'LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16771: //POLESTArray, '^TOLESTArray // will not work;
16772: SIRegister_IInternetBindInfo(CL);
16773: const 'PI_PARSE_URL', 'LongWord').SetUInt( $00000001);
16774: const 'PI_FILTER_MODE', 'LongWord').SetUInt( $00000002);
16775: const 'PI_FORCE_ASYNC', 'LongWord').SetUInt( $00000004);
16776: const 'PI_USE_WORKERTHREAD', 'LongWord').SetUInt( $00000008);
16777: const 'PI_MIMEVERIFICATION', 'LongWord').SetUInt( $00000010);
16778: const 'PI_CLSIDLOOKUP', 'LongWord').SetUInt( $00000020);
16779: const 'PI_DATAPROGRESS', 'LongWord').SetUInt( $00000040);
16780: const 'PI_SYNCHRONOUS', 'LongWord').SetUInt( $00000080);
16781: const 'PI_APARTMENTTHREADED', 'LongWord').SetUInt( $00000100);
16782: const 'PI_CLASSINSTALL', 'LongWord').SetUInt( $00000200);
16783: const 'PD_FORCE_SWITCH', 'LongWord').SetUInt( $00010000);
16784: //const 'PI_DOCFILECLSIDLOOKUP', 'LongWord').SetUInt( $00010000);
16785: //PProtocolData, 'TProtocolData // will not work;
16786: _tagPROTOCOLDATA, record grfFlags:DWORD; dwState:DWORD; pData:TObject; cbData:ULONG; end;
16787: TProtocolData, 'tagPROTOCOLDATA;
16788: PROTOCOLDATA, 'tagPROTOCOLDATA;
16789: Interface(CL.FindInterface('IUNKNOWN'), IInternetProtocolSink, 'IInternetProtocolSink;
16790: SIRegister_IInternetProtocolRoot(CL);
16791: SIRegister_IInternetProtocol(CL);
16792: SIRegister_IInternetProtocolSink(CL);
16793: const 'OIBDG_APARTMENTTHREADED', 'LongWord').SetUInt( $00000100);
16794: SIRegister_IInternetSession(CL);
16795: SIRegister_IInternetThreadSwitch(CL);
16796: SIRegister_IInternetPriority(CL);
16797: const 'PARSE_CANONICALIZE', 'LongInt').SetInt( 1);
16798: const 'PARSE_FRIENDLY', 'LongInt').SetInt( PARSE_CANONICALIZE + 1);
16799: const 'PARSE_SECURITY_URL', 'LongInt').SetInt( PARSE_FRIENDLY + 1);
16800: const 'PARSE_ROOTDOCUMENT', 'LongInt').SetInt( PARSE_SECURITY_URL + 1);
16801: const 'PARSE_DOCUMENT', 'LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16802: const 'PARSE_ANCHOR', 'LongInt').SetInt( PARSE_DOCUMENT + 1);
16803: const 'PARSE_ENCODE', 'LongInt').SetInt( PARSE_ANCHOR + 1);
16804: const 'PARSE_DECODE', 'LongInt').SetInt( PARSE_ENCODE + 1);
16805: const 'PARSE_PATH_FROM_URL', 'LongInt').SetInt( PARSE_DECODE + 1);
16806: const 'PARSE_URL_FROM_PATH', 'LongInt').SetInt( PARSE_PATH_FROM_URL + 1);

```



```

16807: const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16808: const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16809: const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16810: const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16811: const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16812: const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16813: const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16814: const 'PSU_DEFAULT','LongInt').SetInt( 1);
16815: const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16816: const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16817: const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16818: const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16819: const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16820: const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16821: const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16822: const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16823: const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16824: const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16825: const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16826: const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16827: const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16828: const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16829: const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16830: SIRegister_IInternetProtocolInfo(CI);
16831: IOInet', 'IInternet;
16832: IOInetBindInfo', 'IInternetBindInfo;
16833: IOInetProtocolRoot', 'IInternetProtocolRoot;
16834: IOInetProtocol', 'IInternetProtocol;
16835: IOInetProtocolSink', 'IInternetProtocolSink;
16836: IOInetProtocolInfo', 'IInternetProtocolInfo;
16837: IOInetSession', 'IInternetSession;
16838: IOInetPriority', 'IInternetPriority;
16839: IOInetThreadSwitch', 'IInternetThreadSwitch;
16840: Func
CoInternetParseUrl(pwzUrl:LPCWSTR;ParseAction:TParseAction;dwFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD;
var pcchResult:DWORD;dwReserved:DWORD):HResult;
16841: Func CoInternetCombineUrl(pwzBaseUrl,
pwzRelativeUrl:LPCWSTR;dwCombineFlags:DWORD;pszResult:LPWSTR;cchResult: DWORD;var
pcchResult:DWORD;dwReserved:DWORD):HResult;
16842: Func CoInternetCompareUrl(pwzUrl1, pwzUrl2 LPCWSTR;dwFlags:DWORD): HResult;
16843: Func CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags:DWORD;dwReserved:DWORD):HResult;
16844: Func
CoInternetQueryInfo(pwzUrl:LPCWSTR;QueryOptions:TQueryOption;dwQueryFlags:DWORD;pvBuffer:TObject;cbBuffer
:DWORD;var pcbBuffer:DWORD;dwReserved:DWORD):HResult;
16845: Func CoInternetGetSession(dwSessionMode:DWORD;var
pIInternetSession:IInternetSes;dwReserved:DWORD):HResult;
16846: Func CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16847: Func
OInetParseUrl(pwzUrl:LPCWSTR;ParseAction:TParseAction;dwFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD; var
pcchResult:DWORD;dwReserved:DWORD):HResult;
16848: Func OInetCombineUrl(pwzBaseUrl,
pwzRelativeUrl:LPCWSTR;dwCombineFlags:DWORD;pszResult:LPWSTR;cchResult:DWORD;var
pcchResult:DWORD;dwReserved:DWORD):HResult;
16849: Func OInetCompareUrl(pwzUrl1,pwzUrl2 LPCWSTR;dwFlags: DWORD): Hresult;
16850: Func OInetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags:DWORD;dwReserved:DWORD): HResult;
16851: Func
OInetQueryInfo(pwzUrl:LPCWSTR;QueryOptions:TQueryOption;dwQueryFlags:DWORD;pvBuffer:TObject;cbBuffer: DWORD;var
pcbBuffer:DWORD;dwReserved:DWORD):HResult;
16852: Func OInetGetSession(dwSessionMode:DWORD;var pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16853: //Func CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult;
16854: //Proc ReleaseBindInfo( const bindinfo : TBindInfo);
16855: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16856: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16857: //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16858: //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16859: //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16860: const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16861: SIRegister_IInternetSecurityMgrSite(CI);
16862: const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16863: const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16864: const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16865: const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16866: const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16867: const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16868: const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16869: const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16870: const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16871: const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16872: const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16873: const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16874: const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16875: const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16876: const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16877: const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16878: const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16879: const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16880: const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16881: const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);

```

```

16882: const 'PUAF_DEFAULTZONEPOL', 'LongWord').SetUInt( $00040000);
16883: const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED', 'LongWord').SetUInt( $00080000);
16884: const 'PUAF_NOUIIFLOCKED', 'LongWord').SetUInt( $00100000);
16885: const 'PUAFOUT_DEFAULT', 'LongWord').SetUInt( $0);
16886: const 'PUAFOUT_ISLOCKZONEPOLICY', 'LongWord').SetUInt( $1);
16887: const 'SZM_CREATE', 'LongWord').SetUInt( $00000000);
16888: const 'SZM_DELETE', 'LongWord').SetUInt( $00000001);
16889: SIRegister_IInternetSecurityManager(CL);
16890: SIRegister_IInternetHostSecurityManager(CL);
16891: SIRegister_IInternetSecurityManagerEx(CL);
16892: const 'URLACTION_MIN', 'LongWord').SetUInt( $00001000);
16893: const 'URLACTION_DOWNLOAD_MIN', 'LongWord').SetUInt( $00001000);
16894: const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX', 'LongWord').SetUInt( $00001001);
16895: const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX', 'LongWord').SetUInt( $00001004);
16896: const 'URLACTION_DOWNLOAD_CURR_MAX', 'LongWord').SetUInt( $00001004);
16897: const 'URLACTION_DOWNLOAD_MAX', 'LongWord').SetUInt( $000011FF);
16898: const 'URLACTION_ACTIVEX_MIN', 'LongWord').SetUInt( $00001200);
16899: const 'URLACTION_ACTIVEX_RUN', 'LongWord').SetUInt( $00001200);
16900: const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY', 'LongWord').SetUInt( $00001201);
16901: const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY', 'LongWord').SetUInt( $00001202);
16902: const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY', 'LongWord').SetUInt( $00001203);
16903: const 'URLACTION_SCRIPT_OVERRIDE_SAFETY', 'LongWord').SetUInt( $00001401);
16904: const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY', 'LongWord').SetUInt( $00001204);
16905: const 'URLACTION_ACTIVEX_TREATASUNTRUSTED', 'LongWord').SetUInt( $00001205);
16906: const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT', 'LongWord').SetUInt( $00001206);
16907: const 'URLACTION_ACTIVEX_CURR_MAX', 'LongWord').SetUInt( $00001206);
16908: const 'URLACTION_ACTIVEX_MAX', 'LongWord').SetUInt( $000013FF);
16909: const 'URLACTION_SCRIPT_MIN', 'LongWord').SetUInt( $00001400);
16910: const 'URLACTION_SCRIPT_RUN', 'LongWord').SetUInt( $00001400);
16911: const 'URLACTION_SCRIPT_JAVA_USE', 'LongWord').SetUInt( $00001402);
16912: const 'URLACTION_SCRIPT_SAFE_ACTIVEX', 'LongWord').SetUInt( $00001405);
16913: const 'URLACTION_SCRIPT_CURR_MAX', 'LongWord').SetUInt( $00001405);
16914: const 'URLACTION_SCRIPT_MAX', 'LongWord').SetUInt( $000015FF);
16915: const 'URLACTION_HTML_MIN', 'LongWord').SetUInt( $00001600);
16916: const 'URLACTION_HTML_SUBMIT_FORMS', 'LongWord').SetUInt( $00001601);
16917: const 'URLACTION_HTML_SUBMIT_FORMS_FROM', 'LongWord').SetUInt( $00001602);
16918: const 'URLACTION_HTML_SUBMIT_FORMS_TO', 'LongWord').SetUInt( $00001603);
16919: const 'URLACTION_HTML_FONT_DOWNLOAD', 'LongWord').SetUInt( $00001604);
16920: const 'URLACTION_HTML_JAVA_RUN', 'LongWord').SetUInt( $00001605);
16921: const 'URLACTION_HTML_CURR_MAX', 'LongWord').SetUInt( $00001605);
16922: const 'URLACTION_HTML_MAX', 'LongWord').SetUInt( $000017FF);
16923: const 'URLACTION_SHELL_MIN', 'LongWord').SetUInt( $00001800);
16924: const 'URLACTION_SHELL_INSTALL_DTITEMS', 'LongWord').SetUInt( $00001800);
16925: const 'URLACTION_SHELL_MOVE_OR_COPY', 'LongWord').SetUInt( $00001802);
16926: const 'URLACTION_SHELL_FILE_DOWNLOAD', 'LongWord').SetUInt( $00001803);
16927: const 'URLACTION_SHELL_VERB', 'LongWord').SetUInt( $00001804);
16928: const 'URLACTION_SHELL_WEBVIEW_VERB', 'LongWord').SetUInt( $00001805);
16929: const 'URLACTION_SHELL_SHELLEXECUTE', 'LongWord').SetUInt( $00001806);
16930: const 'URLACTION_SHELL_EXECUTE_HIGHRISK', 'LongWord').SetUInt( $00001806);
16931: const 'URLACTION_SHELL_EXECUTE_MODRISK', 'LongWord').SetUInt( $00001807);
16932: const 'URLACTION_SHELL_EXECUTE_LOWRISK', 'LongWord').SetUInt( $00001808);
16933: const 'URLACTION_SHELL_POPUPMGR', 'LongWord').SetUInt( $00001809);
16934: const 'URLACTION_SHELL_CURR_MAX', 'LongWord').SetUInt( $00001809);
16935: const 'URLACTION_SHELL_MAX', 'LongWord').SetUInt( $000019ff);
16936: const 'URLACTION_NETWORK_MIN', 'LongWord').SetUInt( $00001A00);
16937: const 'URLACTION_CREDENTIALS_USE', 'LongWord').SetUInt( $00001A00);
16938: const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK', 'LongWord').SetUInt( $00000000);
16939: const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER', 'LongWord').SetUInt( $00010000);
16940: const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT', 'LongWord').SetUInt( $00020000);
16941: const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY', 'LongWord').SetUInt( $00030000);
16942: const 'URLACTION_AUTHENTICATE_CLIENT', 'LongWord').SetUInt( $00001A01);
16943: const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK', 'LongWord').SetUInt( $00000000);
16944: const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE', 'LongWord').SetUInt( $00010000);
16945: const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY', 'LongWord').SetUInt( $00030000);
16946: const 'URLACTION_NETWORK_CURR_MAX', 'LongWord').SetUInt( $00001A01);
16947: const 'URLACTION_NETWORK_MAX', 'LongWord').SetUInt( $00001BFF);
16948: const 'URLACTION_JAVA_MIN', 'LongWord').SetUInt( $00001C00);
16949: const 'URLACTION_JAVA_PERMISSIONS', 'LongWord').SetUInt( $00001C00);
16950: const 'URLPOLICY_JAVA_PROHIBIT', 'LongWord').SetUInt( $00000000);
16951: const 'URLPOLICY_JAVA_HIGH', 'LongWord').SetUInt( $00010000);
16952: const 'URLPOLICY_JAVA_MEDIUM', 'LongWord').SetUInt( $00020000);
16953: const 'URLPOLICY_JAVA_LOW', 'LongWord').SetUInt( $00030000);
16954: const 'URLPOLICY_JAVA_CUSTOM', 'LongWord').SetUInt( $00800000);
16955: const 'URLACTION_JAVA_CURR_MAX', 'LongWord').SetUInt( $00001C00);
16956: const 'URLACTION_JAVA_MAX', 'LongWord').SetUInt( $00001CFF);
16957: const 'URLACTION_INFODELIVERY_MIN', 'LongWord').SetUInt( $00001D00);
16958: const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS', 'LongWord').SetUInt( $00001D00);
16959: const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS', 'LongWord').SetUInt( $00001D01);
16960: const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS', 'LongWord').SetUInt( $00001D02);
16961: const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D03);
16962: const 'URLACTION_INFODELIVERY_NO_EDITING SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D04);
16963: const 'URLACTION_INFODELIVERY_NO_REMOVING SUBSCRIPTIONS', 'LongWord').SetUInt( $00001D05);
16964: const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING', 'LongWord').SetUInt( $00001D06);
16965: const 'URLACTION_INFODELIVERY_CURR_MAX', 'LongWord').SetUInt( $00001D06);
16966: const 'URLACTION_INFODELIVERY_MAX', 'LongWord').SetUInt( $00001Dff);
16967: const 'URLACTION_CHANNEL_SOFTDIST_MIN', 'LongWord').SetUInt( $00001E00);
16968: const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS', 'LongWord').SetUInt( $00001E05);
16969: const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT', 'LongWord').SetUInt( $00010000);
16970: const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE', 'LongWord').SetUInt( $00020000);

```

```

16971: const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL', 'LongWord').SetUInt( $00030000);
16972: const 'URLACTION_CHANNEL_SOFTDIST_MAX', 'LongWord').SetUInt( $00001EFF);
16973: const 'URLACTION_BEHAVIOR_MIN', 'LongWord').SetUInt( $00002000);
16974: const 'URLACTION_BEHAVIOR_RUN', 'LongWord').SetUInt( $00002000);
16975: const 'URLPOLICY_BEHAVIOR_CHECK_LIST', 'LongWord').SetUInt( $00010000);
16976: const 'URLACTION_FEATURE_MIN', 'LongWord').SetUInt( $00002100);
16977: const 'URLACTION_FEATURE_MIME_SNIFFING', 'LongWord').SetUInt( $00002100);
16978: const 'URLACTION_FEATURE_ZONE_ELEVATION', 'LongWord').SetUInt( $00002101);
16979: const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS', 'LongWord').SetUInt( $00002102);
16980: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN', 'LongWord').SetUInt( $00002200);
16981: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI', 'LongWord').SetUInt( $00002200);
16982: const 'URLACTION_AUTOMATIC_ACTIVEX_UI', 'LongWord').SetUInt( $00002201);
16983: const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS', 'LongWord').SetUInt( $00002300);
16984: const 'URLPOLICY_ALLOW', 'LongWord').SetUInt( $00);
16985: const 'URLPOLICY_QUERY', 'LongWord').SetUInt( $01);
16986: const 'URLPOLICY_DISALLOW', 'LongWord').SetUInt( $03);
16987: const 'URLPOLICY_NOTIFY_ON_ALLOW', 'LongWord').SetUInt( $10);
16988: const 'URLPOLICY_NOTIFY_ON_DISALLOW', 'LongWord').SetUInt( $20);
16989: const 'URLPOLICY_LOG_ON_ALLOW', 'LongWord').SetUInt( $40);
16990: const 'URLPOLICY_LOG_ON_DISALLOW', 'LongWord').SetUInt( $80);
16991: const 'URLPOLICY_MASK_PERMISSIONS', 'LongWord').SetUInt( $0f);
16992: Func GetUrlPolicyPermissions( dw : DWORD) : DWORD;
16993: Func SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD;
16994: const 'URLZONE_PREDEFINED_MIN', 'LongInt').SetInt( 0);
16995: const 'URLZONE_LOCAL_MACHINE', 'LongInt').SetInt( 0);
16996: const 'URLZONE_INTRANET', 'LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16997: const 'URLZONE_TRUSTED', 'LongInt').SetInt( URLZONE_INTRANET + 1);
16998: const 'URLZONE_INTERNET', 'LongInt').SetInt( URLZONE_TRUSTED + 1);
16999: const 'URLZONE_UNTRUSTED', 'LongInt').SetInt( URLZONE_INTERNET + 1);
17000: const 'URLZONE_PREDEFINED_MAX', 'LongInt').SetInt( 999);
17001: const 'URLZONE_USER_MIN', 'LongInt').SetInt( 1000);
17002: const 'URLZONE_USER_MAX', 'LongInt').SetInt( 10000);
17003: const 'URLTEMPLATE_CUSTOM', 'LongWord').SetUInt( $00000000);
17004: const 'URLTEMPLATE_PREDEFINED_MIN', 'LongWord').SetUInt( $00010000);
17005: const 'URLTEMPLATE_LOW', 'LongWord').SetUInt( $00010000);
17006: const 'URLTEMPLATE_MEDIUM', 'LongWord').SetUInt( $00011000);
17007: const 'URLTEMPLATE_HIGH', 'LongWord').SetUInt( $00012000);
17008: const 'URLTEMPLATE_PREDEFINED_MAX', 'LongWord').SetUInt( $00020000);
17009: const 'MAX_ZONE_PATH', 'LongInt').SetInt( 260);
17010: const 'MAX_ZONE_DESCRIPTION', 'LongInt').SetInt( 200);
17011: const 'ZAFLAGS_CUSTOM_EDIT', 'LongWord').SetUInt( $00000001);
17012: const 'ZAFLAGS_ADD_SITES', 'LongWord').SetUInt( $00000002);
17013: const 'ZAFLAGS_REQUIRE_VERIFICATION', 'LongWord').SetUInt( $00000004);
17014: const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE', 'LongWord').SetUInt( $00000008);
17015: const 'ZAFLAGS_INCLUDE_INTRANET_SITES', 'LongWord').SetUInt( $00000010);
17016: const 'ZAFLAGS_NO_UI', 'LongWord').SetUInt( $00000020);
17017: const 'ZAFLAGS_SUPPORTS_VERIFICATION', 'LongWord').SetUInt( $00000040);
17018: const 'ZAFLAGS_UNC_AS_INTRANET', 'LongWord').SetUInt( $00000080);
17019: const 'ZAFLAGS_USE_LOCKED_ZONES', 'LongWord').SetUInt( $00010000);
17020: //PZoneAttributes, '^TZoneAttributes // will not work;
17021: _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;
17022: { _ZONEATTRIBUTES = packed record
17023:   cbSize: ULONG;
17024:   szDisplayName: array [0..260 - 1] of WideChar;
17025:   szDescription: array [0..200 - 1] of WideChar;
17026:   szIconPath: array [0..260 - 1] of WideChar;
17027:   dwTemplateMinLevel: DWORD;
17028:   dwTemplateRecommended: DWORD;
17029:   dwTemplateCurrentLevel: DWORD;
17030:   dwFlags: DWORD;
17031: end; }
17032: TZoneAttributes, '_ZONEATTRIBUTES;
17033: _ZONEATTRIBUTES', '_ZONEATTRIBUTES;
17034: const 'URLZONEREG_DEFAULT', 'LongInt').SetInt( 0);
17035: const 'URLZONEREG_HKLM', 'LongInt').SetInt( URLZONEREG_DEFAULT + 1);
17036: const 'URLZONEREG_HKCU', 'LongInt').SetInt( URLZONEREG_HKLM + 1);
17037: SIRegister_IInternetZoneManager(CL);
17038: SIRegister_IInternetZoneManagerEx(CL);
17039: const 'SOFTDIST_FLAG_USAGE_EMAIL', 'LongWord').SetUInt( $00000001);
17040: const 'SOFTDIST_FLAG_USAGE_PRECACHE', 'LongWord').SetUInt( $00000002);
17041: const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL', 'LongWord').SetUInt( $00000004);
17042: const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION', 'LongWord').SetUInt( $00000008);
17043: const 'SOFTDIST_ADSTATE_NONE', 'LongWord').SetUInt( $00000000);
17044: const 'SOFTDIST_ADSTATE_AVAILABLE', 'LongWord').SetUInt( $00000001);
17045: const 'SOFTDIST_ADSTATE_DOWNLOADED', 'LongWord').SetUInt( $00000002);
17046: const 'SOFTDIST_ADSTATE_INSTALLED', 'LongWord').SetUInt( $00000003);
17047: //PCodeBaseHold, '^TCodeBaseHold // will not work;
17048: _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
17049: + 'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle: DWORD; end;
17050: TCodeBaseHold, '_tagCODEBASEHOLD;
17051: CODEBASEHOLD', '_tagCODEBASEHOLD;
17052: //PSoftDistInfo, '^TSoftDistInfo // will not work;
17053: _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
17054: + 'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
17055: + 'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
17056: + ' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
17057: + 'rtisedVersionLS : DWORD; dwReserved : DWORD; end;

```

```

17058:   TSoftDistInfo', '_tagSOFTDISTINFO;
17059:   SOFTDISTINFO', '_tagSOFTDISTINFO;
17060:   SIRegister_ISoftDistExt(CL);
17061:   Func GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HRESULT;
17062:   Func SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
dwAdvertisedVersionLS : DWORD) : HRESULT;
17063:   SIRegister_IDataFilter(CL);
17064:   // PProtocolFilterData', '^TProtocolFilterData // will not work;
17065:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
17066:   +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
17067:   +'terFlags : DWORD; end;
17068:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA;
17069:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA;
17070:   //PDataInfo', '^TDataInfo // will not work;
17071:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrpaketSize : UL'
17072:   +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end;
17073:   TDataInfo', '_tagDATAINFO;
17074:   DATAINFO', '_tagDATAINFO;
17075:   SIRegister_IEncodingFilterFactory(CL);
17076:   Func IsLoggingEnabled( pszUrl : PChar) : BOOL;
17077:   //Func IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL;
17078:   //Func IsLoggingEnabledW( pszUrl : PWideChar) : BOOL;
17079:   //PHitLoggingInfo', '^THitLoggingInfo // will not work;
17080:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
17081:   +'rlName:LPSTR;StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo:LPSTR; end;
17082:   THitLoggingInfo', '_tagHIT_LOGGING_INFO;
17083:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO;
17084:   Func WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL;
17085: end;
17086:
17087: Proc SIRegister_DFFUtils(CL: TPSPascalCompiler);
17088: begin
17089:   Proc reformatMemo( const m : TCustomMemo);
17090:   Proc SetMemoMargins( m : TCustomMemo; const L, T, R, B : Int);
17091:   Proc MoveToTop( memo : TMemo);
17092:   Proc ScrollToTop( memo : TMemo);
17093:   Func LineNumberClicked( memo : TMemo) : Int;
17094:   Func MemoClickedLine( memo : TMemo) : Int;
17095:   Func ClickedMemoLine( memo : TMemo) : Int;
17096:   Func MemoLineClicked( memo : TMemo) : Int;
17097:   Func LinePositionClicked( Memo : TMemo) : Int;
17098:   Func ClickedMemoPosition( memo : TMemo) : Int;
17099:   Func MemoPositionClicked( memo : TMemo) : Int;
17100:   Proc AdjustGridSize( grid : TDrawGrid);
17101:   Proc DeleteGridRow( Grid : TStringGrid; const ARow : Int);
17102:   Proc InsertgridRow( Grid : TStringGrid; const ARow : Int);
17103:   Proc Sortgrid( Grid : TStringGrid; const SortCol : Int);;
17104:   Proc Sortgridl(Grid:TStringGrid; const SortCol : Int; sortascending :Bool);;
17105:   Proc sortstrDown( var s :Str);
17106:   Proc sortstrUp( var s :Str);
17107:   Proc rotatestrleft( var s :Str);
17108:   Func dffstrtofloatdef( s :Str; default : extended) : extended;
17109:   Func deblank( s :Str);
17110:   Func IntToBinaryString( const n : Int; MinLength : Int) :Str;
17111:   Proc FreeAndClearListBox( C : TListBox);;
17112:   Proc FreeAndClearMemo( C : TMemo);;
17113:   Proc FreeAndClearStringList( C : TStringList);;
17114:   Func dffgetfilesize( f : TSearchrec) : int64;
17115: end;
17116:
17117: Proc SIRegister_MathsLib(CL: TPSPascalCompiler);
17118: begin
17119:   TypeS('intset', set of byte;
17120:   TPoint64', 'record x : int64; y : int64; end;
17121:   Func GetNextPandigital( size : Int; var Digits : array of Int) :Bool;
17122:   Func IsPolygonal( T : int64; var rank : array of Int) :Bool;
17123:   Func GeneratePentagon( n : Int) : Int;
17124:   Func IsPentagon( p : Int) :Bool;
17125:   Func isSquare( const N : int64) :Bool;
17126:   Func isCube( const N : int64) :Bool;
17127:   Func isPalindrome( const n : int64) :Bool;;
17128:   Func isPalindromel( const n : int64; var len : Int) :Bool;;
17129:   Func GetEulerPhi( n : int64) : int64;
17130:   Func dffIntPower( a, b : int64) : int64;;
17131:   Func IntPowerl( a : extended; b : int64) : extended;;
17132:   Func gcd2( a, b : int64) : int64;
17133:   Func GCDMany( A : array of Int) : Int;
17134:   Func LCMMMany( A : array of Int) : Int;
17135:   Proc ContinuedFraction(A: array of int64;const wholepart:int;var numerator,denominator:int64);
17136:   Func dffFactorial( n : int64) : int64;
17137:   Func digitcount( n : int64) : Int;
17138:   Func nextpermute( var a : array of Int) :Bool;
17139:   Func convertfloattofractionstring(N:extended;maxdenom:Int;multipleof:boolean):str;
17140:   Func convertStringToDecimal( s :Str; var n : extended) :Bool;
17141:   Func InttoBinaryStr( nn : Int) :Str;
17142:   Func StrtoAngle( const s :Str; var angle : extended) :Bool;
17143:   Func AngleToStr( angle : extended) :Str;
17144:   Func AngleToStr2( angle : extended; decimal:boolean) :Str;
17145:   Func deg2rad( deg : extended) : extended;

```



```

17146: Func rad2deg( rad : extended) : extended;
17147: Func GetLongToMercProjection( const long : extended) : extended;
17148: Func GetLatToMercProjection( const Lat : Extended) : Extended;
17149: Func GetMercProjectionToLong( const ProjLong : extended) : extended;
17150: Func GetMercProjectionToLat( const ProjLat : extended) : extended;
17151: SIRegister_TPrimes(CL);
17152: //RIRegister_TPrimes(CL);
17153: //ConstantN('deg','LongInt').SetInt( char( 176));
17154: ConstantN('minmark','LongInt').SetInt(( 180));
17155: Func Random64( const N : Int64) : Int64;;
17156: Proc Randomize64;
17157: Func Random641 : extended;;
17158: Proc GetParens( Variables:Str; OpChar:char; var list:TStringlist)//DFUtils
17159: add 4.7.4.62
17160: Func MercScaling(Long0,lat0, long1, lat1:extended; x0,y0,x1,y1:integer) :TMercScalingRec';
17161: Func LonglatToPlotPt(Long,lat:Extended; Scalerec:TMercScalingrec):TPoint';
17162: Func PlotPtToLonglat(PlotPt:TPoint; Scalerec:TMercScalingrec):TRealPoint';
17163: Func SphericalEarthDistance(lat1,lon1,lat2,lon2:extended; Units:integer):extended';
17164: end;
17165:
17166: Proc SIRegister_UGeometry(CL: TPSPascalCompiler);
17167: begin
17168:   TrealPoint', 'record x : extended; y : extended; end;
17169:   Tline', 'record p1 : TPoint; p2 : TPoint; end;
17170:   TRealline', 'record p1 : TRealPoint; p2 : TRealPoint; end;
17171:   TCircle', 'record cx : Int; cy : Int; r : Int; end;
17172:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end;
17173:   PResult', '( PPOutside, PPInside, PPVertex, PPEdge, PPErr )';
17174: Func realpoint( x, y : extended) : TRealPoint;
17175: Func dist( const p1, p2 : TrealPoint) : extended;
17176: Func intdist( const p1, p2 : TPoint) : Int;
17177: Func dffLine( const p1, p2 : TPoint) : Tline;;
17178: Func Line1( const p1, p2 : TRealPoint) : TRealline;;
17179: Func dffCircle( const cx, cy, R : Int) : TCircle;;
17180: Func Circle1( const cx, cy, R : extended) : TRealCircle;;
17181: Func GetTheta( const L : TLine) : extended;;
17182: Func GetThetal( const p1, p2 : TPoint) : extended;;
17183: Func GetTheta2( const p1, p2 : TRealPoint) : extended;;
17184: Proc Extendline( var L : TLine; dist : Int);
17185: Proc Extendline1( var L : TRealLine; dist : extended);
17186: Func Linesintersect( line1, line2 : TLine) :Bool;
17187: Func ExtendedLinesIntersect( Line1,Line2:TLine;const extendlines:bool;var IP:TPoint):bool;
17188: Func ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
17189: Func Intersect(L1,L2: TLine; var pointonborder:boolean; var IP : TPoint) :Bool;
17190: Func PointPerpendicularLine( L : TLine; P : TPoint) : TLine;
17191: Func PerpDistance( L : TLine; P : TPoint) : Int;
17192: Func AngledLineFromLine(L:TLine;P:TPoint;Dist:extended; alpha : extended) : TLine;;
17193: Func AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
17194: Func PointInPoly( const p : TPoint; Points : array of TPoint) : PResult;
17195: Func PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var Clockwise:bool):Int;
17196: Proc InflatePolygon(const points:array of Tpoint;var points2:array of TPoint;var area:Int;const
screenCoordinates:bool; const inflateby : Int);
17197: Func PolyBuiltClockwise(const points:array of TPoint;const screencoordies:bool):bool;
17198: Func DegtoRad( d : extended) : extended;
17199: Func RadtoDeg( r : extended) : extended;
17200: Proc TranslateLeftTo( var L : TLine; newend : TPoint);
17201: Proc TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);
17202: Proc RotateRightEndBy( var L : TLine; alpha : extended);
17203: Proc RotateRightEndTo( var L : TLine; alpha : extended);
17204: Proc RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);
17205: Func CircleCircleIntersect( c1, c2 : TCircle; var IP1, IP2 : TPoint) :Bool;;
17206: Func CircleCircleIntersect1(c1,c2:TRealCircle; var IP1,IP2 TRealPoint) :Bool;;
17207: Func PointCircleTangentLines(const C:TCircle;const P:TPoint;var L1,L2:TLine):bool;
17208: Func CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,TL2:TLine):Bool;
17209: end;
17210:
17211:
17212: Proc SIRegister_UAstronomy(CL: TPSPascalCompiler);
17213: begin
17214:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat );
17215:   TDTType', '( ttLocal, ttUT, ttGST, ttLST );
17216:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end;
17217:   TSunrec', 'record TrueEclLon:extended;
AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl:TRPoint;
TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
17218:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl : '
+ TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
17219:   +arth : extended; Phase : extended; end;
17220:   TMoonEclipseAdd', 'record UmbralStartTime : TDateTime; UmbralEnd'
+Time : TDateTime; TotalStartTime : TDateTime; TotalEndTime : TDateTime; end;
17221:   TEclipseRec', 'record Msg :Str; Status : Int; FirstConta'
+ct: TDateTime;LastContact:TDateTime;Magnitude:Extended;MaxeclipseUTime:TDateTime;end;
17222:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO );
17223:   TPlanetRec', 'record AsOf : TDateTime; Name :Str; MeanLon : '
+extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
17224:   +ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
17225:   +jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end;
17226:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
extended;
UncorrectedEarthDistance:extended;GeoEclLonLat:TRpoint;CorrectedEarthDistance:extended;ApparentRADecl:TRPoint;end;

```



```

17231:   SIRegister_TAstronomy(CL);
17232: Func AngleToStr( angle : extended) :Str;
17233: Func AngleToStr2( angle : extended; decimal:boolean) :Str;
17234: Func StrToAngle( s :Str; var angle : extended) :Bool;
17235: Func HoursToStr24( t : extended) :Str;
17236: Func RPoint( x, y : extended) : TRPoint;
17237: Func getStimename( t : TDTType) :Str;
17238: end;
17239:
17240: Proc SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
17241: begin
17242:   TCardValue', 'Int;
17243:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts );
17244:   TShortSuit', '( cardS, cardD, cardC, cardH );
17245: Type(TDecks, Stand1, Standr2, Fises1, Fishes2, Beach, Leaves1, Leaves2, Robot, Roses, Shell, Castle, Hand);
17246:   SIRegister_TCard(CL);
17247:   SIRegister_TDeck(CL);
17248: end;
17249:
17250: Proc SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
17251: begin
17252:   tMethodCall', 'Procedure;tVerboseCall', 'Proc ( s :Str);
17253:   // PTEdge', '^TEdge // will not work;
17254:   TEdge', 'record FromNodeIndex : Int; ToNodeIndex : Int; '
17255:     +Weight : Int; work : Int; Len : Int; Highlight :Bool; end;
17256:   SIRegister_TNode(CL);
17257:   SIRegister_TGraphList(CL);
17258: end;
17259:
17260: Proc SIRegister_UParser10(CL: TPSPascalCompiler);
17261: begin
17262:   ParserFloat', 'extended;
17263:   //PParserFloat', '^ParserFloat /will not work;
17264:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
17265:     +ulo, IntDiv, IntDIVZ, Intpower, realpower, square, third, fourth, FuncOneVar, FuncTwoVar );
17266:   //POperation', '^TOperation // will not work;
17267:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
17268:     +est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
17269:     +; Token : TDFFToken; end;
17270:   TMathProcedure', 'procedure (AnOperation: TDFFOperation);
17271:   (TOBJECT)', 'EMathParserError;
17272:   (TOBJECT)', 'ESyntaxError;
17273:   (TOBJECT)', 'EExpressionHasBlanks;
17274:   (TOBJECT)', 'EExpressionTooComplex;
17275:   (TOBJECT)', 'ETooManyNestings;
17276:   (TOBJECT)', 'EMissMatchingBracket;
17277:   (TOBJECT)', 'EBadName;
17278:   (CL.FindClass('TOBJECT')), 'EParserInternalError;
17279:   (TExParserExceptionEvent', 'Proc ( Sender : TObject; E : Exception);
17280:   SIRegister_TCustomParser(CL);
17281:   SIRegister_TExParser(CL);
17282: end;
17283:
17284: Func isService:Bool;
17285: begin
17286:   result:= NOT(Application is TApplication);
17287:   {result:= Application is TServiceApplication;}
17288: end;
17289: Func isApplication:Bool;
17290: begin
17291:   result:= Application is TApplication;
17292: end;
17293: //SM_REMOTESESSION = $1000
17294: Func isTerminalSession:Bool;
17295: begin
17296:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
17297: end;
17298: writeln(inttostr(getmemoryload))
17299:
17300: Proc SIRegister_cyIEUtils(CL: TPSPascalCompiler);
17301: begin
17302:   TypeS('TwbPageSetup', record font :Str; footer :Str; header : '
17303:     +String; margin_bottom :Str; margin_left :Str; margin_right : Strin'
17304:     +g; margin_top :Str; Print_Background :Str; Shrink_To_Fit:str; end;
17305:   Func cyURLEncode( const S :Str) :Str;
17306:   Func MakeResourceURL(const ModulName:str;const ResName:PChar;const ResType:PChar):str;
17307:   Func MakeResourceURL1(const Modul:HMODULE;const ResName:PChar;const ResType:PChar):str;
17308:   Func cyColorToHtml( aColor : TColor) :Str;
17309:   Func HtmlToColor( aHtmlColor :Str) : TColor;
17310:   //Func GetStreamEncoding( aStream : TStream) : TEncoding;
17311:   // Func IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) :Bool;
17312:   Func AddHtmlUnicodePrefix( aHtml :Str) :Str;
17313:   Func RemoveHtmlUnicodePrefix( aHtml :Str) :Str;
17314:   Proc GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup);
17315:   Proc SetPageSetupToRegistry( awbPageSetup : TwbPageSetup);
17316:   ConstantN('IEBodyBorderless', 'String').SetString( 'none;
17317:   IEBodySingleBorder', 'String').SetString( '
17318:   IEDesignModeOn', 'String').SetString( 'On;
17319:   IEDesignModeOff', 'String').SetString( 'Off;

```

```

17320:   cHtmlUnicodePrefixChar', 'Char').SetString( # $FEFF);
17321:   cHtmlUnicodePrefixChar', 'Char').SetString( # $FE);
17322: end;
17323:
17324:
17325: Proc SIRegister_UcomboV2(CL: TPSPascalCompiler);
17326: begin
17327:   ConstantN('UMaxEntries', 'LongInt').SetInt( 600);
17328:   TypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
17329:     + 'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
17330:     + 'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
17331:     + 'inationsRepeat, CombinationsRepeatDown );
17332:   TypeS('TByteArray64', 'array[0..600 + 1] of int64;;
17333:   SIRegister_TComboSet(CL);
17334: end;
17335:
17336: Proc SIRegister_cyBaseComm(CL: TPSPascalCompiler);
17337: begin
17338:   TcyCommandType', '( ctDeveloperDefined, ctUserDefined );
17339:   TStreamContentType', '( scDeveloperDefined, scUserDefined, scString );
17340:   TProcOnReceiveCommand', 'Proc (Sender: TObject; aCommand: Word; userParam : Int);
17341:   TProcOnReceiveString', 'Proc ( Sender : TObject; fromBaseCo'
17342:     + 'mmHandle : THandle; aString :Str; userParam : Int);
17343:   TProcOnReceiveMemoryStream', 'Proc ( Sender : TObject; from'
17344:     + 'BaseCommHandle : THandle; aStream : TMemoryStream; userParam: Int);
17345:   SIRegister_TcyBaseComm(CL);
17346:   ConstantN('MsgCommand', 'LongInt').SetInt( WM_USER + 1);
17347:   ConstantN('MsgResultOk', 'LongInt').SetInt( 99);
17348:   Func ValidateFileMappingName( aName :Str) :Str;
17349:   Proc makeCaption(leftSide, Rightside:str; form:TForm);
17350: end;
17351:
17352: Proc SIRegister_cyDERUtils(CL: TPSPascalCompiler);
17353: begin
17354:   TypeS('DERString', 'String;
17355:   TypeS('DERChar', 'Char;
17356:   TypeS('TElementsType', ( etText, etExpressionKeyWord, etNumbers, etInt'
17357:     + 'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph);
17358:   TypeS('TElementsTypes', 'set of TElementsType;
17359:   TypeS('DERNString', 'String;
17360:   const DERDecimalSeparator', 'String').SetString( '.';
17361:   const DERDefaultChars, 'String') (+@/%-_.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNPOQRSTUVWXYZ;
17362:   const DERNDefaultChars', 'String').SetString( '/%-:0123456789abcdefghijklmnopqrstuvwxyz;
17363:   Func isValidWebSiteChar( aChar : Char) :Bool;
17364:   Func isValidWebMailChar( aChar : Char) :Bool;
17365:   Func isValidwebSite( aStr :Str) :Bool;
17366:   Func isValidWebMail( aStr :Str) :Bool;
17367:   Func ValidateDate( aDERStr : DERString; var RsltFormat :Str) :Bool;
17368:   Func DERStrToDate( aDERStr, aFormat:Str) : TDate;
17369:   Func IsDERChar( aChar : Char) :Bool;
17370:   Func IsDERDefaultChar( aChar : Char) :Bool;
17371:   Func IsDERMoneyChar( aChar : Char) :Bool;
17372:   Func IsDERExceptionCar( aChar : Char) :Bool;
17373:   Func IsDERSymbols( aDERString :Str) :Bool;
17374:   Func StringToDERCharSet( aStr :Str) : DERString;
17375:   Func IsDERNDefaultChar( aChar : Char) :Bool;
17376:   Func IsDERNChar( aChar : Char) :Bool;
17377:   Func DERTToDERNCharset( aDERStr : DERString) : DERNString;
17378:   Func DERExtractwebSite( aDERStr : DERString; SmartRecognition :Bool) :Str;
17379:   Func DERExtractWebMail( aDERStr : DERString) :Str;
17380:   Func DERExtractPhoneNr( aDERStr : DERString) :Str;
17381:   Func DERExecute(aDERStr:DERString; SmartNumbersRecognition, SmartWebsiteRecognition:Bool):TElementsType;
17382:   Func DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInt, RsltFloat, RsltPercentage, RsltwebSite,
17383:     RsltWebMail, RsltMoney, RsltDate:str; SmartNumbersRecognition, SmartWebsiteRecognition:Bool):TElementsType;
17384:   Proc RetrieveElementValue(aStr:str; SmartNumbersRecognition, SmartWebsiteRecognition:Bool; var RsltDERStr:
17385:     DERString; var RsltElementType:TElementsType);
17386: end;
17387: Proc SIRegister_cyImage(CL: TPSPascalCompiler);
17388: begin
17389:   pRGBQuadArray', '^TRGBQuadArray // will not work;
17390:   Func BitmapsCompare(Bmp1:TBitmap; Bmp2:TBitmap; FirstCol, LastCol, FirstRow, LastRow: Int): Int);
17391:   Proc BitmapSetPercentBrightness(Bmp: TBitmap; IncPercent: Int; RefreshBmp :Bool);
17392:   Proc BitmapSetPixelsBrightness(Bmp: TBitmap; IncPixels : Int; RefreshBmp:Bool);
17393:   Proc BitmapSetPercentContrast(Bmp: TBitmap; IncPercent : Int; RefreshBmp:Bool);
17394:   Proc BitmapSetPixelsContrast(Bmp: TBitmap; IncPixels : Int; RefreshBmp:Bool);
17395:   Proc BitmapNegative( Bmp : TBitmap; RefreshBmp :Bool);
17396:   Proc BitmapModifyRGB(Bmp:TBitmap; IncRed: Int; IncGreen: Int; IncBlue: Int; RefreshBmp: Bool;
17397:   Proc BitmapReplaceColor(Bmp:TBitmap; OldColor:TColor; NewColor:TColor; RangeRed, RangeGreen,
17398:   RangeBlue: Word; SingleDestinationColor: Bool; RefreshBmp: Bool);
17399:   Proc BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor; NewColor:TColor; PercentRange1Red, PercentRange1Green,
17400:   PercentRange1Blue: Extended; PercentRange2Red, PercentRange2Green,
17401:   PercentRange2Blue: Double; SingleDestinationColor: Bool; RefreshBmp: Bool;
17402:   Proc BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette: array of
17403:   TColor; SingleDestinationColor, RefreshBmp: Bool);
17404:   Proc BitmapResize(SourceBmp TBitmap; DestinationBmp: TBitmap; Percent: Extended; RefreshBmp: Bool
17405:   Proc BitmapRotate(Bmp: TBitmap; AngleDegree: Word; AdjustSize: Bool; BkColor: TColor);

```

```

17402: Proc BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Int;RefreshBmp:Bool;
17403: Proc GraphicMirror(Source:TGraphic;Dest:TCanvas;Left,Top:Int;Horizontal,Vertical:Bool;
17404: Proc GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Bool;
17405: Func BitmapCreate(BmpWidth:Int;BmpHeight:Int;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
17406: Proc BitmapSaveToJpegFile(Bmp : TBitmap; FileName :Str; QualityPercent : Word);
17407: Proc JpegSaveToBitmapFile(JPEG : TJPEGImage; FileName :Str);
17408: end;
17409:
17410: Proc SIRegister_WaveUtils(CL: TPSPascalCompiler);
17411: begin
17412:   TMS2StrFormat', '( msHMSH, msHMS, msMSH, msMS, msSh, msS, msAh,msA );
17413:   TPCMChannel', '( cMono, cStereo );
17414:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz );
17415:   TPCMBitsPerSample', '( bs8Bit, bs16Bit );
17416:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1
17417:     +6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
17418:     +it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
17419:     +it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
17420:     +it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
17421:     +it48000Hz, Stereo16bit48000Hz );
17422:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
17423:     +nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word; ecbSiz: Word; end;
17424:   tWaveFormatEx', 'PWaveFormatEx; HMMIO', 'Int;
17425:   TWaveDeviceFormats', 'set of TPCMFormat;
17426:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
17427:     +PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound );
17428:   TypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport;
17429:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate );
17430:   TWaveOutOptions', 'set of TWaveOutOption;
17431:   TStreamOwnership2', '( soReference, soOwned );
17432:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx );
17433:   // PRawWave', '^TRawWave // will not work;
17434:   TRawWave', 'record pData : TObject; dwSize : DWORD; end;
17435:   ClassN(CL.FindClass('TObject'),'EWaveAudioError;
17436:   ClassN(CL.FindClass('TObject'),'EWaveAudioSysError;
17437:   ClassN(CL.FindClass('TObject'),'EWaveAudioInvalidOperation;
17438:   TWaveAudioEvent', 'Proc (Sender : TObject);
17439:   TWaveAudioGetFormatEvent', 'Proc (Sender : TObject; var pW'
17440:     +aveFormat : PWaveFormatEx; var FreeIt : Bool);
17441:   TWaveAudioGetDataEvent', 'Func (Sender : TObject; const Buf'
17442:     +fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD;
17443:   TWaveAudioGetDataPtrEvent', 'Func ( Sender : TObject; var Bu'
17444:     +ffer : TObject; var NumLoops : DWORD; var FreeIt :Bool) : DWORD;
17445:   TWaveAudioDataReadyEvent', 'Proc ( Sender : TObject; const '
17446:     +Buffer : TObject; BufferSize : DWORD; var FreeIt :Bool);
17447:   TWaveAudioLevelEvent', 'Proc ( Sender : TObject; Level : Int);
17448:   TWaveAudioFilterEvent, 'Procedure (Sender:TObject;const Buffer:TObject; BufferSize:DWORD);
17449:   GetWaveAudioInfo (mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
17450:   CreateWaveAudio (mmIO:HMMIO;const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
17451:   CloseWaveAudio ( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo);
17452:   GetStreamWaveAudioInfo (Stream:TStream;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
17453:   CreateStreamWaveAudio (Stream:TStream;const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):HMMIO;
17454:   OpenStreamWaveAudio ( Stream : TStream) : HMMIO;
17455:   CalcWaveBufferSize ( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD;
17456:   GetAudioFormat ( FormatTag : Word) : Str;
17457:   GetWaveAudioFormat ( const pWaveFormat : PWaveFormatEx) : Str;
17458:   GetWaveAudioLength ( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD;
17459:   GetWaveAudioBitRate ( const pWaveFormat : PWaveFormatEx) : DWORD;
17460:   GetWaveAudioPeakLevel (const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx) : Int;
17461:   InvertWaveAudio (const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx):Bool;
17462:   SilenceWaveAudio (const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx):Bool;
17463:   ChangeWaveAudioVolume (const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
17464:   MixWaveAudio (const RawWaves:TRawWave;Count:Int;const
    pWaveFormat:PWaveFormatEx;Buffer:TObject;BufferSize:DWORD):Bool;
17465:   ConvertWaveFormat (const srcFormat:PWaveFormatEx;srcData:TObject;srcDataSize:DWORD;const
    dstFormat:PWaveFormatEx; var dstData:TObject;var dstDataSize:DWORD):Bool;
17466:   SetPCMAudioFormat (const pWaveFormat:PWaveFormatEx;Channels:TPCMChannel;SamplesPerSec:TPCMSamplesPerSec;
    BitsPerSample:TPCMBitsPerSample);
17467:   Proc SetPCMAudioFormatS ( const pWaveFormat : PWaveFormatEx; PCMFormat:TPCMFormat);
17468:   GetPCMAudioFormat ( const pWaveFormat : PWaveFormatEx) : TPCMFormat;
17469:   GetWaveDataPositionOffset ( const pWaveFormat : PWaveFormatEx; Position:DWORD):DWORD;
17470:   MS2Str ( Milliseconds : DWORD; Fmt : TMS2StrFormat) : Str;
17471:   WaitForSyncObject (SyncObject : THandle; Timeout : DWORD) : DWORD;
17472:   //mmioStreamProc ( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT;
17473: end;
17474:
17475: Proc SIRegister_NamedPipes (CL: TPSPascalCompiler);
17476: begin
17477:   'DEFAULT_PIPE_BUFFER_SIZE', 'LongInt').SetInt ( 4096);
17478:   ConstantN('DEFAULT_PIPE_TIMEOUT', 'LongInt').SetInt ( 5000);
17479:   'PIPE_NAMING_SCHEME', 'String).SetString ( '\\%s\pipe\%s;
17480:   'WAIT_ERROR', 'LongWord').SetUInt ( DWORD ( $FFFFFFF ));
17481:   WAIT_OBJECT_1', 'LongInt').SetInt ( WAIT_OBJECT_0 + 1);
17482:   'STATUS_SUCCESS', 'LongWord').SetUInt ( $00000000);
17483:   'STATUS_BUFFER_OVERFLOW', 'LongWord').SetUInt ( $80000005);
17484:   TypeS('TPipeDirection', ( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient );
17485:   TypeS('TPipeType', ( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg );
17486:   TypeS('TOverlappedResult', ' (ov_Failed, ov_Pending, ov_MoreData, ov_Complete );
17487:   SIRegister_TNamedPipe (CL);

```

```

17488:   SIRegister_TServerPipe(CL);
17489:   SIRegister_TClientPipe(CL);
17490: Func CalculateTimeout( aBasis : DWORD) : DWORD;
17491: Func HasOverlappedIoCompleted( const ov : OVERLAPPED) :Bool;
17492: Func GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var
dwBytes:DWORD;bWait:Bool):TOverlappedResult;
17493: Func GetStreamAsText( stm : TStream) :Str;
17494: Proc SetStreamAsText(const aTxt:Str; stm : TStream);
17495: end;
17496:
17497: Proc SIRegister_DPUtils(CL: TPSPascalCompiler);
17498: begin
17499:   //TypeS('pRGBTripleArray', '^TRGBTripleArray // will not work;
17500:   SIRegister_TThumbData(CL);
17501:   'PIC_BMP','LongInt').SetInt( 0);
17502:   'PIC_JPG','LongInt').SetInt( 1);
17503:   'THUMB_WIDTH','LongInt').SetInt( 60);
17504:   'THUMB_HEIGHT','LongInt').SetInt( 60);
17505: Func IsEqualFile(Filename:Str; Size:Int; LastWriteTime : TDateTime):Bool;
17506: Proc GetFileInfo(Filename:Str; var Size : Int; var LastWriteTime : TDateTime);
17507: Func ReadBitmap( Filehandle : Int; Width, Height : Int) : TBitmap;
17508: Proc WriteBitmap( Filehandle : Int; bmp : TBitmap);
17509: Func OpenPicture( fn :Str; var tp : Int) : Int;
17510: Func ConvertPicture( pi : Int; tp : Int) : TBitmap;
17511: Func LoadPicture( fn :Str; var w, h : Int) : TBitmap;
17512: Func TurnBitmap( bmp : TBitmap; ang : Int) : TBitmap;
17513: Func RotateBitmap( Bitmap : TBitmap; Direction : Int) : TBitmap;
17514: Func StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect;
17515: Func ThumbBitmap( Bitmap : TBitmap; Width, Height : Int) : TBitmap;
17516: Proc ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Int);
17517: Proc FindFiles( path, mask :Str; items : TStringList);
17518: Func LetFileName( s :Str) :Str;
17519: Func LetParentPath( path :Str) :Str;
17520: Func AddBackSlash( path :Str) :Str;
17521: Func CutBackSlash( path :Str) :Str;
17522: end;
17523:
17524: Proc SIRegister_CommonTools(CL: TPSPascalCompiler);
17525: begin
17526:   //'BYTES','LongInt').SetInt( 1);
17527:   'KBBYTES','LongInt').SetInt( 1024);
17528:   'DBG_ALIVE','LongWord').SetUInt( Int ( $11BABEL1 ));
17529:   'DBG_DESTROYING','LongWord').SetUInt( Int ( $44FADE44 ));
17530:   'DBG_GONE','LongWord').SetUInt( $99ACID99);
17531:   'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
17532:   SIRegister_MakeComServerMethodsPublic(CL);
17533:   TypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application );
17534: Func IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) :Bool;
17535: Proc TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet :Bool);
17536: Func TBGetTempFolder :Str;
17537: Func TBGetTempFile :Str;
17538: Func TBGetModuleFilename :Str;
17539: Func FormatModuleVersionInfo( const aFilename :Str) :Str;
17540: Func GetVersionInfoString( const aFile, aEntry :Str; aLang : WORD) :Str;
17541: Func TBGetFileSize( aFile :Str; aMultipleOf : Int) : Int;
17542: Func FormatAttribString( aAttr : Int) :Str;
17543: Func GetSomeFileInfo( aFile :Str; aWhatInfo : TSomeFileInfo) :Str;
17544: Func ShellRecycle( aWnd : HWND; aFileOrFolder :Str) :Bool;
17545: Func IsDebuggerPresent : BOOL;
17546: Func TBNotImplemented : HRESULT;
17547: end;
17548:
17549: Proc SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
17550: begin
17551:   //TypeS('OSVERSIONINFOEXA', ' OSVERSIONINFOEXA;
17552:   //TypeS('TOSVersionInfoExA', ' OSVERSIONINFOEXA;
17553:   TypeS('TOSVersionInfoEx', 'TOSVersionInfo;
17554:   TypeS('TDrivesProperty', 'array[1..26] of boolean;
17555:   //TDrivesProperty = array['A'..'Z'] of boolean;
17556: Func TBSetSystemTime( DateTime : TDateTime; DOW : word) :Bool;
17557: Func IsElevated :Bool;
17558: Proc CoCreateInstanceAsAdmin(aHwnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
17559:   TypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined );
17560: Func TrimNetResource( UNC :Str) :Str;
17561: Proc GetFreeDrives( var FreeDrives : TDrivesProperty);
17562: Proc GetMappedDrives( var MappedDrives : TDrivesProperty);
17563: Func MapDrive(UNCPath:Str;Drive:char;PasswordUsage:TPasswordUsage;Password:Str;
UserUsage:TPasswordUsage;User:Str;Comment:Str):bool;
17564: Func UnmapDrive( Drive : char; Force :Bool):Bool;
17565: Func TBIsWindowsVista :Bool;
17566: Proc SetVistaFonts( const AForm : TForm);
17567: Proc SetVistaContentFonts( const AFont : TFont);
17568: Func GetProductType( var sType :Str) :Bool;
17569: Func lstrcmp( lpString1, lpString2 : PChar) : Int;
17570: Func lstrcmpi( lpString1, lpString2 : PChar) : Int;
17571: Func lstrcmpyn( lpString1, lpString2 : PChar; iMaxLength : Int) : PChar;
17572: Func lstrcpy( lpString1, lpString2 : PChar) : PChar;
17573: Func lstrcat( lpString1, lpString2 : PChar) : PChar;
17574: Func lstrlen( lpString : PChar) : Int;

```

```

17575: Func
GetTokenInformation(TokenHandle:THandle;TokenInformationClass:TTokenInformationClass;TokenInformation:___Pointer;Token
ReturnLength:DWORD):BOOL;
17576: Func
SetTokenInformation(TokenHandle:THandle;TokenInformationClass:TTokenInformationClass;TokenInformation:___Pointer;Token
17577: end;
17578:
17579: Proc SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
17580: begin
17581:   'cLineTerminator','Char').SetString( #10);
17582:   'cLineTerminators','String').SetString( #13#10);
17583:   'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
17584:   SIRegister_TFixedCriticalSection(CL);
17585:   SIRegister_TMultiReadSingleWrite(CL);
17586: Proc SetDecimalSeparator( c : Char);
17587: Func GetDecimalSeparator : Char;
17588: Func GetDecimalSeparator : Char;
17589: Proc SetDateSeparator( c : Char);
17590: Func GetDateSeparator : Char;
17591: Proc setTimeSeparator( c : Char);
17592: Func getTimeSeparator : Char;
17593: Proc setShortDateFormat( c :Str);
17594: Func getShortDateFormat :Str;
17595: Proc setlongDateFormat( c :Str);
17596: Func getlongDateFormat :Str;
17597: Proc setShorttimeFormat( c :Str);
17598: Func getShorttimeFormat :Str;
17599: Proc setlongtimeFormat( c :Str);
17600: Func getlongtimeFormat :Str;
17601: //getTimeSeparator
17602: Proc setThousandSeparator(c : Char);
17603: Func getThousandSeparator : Char;
17604: Proc setListSeparator( c : Char);
17605: Func getListSeparator : Char;
17606: 'LOGON32_LOGON_INTERACTIVE','LongInt').SetInt( 2);
17607: 'LOGON32_LOGON_NETWORK','LongInt').SetInt( 3);
17608: 'LOGON32_LOGON_BATCH','LongInt').SetInt( 4);
17609: 'LOGON32_LOGON_SERVICE','LongInt').SetInt( 5);
17610: 'LOGON32_PROVIDER_DEFAULT','LongInt').SetInt( 0);
17611: 'LOGON32_PROVIDER_WINNT35','LongInt').SetInt( 1);
17612: 'LOGON32_PROVIDER_WINNT40','LongInt').SetInt( 2);
17613: 'LOGON32_PROVIDER_WINNT50','LongInt').SetInt( 3);
17614:
17615: TCollectFileProgressEvent', 'Proc ( const directory:Str; var skipScan:Bool);
17616: Proc CollectFiles(const directory,fileMask:UnicodeString; list:TStrings; recurseSubdirectories:Boolean;
onProgress : TCollectFileProgressEvent);
17617: TypeS('NativeInt', 'Int;
17618: //TypeS('PNativeInt', '^NativeInt // will not work;
17619: TypeS('NativeUInt', 'Cardinal;
17620: //TypeS('PNativeUInt', '^NativeUInt // will not work;
17621: //TypeS('TBytes', 'array of Byte;
17622: TypeS('RawByteString', 'UnicodeString;
17623: //TypeS('PNativeInt', '^NativeInt // will not work;
17624: //TypeS('PUInt64', '^UInt64 // will not work;
17625: SIRegister_TPath(CL);
17626: SIRegister_TFile(CL);
17627: SIRegister_TdwsThread(CL);
17628: Func GetSystemMilliseconds : Int64;
17629: Func UTCDateTime : TDateTime;
17630: Func UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString;
17631: Func UnicodeCompareStr( const s1, s2 : UnicodeString) : Int;
17632: Func dwsAnsiCompareText( const s1, s2 : UnicodeString) : Int;
17633: Func dwsAnsiCompareStr( const s1, s2 : UnicodeString) : Int;
17634: Func UnicodeComparePChars(pl:PChar; n1 : Int; p2 : PChar; n2 : Int): Int;;
17635: Func UnicodeComparePChars1( pl, p2 : PChar; n : Int) : Int;;
17636: Func UnicodeLowerCase( const s : UnicodeString) : UnicodeString;
17637: Func UnicodeUpperCase( const s : UnicodeString) : UnicodeString;
17638: Func ASCIICompareText( const s1, s2 : UnicodeString) : Int;
17639: Func ASCIISameText( const s1, s2 : UnicodeString) :Bool;
17640: Func InterlockedIncrement( var val : Int) : Int;
17641: Func InterlockedDecrement( var val : Int) : Int;
17642: Proc FastInterlockedIncrement( var val : Int);
17643: Proc FastInterlockedDecrement( var val : Int);
17644: Func InterlockedExchangePointer(var target:___Pointer; val:___Pointer):___Pointer;
17645: Proc SetThreadName( const threadName : Char; threadID :Card);
17646: Proc dwsOutputDebugString( const msg : UnicodeString);
17647: Proc WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeStr;const logRawData:Str)
17648: Func TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
17649: Proc VarCopy( out dest : Variant; const src : Variant);
17650: Func VarToUnicodeStr( const v : Variant) : UnicodeString;
17651: Func LoadTextFromBuffer( const buf : TBytes) : UnicodeString;
17652: Func LoadTextFromStream( aStream : TStream) : UnicodeString;
17653: Func LoadTextFromFile( const fileName : UnicodeString) : UnicodeString;
17654: Proc SaveTextToUTF8File( const fileName, text : UnicodeString);
17655: Func OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle;
17656: Func OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle;
17657: Proc CloseFileHandle( hFile : THandle);
17658: Func FileCopy(const existing, new : UnicodeString; failIfExists :Bool):Boolean;
17659: Func FileMove( const existing, new : UnicodeString) :Bool;

```



```

17660: Func dwsFileDelete( const fileName :Str) :Bool;
17661: Func FileRename( const oldName, newName :Str) :Bool;
17662: Func dwsFileSize( const name :Str) : Int64;
17663: Func dwsFileDateTime( const name :Str) : TDateTime;
17664: Func DirectSet8087CW( newValue : Word) : Word;
17665: Func DirectSetMXCSR( newValue : Word) : Word;
17666: Func TtoObject( const T: byte) : TObject;
17667: Func TtoPointer( const T: byte) : __Pointer;
17668: Proc GetMemForT( var T: byte; Size : Int);
17669: Func FindDelimiter( const Delimiters, S :Str; StartIdx : Int) : Int;
17670: end;
17671:
17672: Proc SIRegister_AdSocket(CL: TPSPascalCompiler);
17673: begin
17674:   'IPStrSize','LongInt').SetInt( 15);
17675:   'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17676:   'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
17677:   'ADWSBASE','LongInt').SetInt( 9000);
17678:   TypeS('TCMAPDSOCKETMESSAGE', 'record Msg :Card; Socket : TSocket; '
17679:     +'SelectEvent : Word; SelectError : Word; Result : Longint; end;
17680:   SIRegister_EApdSocketException(CL);
17681:   TWsMode', '( wsClient, wsServer );
17682:   TWsNotifyEvent', 'Proc ( Sender : TObject; Socket : TSocket;
17683:   TWsSocketErrorEvent', 'Proc ( Sender : TObject; Socket : TSocket;ErrCode : Int);
17684:   SIRegister_TApdSocket(CL);
17685: end;
17686:
17687: Proc SIRegister_AdPort(CL: TPSPascalCompiler);
17688: begin
17689:   SIRegister_TApdCustomComPort(CL);
17690:   SIRegister_TApdComPort(CL);
17691:   Func ComName( const ComNumber : Word) : ShortString;
17692:   Func SearchComPort( const C : TComponent) : TApdCustomComPort;
17693: end;
17694:
17695: Proc SIRegister_PathFunc(CL: TPSPascalCompiler);
17696: begin
17697:   Func inAddBackslash( const S :Str) :Str;
17698:   Func PathChangeExt( const Filename, Extension :Str) :Str;
17699:   Func PathCharCompare( const S1, S2 : PChar) :Bool;
17700:   Func PathCharIsSlash( const C : Char) :Bool;
17701:   Func PathCharIsTrailByte( const S :Str; const Index : Int) :Bool;
17702:   Func PathCharLength( const S :Str; const Index : Int) : Int;
17703:   Func inPathCombine( const Dir, Filename :Str) :Str;
17704:   Func PathCompare( const S1, S2 :Str) : Int;
17705:   Func PathDrivePartLength( const Filename :Str) : Int;
17706:   Func PathDrivePartLengthEx(const Filename:str;const IncludeSignificantSlash:Bool):Int;
17707:   Func inPathExpand( const Filename :Str) :Str;
17708:   Func PathExtensionPos( const Filename :Str) : Int;
17709:   Func PathExtractDir( const Filename :Str) :Str;
17710:   Func PathExtractDrive( const Filename :Str) :Str;
17711:   Func PathExtractExt( const Filename :Str) :Str;
17712:   Func PathExtractName( const Filename :Str) :Str;
17713:   Func PathExtractPath( const Filename :Str) :Str;
17714:   Func PathIsRooted( const Filename :Str) :Bool;
17715:   Func PathLastChar( const S :Str) : PChar;
17716:   Func PathLastDelimiter( const Delimiters, S :Str) : Int;
17717:   Func PathLowercase( const S :Str) :Str;
17718:   Func PathNormalizeSlashes( const S :Str) :Str;
17719:   Func PathPathPartLength(const Filename:str;const IncludeSlashesAfterPath:Bool):Int;
17720:   Func PathPos( Ch : Char; const S :Str) : Int;
17721:   Func PathStartsWith( const S, AStartsWith :Str) :Bool;
17722:   Func PathStrNextChar( const S : PChar) : PChar;
17723:   Func PathStrPrevChar( const Start, Current : PChar) : PChar;
17724:   Func PathStrScan( const S : PChar; const C : Char) : PChar;
17725:   Func inRemoveBackslash( const S :Str) :Str;
17726:   Func RemoveBackslashUnlessRoot( const S :Str) :Str;
17727:   Proc PathFuncRunTests( const AlsoTestJapaneseDBCS :Bool);
17728: end;
17729:
17730: Proc SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17731: begin
17732:   NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion;
17733:   NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer;
17734:   NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetStr(NEWREGSTR_PATH_EXPLORER + '\Shell Folders;
17735:   NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall;
17736:   NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName;
17737:   NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString;
17738:   KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17739:   //TypeS('PLeadByteSet', '^TLeadByteSet // will not work;
17740:   TypeS('TLeadByteSet', 'set of Char;
17741:   SIRegister_TOneShotTimer(CL);
17742:   TypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit );
17743:   'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17744:   Func NewFileExists( const Name :Str) :Bool;
17745:   Func inDirExists( const Name :Str) :Bool;
17746:   Func FileOrDirExists( const Name :Str) :Bool;
17747:   Func IsDirectoryAndNotReparsePoint( const Name :Str) :Bool;
17748:   Func GetIniString(const Section,Key:str; Default:str;const Filename:str):str;

```

```

17749: Func GetIniInt(const Section,Key:Str;const Default,Min,Max:Longint;const Filename:Str):Longint;
17750: Func GetIniBool(const Section,Key:Str;const Default:Bool;const Filename:Str):Bool;
17751: Func IniKeyExists(const Section, Key, Filename :Str) :Bool;
17752: Func IsIniSectionEmpty(const Section, Filename :Str) :Bool;
17753: Func SetIniString(const Section, Key, Value, Filename :Str) :Bool;
17754: Func SetIniInt(const Section,Key:Str;const Value:Longint;const Filename:Str):Bool;
17755: Func SetIniBool(const Section,Key:Str;const Value:Bool;const Filename:Str):Bool;
17756: Proc DeleteIniEntry(const Section, Key, Filename :Str);
17757: Proc DeleteIniSection(const Section, Filename :Str);
17758: Func GetEnv(const EnvVar :Str) :Str;
17759: Func GetCmdTail :Str;
17760: Func GetCmdTailEx( StartIndex : Int) :Str;
17761: Func NewParamCount : Int;
17762: Func NewParamStr( Index : Int) :Str;
17763: Func AddQuotes(const S :Str) :Str;
17764: Func RemoveQuotes(const S :Str) :Str;
17765: Func inGetShortName(const LongName :Str) :Str;
17766: Func inGetWinDir :Str;
17767: Func inGetSystemDir :Str;
17768: Func GetSysWow64Dir :Str;
17769: Func GetSysNativeDir(const IsWin64 :Bool) :Str;
17770: Func inGetTempDir :Str;
17771: Func StringChange(var S :Str; const FromStr, ToStr :Str) : Int;
17772: Func StringChangeEx(var S:Str;const FromStr,ToStr:Str;const SupportDBCS:Bool):Int;
17773: Func AdjustLength(var S :Str; const Res :Card) :Bool;
17774: Func UsingWinNT :Bool;
17775: Func ConvertConstPercentStr(var S :Str) :Bool;
17776: Func ConvertPercentStr(var S :Str) :Bool;
17777: Func ConstPos(const Ch : Char; const S :Str) : Int;
17778: Func SkipPastConst(const S :Str; const Start : Int) : Int;
17779: Func RegQueryStringValue(H : HKEY; Name : PChar; var ResultStr :Str) :Bool;
17780: Func RegQueryMultiStringValue(H : HKEY; Name : PChar; var ResultStr:Str):Bool;
17781: Func RegValueExists(H : HKEY; Name : PChar) :Bool;
17782: 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;
17783: Func RegOpenKeyExView(const
RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17784: Func RegDeleteKeyView(const RegView:TRegView;const Key:HKEY;const Name:PChar): Longint;
17785: Func RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17786: Func RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyNam:PChar):Longint;
17787: Func GetShellFolderPath(const FolderID : Int) :Str;
17788: Func IsAdminLoggedOn :Bool;
17789: Func IsPowerUserLoggedOn :Bool;
17790: Func IsMultiByteString(const S : Ansistr) :Bool;
17791: Func FontExists(const FaceName :Str) :Bool;
17792: //Proc FreeAndNil(var Obj);
17793: Func SafeLoadLibrary(const Filename :Str; ErrorMode : UINT) : HMODULE;
17794: Func GetUILanguage : LANGID;
17795: Func RemoveAccelChar(const S :Str) :Str;
17796: Func GetTextWidth(const DC : HDC; S :Str; const Prefix:Boolean):Int;
17797: Func AddPeriod(const S :Str) :Str;
17798: Func GetExceptMessage :Str;
17799: Func GetPreferredUIFont :Str;
17800: Func IsWildcard(const Pattern :Str) :Bool;
17801: Func WildcardMatch(const Text, Pattern : PChar) :Bool;
17802: Func IntMax(const A, B : Int) : Int;
17803: Func Win32ErrorString( ErrorCode : Int) :Str;
17804: Proc GetLeadBytes(var ALeadBytes : TLeadByteSet);
17805: Func inCompareMem(P1, P2 : TObject; Length : Int) :Bool;
17806: Func DeleteDirTree(const Dir :Str) :Bool;
17807: Func SetNTFSCompression(const FileOrDir:Str; Compress :Bool) :Bool;
17808: Proc AddToWindowMessageFilterEx(const Wnd : HWND; const Msg : UINT);
17809: // TypeS('TSysCharSet', 'set of AnsiChar;
17810: Func inCharInSet(C : Char; const CharSet : TSysCharSet) :Bool;
17811: Func ShutdownBlockReasonCreate(Wnd : HWND; const Reason :Str) :Bool;
17812: Func ShutdownBlockReasonDestroy(Wnd : HWND) :Bool;
17813: Func TryStrToBoolean(const S :Str; var BoolResult :Bool) :Bool;
17814: Proc WaitMessageWithTimeout(const Milliseconds : DWORD);
17815: Func MoveFileReplace(const ExistingFileName, NewFileName :Str) :Bool;
17816: Proc TryEnableAutoCompleteFileSystem(Wnd : HWND);
17817: end;
17818:
17819: Proc SIRegister_CmnFunc(CL: TPSPascalCompiler);
17820: begin
17821: SIRegister_TWindowDisabler(CL);
17822: TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError );
17823: TMsgBoxCallbackFunc,procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17824: Proc UpdateHorizontalExtent(const ListBox : TCustomListBox);
17825: Func MinimizePathName(const Filename:Str; const Font:TFont;MaxLen:Int):Str;
17826: Func AppMessageBox(const Text, Caption : PChar; Flags : Longint) : Int;
17827: Func MsgBoxP(const Text,Caption:PChar;const Typ: TMsgBoxType;const Buttons:Card):Int;
17828: Func inMsgBox(const Text,Caption:Str;const Typ:TMsgBoxType;const Buttons:Card):Int;
17829: Func MsgBoxFmt(const Text:Str;const Args:array of const;const Caption:Str;const Typ:TMsgBoxType;const
Buttons:Card):Int;
17830: Proc ReactivateTopWindow;
17831: Proc SetMessageBoxCaption(const Typ : TMsgBoxType; const NewCaption : PChar);
17832: Proc SetMessageBoxRightToLeft(const ARightToLeft :Bool);
17833: Proc SetMessageBoxCallbakFunc(const AFunc:TMsgBoxCallbackFunc;const AParam:LongInt);

```

```

17834: end;
17835:
17836: Proc SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17837: begin
17838:   SIRegister_TImageGrabber(CL);
17839:   SIRegister_TCaptureDrivers(CL);
17840:   SIRegister_TCaptureDriver(CL);
17841: end;
17842:
17843: Proc SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17844: begin
17845:   Func GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:str;const
17846:   Entries:TGrantPermissionEntry; const EntryCount:Int):Boolean;
17847:   Func GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:str;const Entries:
17848:   TGrantPermissionEntry; const EntryCount:Int):Boolean;
17849: end;
17850:
17851: Proc SIRegister_RedirFunc(CL: TPSPascalCompiler);
17852: begin
17853:   TypeS(TPreviousFsRedirectionState,record DidDisable:Boolean;OldValue: __Pointer;end;
17854:   Func AreFsRedirectionFunctionsAvailable :Bool;
17855:   Func DisableFsRedirectionIf(const Disable:Bool;var PreviousState:TPreviousFsRedirectionState):Bool;
17856:   Proc RestoreFsRedirection(const PreviousState : TPreviousFsRedirectionState);
17857:   Func CreateDirectoryRedir(const DisableFsRedir:Boolean; const Filename:Str) : Bool;
17858:   Func CreateProcessRedir(const DisableFsRedir:Boolean;const lpApplicationName:PChar;const
17859:   lpCommandLine:PChar; const lpProcessAttributes, lpThreadAttributes:PSecurityAttributes;const
17860:   bInheritHandles:BOOL; const dwCreationFlags:DWORD;const lpEnvironment:Pointer; const
17861:   lpCurrentDirectory:PChar; const lpStartupInfo : TStartupInfo; var
17862:   lpProcessInformation:TProcessInformation):BOOL;
17863:   Func CopyFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:str;const
17864:   FailIfExists:BOOL):BOOL;
17865:   Func DeleteFileRedir(const DisableFsRedir:Boolean; const Filename:Str): Bool;
17866:   Func DirExistsRedir(const DisableFsRedir:Boolean; const Filename:Str):Bool;
17867:   Func FileOrDirExistsRedir(const DisableFsRedir:Bool; const Filename:str):Bool;
17868:   Func FindFirstFileRedir(const DisableFsRedir:Boolean;const Filename:str;var
17869:   FindData:TWin32FindData):THandle;
17870:   Func GetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:str): DWORD;
17871:   Func GetShortNameRedir(const DisableFsRedir:Boolean; const Filename:str):str;
17872:   Func GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:str;var
17873:   VersionNumbers:TFileVersionNumbers): Bool;
17874:   Func IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Bool;const Name:str):Bool;
17875:   Func MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:str):BOOL;
17876:   Func MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:str;const
17877:   Flags:DWORD):BOOL;
17878:   Func NewFileExistsRedir(const DisableFsRedir:Boolean;const Filename:str):Bool;
17879:   Func RemoveDirectoryRedir(const DisableFsRedir:Boolean;const Filename :Str) : Bool;
17880:   Func SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:str;const Attrib:DWORD):BOOL;
17881:   Func SetNTFSCompressionRedir(const DisableFsRedir:Bool;const FileOrDir:Str;Compress:Bool;Bool;
17882:   SIRegister_TFileRedir(CL);
17883:   SIRegister_TTextFileReaderRedir(CL);
17884:   SIRegister_TTextFileWriterRedir(CL);
17885: end;
17886:
17887: Proc SIRegister_Int64Em(CL: TPSPascalCompiler);
17888: begin
17889:   //TypeS('LongWord', 'Cardinal;
17890:   TypeS('Int64', 'record Lo : LongWord; Hi : LongWord; end;
17891:   Func Compare64(const N1, N2 : Int64) : Int;
17892:   Proc Dec64(var X : Int64; N : LongWord);
17893:   Proc Dec6464(var X : Int64; const N : Int64);
17894:   Func Div64(var X : Int64; const Divisor : LongWord) : LongWord;
17895:   Func Inc64(var X : Int64; N : LongWord) :Bool;
17896:   Func Inc6464(var X : Int64; const N : Int64) :Bool;
17897:   Func Int64ToStr(X : Int64) :Str;
17898:   Func Mod64(const X : Int64; const Divisor : LongWord) : LongWord;
17899:   Func Mul64(var X : Int64; N : LongWord) :Bool;
17900:   Proc Multiply32x32to64(N1, N2 : LongWord; var X : Int64);
17901:   Proc Shr64(var X : Int64; Count : LongWord);
17902:   Func StrToInt64(const S :Str; var X : Int64) :Bool;
17903: end;
17904:
17905: Proc SIRegister_InstFunc(CL: TPSPascalCompiler);
17906: begin
17907:   //TypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work;
17908:   SIRegister_TSimpleStringList(CL);
17909:   TypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle);
17910:   TypeS('TDetermineDefaultLanguageResult','(ddNoMatch,ddMatch,ddMatchLangParameter );
17911:   TypeS('TMD5Digest', 'array[0..15] of Byte;
17912:   TypeS('TSHA1Digest', 'array[0..19] of Byte;
17913:   // TMD5Digest = array[0..15] of Byte; TSHA1Digest = array[0..19] of Byte;
17914:   Func CheckForMutexes(Mutexes :Str) :Bool;
17915:   Func CreateTempDir :Str;
17916:   Func DecrementSharedCount(const RegView:TRegView;const Filename:str):Bool;
17917:   Proc DelayDeleteFile(const DisableFsRedir :Bool; const Filename :Str; const MaxTries, FirstRetryDelayMS,
17918:   SubsequentRetryDelayMS : Int);
17919:   //Func DelTree(const DisableFsRedir :Bool; const Path :Str; const IsDir, DeleteFiles, DeleteSubdirsAlso,
17920:   BreakOnError :Bool; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc : TDeleteFileProc; const
17921:   Param : Pointer) :Bool;
17922:   //Func DetermineDefaultLanguage(const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
17923:   TSetupLanguageDetectionMethod;const LangParameter:str;var ResultIndex:Int):
17924:   TDetermineDefaultLanguageResult;

```

```

17910: //Proc EnumFileReplaceOperationsFileNames(const EnumFunc:TEnumFROFileNamesProc;Param:Pointer);
17911: Func GenerateNonRandomUniqueFilename(Path:Str; var Filename:Str):Bool;
17912: Func GenerateUniqueName(const DisableFsRedir:Bool;Path:Str;const Extension:str):str;
17913: Func GetComputerNameString :Str;
17914: Func GetFileDateTime(const DisableFsRedir:Bool;const Filename:Str;var DateTime:TFileTime):Bool;
17915: Func GetMD5OfFile(const DisableFsRedir:Boolean; const Filename :Str): TMD5Digest;
17916: Func GetMD5OfAnsistr( const S : Ansistr) : TMD5Digest;
17917: // Func GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest;
17918: Func GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:str):TSHA1Digest;
17919: Func GetSHA1OfAnsistr( const S : Ansistr) : TSHA1Digest;
17920: // Func GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest;
17921: Func GetRegRootKeyName( const RootKey : HKEY) :Str;
17922: Func GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:str;var FreeBytes,TotBytes:Int64):Bool;
17923: Func GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:str;var FreeBytes,
TotalBytes:Int64):Bool;
17924: Func GetUserNameString :Str;
17925: Proc IncrementSharedCount(const RegView:TRegView;const Filename:str;const AlreadyExisted:Bool;
17926: //Func InstExec( const DisableFsRedir :Bool; const Filename, Params :Str; WorkingDir :Str; const
Wait:TExecWait;const ShowCmd:Int;const ProcessMessagesProc:TProcedure;var ResultCode:Int) :Bool;
17927: // Func InstShellExec( const Verb, Filename, Params :Str; WorkingDir :Str; const Wait:TExecWait;const
ShowCmd:Int;const ProcessMessagesProc:TProcedure;var ResultCode:Int):Bool;
17928: Proc InternalError( const Id :Str);
17929: Proc InternalErrorFmt( const S :Str; const Args : array of const);
17930: Func IsDirEmpty( const DisableFsRedir :Bool; const Dir :Str) :Bool;
17931: Func IsProtectedSystemFile(const DisableFsRedir:Boolean;const Filename:str):Bool;
17932: Func MakePendingFileRenameOperationsChecksum : TMD5Digest;
17933: Func ModifyPifFile( const Filename :Str; const CloseOnExit :Bool) :Bool;
17934: Proc RaiseFunctionFailedError( const FunctionName :Str);
17935: Proc RaiseOleError( const FunctionName :Str; const ResultCode : HRESULT);
17936: Proc RefreshEnvironment;
17937: Func ReplaceSystemDirWithSysWow64( const Path :Str) :Str;
17938: Func ReplaceSystemDirWithSysNative( Path :Str; const IsWin64 :Bool) :Str;
17939: Proc UnregisterFont( const FontName, FontFilename :Str);
17940: Func RestartComputer :Bool;
17941: Proc RestartReplace( const DisableFsRedir :Bool; TempFile, DestFile :Str);
17942: Proc SplitNewParamStr( const Index : Int; var AName, AValue :Str);
17943: Proc Win32ErrorMsg( const FunctionName :Str);
17944: Proc Win32ErrorMsgEx( const FunctionName :Str; const Errorcode : DWORD);
17945: Func inForceDirectories(const DisableFsRedir:Boolean;Dir:str):Bool;
17946: //from Func2
17947: //Func inCreateShellLink(const Filename,Description,ShortcutTo,Parameters, WorkingDir:Str;
IconFilename:Str; const IconIndex, ShowCmd : Int; const HotKey : Word; FolderShortcut :Bool; '
17948: //+'const AppUserModelID:Str;const ExcludeFromShowInNewInstall,PreventPinning:Bool):Str;
17949: Proc RegisterTypeLibrary( const Filename :Str);
17950: //Proc UnregisterTypeLibrary( const Filename :Str);
17951: //Func UnpinShellLink( const Filename :Str) :Bool;}
17952: Func getVersionInfoEx3: TOSVersionInfoEx;
17953: Func GetVersionEx3(out verinfo: TOSVersionInfoEx):Bool;
17954: Proc InitOle;;
17955: Func ExpandConst( const S :Str) :Str;
17956: Func ExpandConstEx( const S :Str; const CustomConsts : array of String) :Str;
17957: Func ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividConst:Bool):Str;
17958: Func ExpandConstIfPrefixed( const S :Str) :Str;
17959: Proc LogWindowsVersion;
17960: Func EvalCheck( const Expression :Str) :Bool;
17961: end;
17962:
17963: Proc SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17964: begin
17965: ClassN(CL.FindClass('TOBJECT'),'TResourceDetails;
17966: //TypeS('TResourceDetailsClass', 'class of TResourceDetails;
17967: SIRegister_TResourceModule(CL);
17968: SIRegister_TResourceDetails(CL);
17969: SIRegister_TAnsiResourceDetails(CL);
17970: SIRegister_TUnicodeResourceDetails(CL);
17971: Proc RegisterResourceDetails( resourceClass : TResourceDetailsClass);
17972: Proc UnRegisterResourceDetails( resourceClass : TResourceDetailsClass);
17973: Func ResourceWideCharToStr( var wstr : PWideChar; codePage : Int) :Str;
17974: Proc ResourceStrToWideChar( const s :Str; var p : PWideChar; codePage : Int);
17975: Func ResourceNameToInt( const s :Str) : Int;
17976: Func CompareDetails( p1, p2 : TObject(Pointer)) : Int;
17977: end;
17978:
17979:
17980: Proc SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17981: begin
17982: //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17983: with ClassN(CL.FindClass('TOBJECT'),'TSimpleComPort') do begin
17984: RegisterMethod(Constructor Create;
17985: RegisterMethod(Proc Free;
17986: RegisterMethod(Proc Open( PortNumber : Int; const Parameters:str);
17987: RegisterMethod(Proc WriteString( const S :Str);
17988: RegisterMethod(Proc ReadString( var S :Str);
17989: end;
17990: Ex.: SimpleComPort:= TSimpleComPort.Create;
17991: SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1;
17992: SimpleComPort.WriteString(AsciiChar);
17993: end;
17994:

```

```

17995: Proc SIRegister_Console(CL: TPSPascalCompiler);
17996: begin
17997:   ConstantN('White','LongInt').SetInt( 15); }
17998: // ConstantN('Blink','LongInt').SetInt( 128);
17999:   ('conBW40','LongInt').SetInt( 0);
18000:   ('conCO40','LongInt').SetInt( 1);
18001:   ('conBW80','LongInt').SetInt( 2);
18002:   ('conCO80','LongInt').SetInt( 3);
18003:   ('conMono','LongInt').SetInt( 7);
18004:   ConstantN('conFont8x8','LongInt').SetInt( 256);
18005: //ConstantN('C40','').SetString( C040);
18006: //ConstantN('C80','').SetString( C080);
18007: Func conReadKey : Char;
18008: Func conKeyPressed :Bool;
18009: Proc conGotoXY( X, Y : Smallint);
18010: Func conWhereX : Int;
18011: Func conWhereY : Int;
18012: Proc conTextColor( Color : Byte);;
18013: Func conTextColor1 : Byte;;
18014: Proc conTextBackground( Color : Byte);;
18015: Func conTextBackground1 : Byte;;
18016: Proc conTextMode( Mode : Word);
18017: Proc conLowVideo;
18018: Proc conHighVideo;
18019: Proc conNormVideo;
18020: Proc conClrScr;
18021: Proc conClrEol;
18022: Proc conInsLine;
18023: Proc conDelLine;
18024: Proc conWindow( Left, Top, Right, Bottom : Int);
18025: Func conScreenWidth : Smallint;
18026: Func conScreenHeight : Smallint;
18027: Func conBufferWidth : Smallint;
18028: Func conBufferHeight : Smallint;
18029: Proc InitScreenMode;;
18030: end;
18031:
18032: (*-----*)
18033: Proc SIRegister_testutils(CL: TPSPascalCompiler);
18034: begin
18035:   SIRegister_TNoRefCountObject(CL);
18036: Proc FreeObjects( List : TFPList);
18037: Proc GetMethodList( AObject : TObject; AList : TStringList);;
18038: Proc GetMethodList1( AClass : TClass; AList : TStringList);;
18039: end;
18040:
18041: Proc SIRegister_ToolsUnit(CL: TPSPascalCompiler);
18042: begin
18043:   'MaxDataSet','LongInt').SetInt( 35);
18044: //TypeS('TDBConnectorClass', 'class of TDBConnector;
18045: SIRegister_TDBConnector(CL);
18046: SIRegister_TDBBasicsTestSetup(CL);
18047: SIRegister_TTestDataLink(CL);
18048: 'testValuesCount','LongInt').SetInt( 25);
18049: Proc InitialiseDBConnector;
18050: Proc FreeDBConnector;
18051: Func DateTimeToTimeString( d : tdatetime) :Str;
18052: Func TimeStringToDateTime( d :Str) : TDateTime;
18053: end;
18054:
18055: Proc SIRegister_fpcunit(CL: TPSPascalCompiler);
18056: begin
18057:   SIRegister_EAssertionFailedError(CL);
18058: TypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing );
18059: TypeS('TRunMethod', 'Procedure;
18060: ClassN(CL.FindClass('TOBJECT'),'TTestResult;
18061: SIRegister_TTest(CL);
18062: SIRegister_TAssert(CL);
18063: SIRegister_TTestFailure(CL);
18064: SIRegister_ITestListener(CL);
18065: SIRegister_TTestCase(CL);
18066: //TypeS('TTestCaseClass', 'class of TTestCase;
18067: SIRegister_TTestSuite(CL);
18068: SIRegister_TTestResult(CL);
18069: Func ComparisonMsg( const aExpected:Str; const aActual :Str) :Str;
18070: end;
18071:
18072: Proc SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
18073: begin
18074:   TOBJECT'),'ETCPBuffer;
18075:   TTCPBuffer', 'record Ptr : TObject; Size : Int; Max : Integer'
18076:   +r; Head : Int; Used : Int; end;
18077: 'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
18078: 'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
18079: 'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
18080: 'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
18081: Proc TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSiz:Int;const TCPBufSize:Int;
18082: Proc TCPBufferFinalise( var TCPBuf : TTCPBuffer);
18083: Proc TCPBufferPack( var TCPBuf : TTCPBuffer);

```



```

18084: Proc TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Int);
18085: Proc TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Int);
18086: Proc TCPBufferShrink( var TCPBuf : TTCPBuffer);
18087: Func TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Int) : Pointer;
18088: Proc TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf:Str; const Size : Int);
18089: Func TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Int;
18090: Func TCPBufferPeek( var TCPBuf:TTCPBuffer;var Buf:Str; const Size:Int) : Int;
18091: Func TCPBufferRemove( var TCPBuf:TTCPBuffer;var Buf:Str;const Size:Int) : Int;
18092: Proc TCPBufferClear( var TCPBuf : TTCPBuffer);
18093: Func TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Int) : Int;
18094: Func TCPBufferUsed( const TCPBuf : TTCPBuffer) : Int;
18095: Func TCPBufferEmpty( const TCPBuf : TTCPBuffer) : Bool;
18096: Func TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Int;
18097: Func TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer;
18098: Proc TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Int);
18099: end;
18100:
18101: Proc SIRegister_Glut(CL: TPSPascalCompiler);
18102: begin
18103:   //TypeS('PInt', '^Int // will not work;
18104:   //TypeS('PPChar', '^PChar // will not work;
18105:   ConstantN('GLUT_API_VERSION', 'LongInt').SetInt( 3);
18106:   ConstantN('GLUT_XLIB_IMPLEMENTATION', 'LongInt').SetInt( 12);
18107:   ConstantN('GLUT_RGB', 'LongInt').SetInt( 0);
18108:   ConstantN('GLUT_GAME_MODE_REFRESH_RATE', 'LongInt').SetInt( 5);
18109:   ConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED', 'LongInt').SetInt( 6);
18110:   Proc LoadGlut( const dll :Str);
18111:   Proc FreeGlut;
18112: end;
18113:
18114: Proc SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
18115: begin
18116:   TypeS('TLEDColor', ( ledcGreen, ledcRed, ledcGray );
18117:   Func GetLEDBitmapHandle( Color : TLEDColor; State:Boolean) : THandle;
18118: end;
18119:
18120: Proc SIRegister_SwitchLed(CL: TPSPascalCompiler);
18121: begin
18122:   TLedColor, ( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black );
18123:   TTledState, ( LedOn, LedOff, LedDisabled );
18124:   SIRegister_TSwitchLed(CL);
18125:   //Proc Register;
18126: end;
18127:
18128: Proc SIRegister_FileClass(CL: TPSPascalCompiler);
18129: begin
18130:   TypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
18131:   + 'xisting, fdOpenAlways, fdTruncateExisting );
18132:   TypeS('TFileAccess', '( faRead, faWrite, faReadWrite );
18133:   TypeS('TFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite );
18134:   SIRegister_TCustomFile(CL);
18135:   SIRegister_TFile(CL);
18136:   SIRegister_TMemoryFile(CL);
18137:   SIRegister_TTextFileReader(CL);
18138:   SIRegister_TTextFileWriter(CL);
18139:   SIRegister_TFileMapping(CL);
18140:   SIRegister_EFileError(CL);
18141: end;
18142:
18143: Proc SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
18144: begin
18145:   TypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
18146:   + ', ffaDirectory, ffaArchive, ffaAnyFile );
18147:   TypeS('TAttributeSet', 'set of TFileAttributes;
18148:   SIRegister_TFileSearch(CL);
18149: end;
18150:
18151: Proc SIRegister_uColorFunctions(CL: TPSPascalCompiler);
18152: begin
18153:   TRGBType, 'record RedHex :Str; GreenHex :Str; BlueHex : '
18154:   + 'string; Red : Int; Green : Int; Blue : Int; end;
18155:   Func FadeColor( aColor : Longint; aFade : Int) : Tcolor;
18156:   Func CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:Int):Tcolor;
18157: end;
18158:
18159: Proc SIRegister_uSettings(CL: TPSPascalCompiler);
18160: begin
18161:   Proc SaveOscSettings;
18162:   Proc GetOscSettings;
18163: end;
18164:
18165: Proc SIRegister_cyDebug(CL: TPSPascalCompiler);
18166: begin
18167:   TProcProcessEvent, 'Proc ( Sender : TObject; Index : Int);
18168:   RecProcess, 'record Name : ShortString; DurationMs :Card; '
18169:   FirstDurationMs :Card; LastDurationMs :Card; MinDurationMs : Int'
18170:   64; MaxDurationMs :Card; MarksCount : Int; ArrayMarks : array of '
18171:   Cardinal; EnterCount : Int; ExitCount : Int; end;
18172:   SIRegister_TcyDebug(CL);

```

```

18173: end;
18174:
18175: (*-----*)
18176: Proc SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
18177: begin
18178:   TCopyFileResult, (cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError );
18179:   TCopyFileExists, '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent );
18180:   TCopyFileNotExists, '( fnDoNothing, fnCopy, fnCopyForceDir );
18181:   SIRegister_TDestinationOptions(CL);
18182:   TProcCustomCopyFileEvent, Procedure(Sender:TObject;var CopyFileResult:TCopyFileResult);
18183:   TProcOnCopyFileProgressEvent, Procedure(Sender:TObject;FileBytes,TransferredBytes:int64;PercentDone:Int64);
18184:   TProcOnCustomSetFileDestination, 'Proc (Sender:TObject;var FileName:str);
18185:   SIRegister_TcyCopyFiles(CL);
18186:   Func cyCopyFile(FromFile,ToFile:str; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
   ResetAttr:boolean): TCopyFileResult;
18187:   Func cyCopyFileEx(FromFile,ToFile:str;FileExists:TCopyFileExists;FileNotExists
   TCopyFileNotExists;ResetAttr:bool;aProgressBar:TProgressBar):TCopyFileResult;
18188:   Func cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters :Str; ArchiveFiles,
   ReadOnlyFiles, HiddenFiles,
   SystemFiles:TcyFileAttributeMode;FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;SubFolders,
   ResetAttributes:Boolean):Int;
18189: end;
18190:
18191: Proc SIRegister_cySearchFiles(CL: TPSPascalCompiler);
18192: begin
18193:   TypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth );
18194:   SIRegister_TcyFileAttributes(CL);
18195:   SIRegister_TSearchRecInstance(CL);
18196:   TOption, '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude );
18197:   TOptions, 'set of TOption;
18198:   TSearchState, '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting );
18199:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
   Accept:bool;
18200:   TProcOnValidateDirectoryEvent, Procedure(Sender:TObject;Directry:str;var Accept:bool);
18201:   SIRegister_TcyCustomSearchFiles(CL);
18202:   SIRegister_TcySearchFiles(CL);
18203:   Func FileNameRespondToMask( aFileName :Str; aMask :Str) :Bool;
18204:   Func IsCpyFolder( aSRec : TSearchrec) :Bool;
18205: end;
18206:
18207: Proc SIRegister_jcontrolutils(CL: TPSPascalCompiler);
18208: begin
18209:   Func jCountChar( const s :Str; ch : char) : Int;
18210:   Proc jSplit( const Delimiter : char; Input :Str; Strings : TStrings);
18211:   Func jNormalizeDate(const Value:str;theValue: TDateTime;const theFormat:str):str;
18212:   Func jNormalizeTime(const Value:str;theValue:TTime;const theFormat:str) :Str;
18213:   Func jNormalizeDateTime(const Val:str;theValue:TDateTime;const theFormat:str):str;
18214:   Func jNormalizeDateSeparator( const s :Str) :Str;
18215:   Func jIsValidDateString( const Value :Str) :Bool;
18216:   Func jIsValidTimeString( const Value :Str) :Bool;
18217:   Func jIsValidDateTimeString( const Value :Str) :Bool;
18218: end;
18219:
18220: Proc SIRegister_kcMapViewier(CL: TPSPascalCompiler);
18221: begin
18222:   ClassN(CL.FindClass('TOBJECT'),'TMapViewier;
18223:   TypeS('TMapSource', ( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
18224:   +gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
18225:   +, msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
18226:   +EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
18227:   +hooSatellite, msYahooHybrid,msOviNormal,msOviSatellite, msOviHybrid, msOviPhysical );
18228:   TArea, 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end;
18229:   TRealArea, 'record top: Extended; left:Extended; bottom: Extended; right: Extended; end;
18230:   TIntPoint, 'record X : Int64; Y : Int64; end;
18231:   TkRealPoint, 'record X : Extended; Y : Extended; end;
18232:   TOnBeforeDownloadEvent, 'Procedure(Url:str; str:TStream; var CanHandle:Bool);
18233:   TOnAfterDownloadEvent, 'Procedure( Url :Str; str : TStream);
18234:   SIRegister_TCustomDownloadEngine(CL);
18235:   SIRegister_TCustomGeolocationEngine(CL);
18236:   SIRegister_TMapViewier(CL);
18237: end;
18238:
18239: Proc SIRegister_cparserrutils(CL: TPSPascalCompiler);
18240: begin
18241:   (*Func isFunc( name : TNamePart) :Bool;*)
18242:   Func isUnnamedFunc( name : TNamepart) :Bool;
18243:   Func isPtrToFunc( name : TNamePart) :Bool;
18244:   Func isFuncRetFuncPtr( name : TNamePart) :Bool;
18245:   Func isPtrToFuncRetFuncPtr( name : TNamePart) :Bool;
18246:   Func GetFuncParam( name : TNamePart) : TNamePart;
18247:   Func isArray( name : TNamePart) :Bool;
18248:   Func GetArrayPart( name : TNamePart) : TNamePart;
18249:   Func GetIdFromPart( name : TNamePart) : Ansistr;
18250:   Func GetIdPart( name : TNamePart) : TNamePart;
18251:   Func isNamePartPtrToFunc( part : TNamePart) :Bool;
18252:   Func isAnyBlock( part : TNamePart) :Bool;*)
18253:   TypeS('TLineInfo', record linestart : Int; lineend : Int; end;
18254:   SIRegister_TLineBreaker(CL);
18255:   TypeS('TNameKind', 'Int;

```

```

18256:   ClassN(CL.FindClass('TOBJECT'),'TNamePart;
18257:   //Types('TFuncParam', 'record prmtyp : TEntity; name : TNamePart; end;
18258: Func SphericalMod( X : Extended) : Extended;
18259: Func cSign( Value : Extended) : Extended;
18260: Func LimitFloat( const eValue, eMin, eMax : Extended) : Extended;
18261: Func AngleToRadians( iAngle : Extended) : Extended;
18262: Func RadiansToAngle( eRad : Extended) : Extended;
18263: Func Cross180( iLong : Double) : Bool;
18264: Func Mod180( Value : Int) : Int;
18265: Func Mod180Float( Value : Extended) : Extended;
18266: Func MulDivFloat( a, b, d : Extended) : Extended;
18267: Func LongDiff( iLong1, iLong2 : Double) : Double;
18268: Proc Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap);
18269: Func Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap;
18270: Func FixFilePath( const Inpath, CheckPath :Str) :Str;
18271: Func UnFixFilePath( const Inpath, CheckPath :Str) :Str;
18272: Proc FillStringList( sl : TStringList; const aText :Str);
18273: end;
18274:
18275: Proc SIRegister_LedNumber(CL: TPSPascalCompiler);
18276: begin
18277:   TLedSegmentSize', 'Int;
18278:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised );
18279:   SIRegister_TCustomLEDNumber(CL);
18280:   SIRegister_TLEDNumber(CL);
18281: end;
18282:
18283: Proc SIRegister_StStrL(CL: TPSPascalCompiler);
18284: begin
18285:   TypeS('LStrRec', record AllocSize:Longint;RefCount : Longint; Length : Longint; end;
18286:   TypeS('AnsiChar', 'Char;
18287:   TypeS('BTable', array[0..255] of Byte; //!!!
18288:   Func HexBL( B : Byte) : Ansistr;
18289:   Func HexWL( W : Word) : Ansistr;
18290:   Func HexLL( L : LongInt) : Ansistr;
18291:   Func HexPtrL( P : _Pointer) : Ansistr;
18292:   Func BinaryBL( B : Byte) : Ansistr;
18293:   Func BinaryWL( W : Word) : Ansistr;
18294:   Func BinaryLL( L : LongInt) : Ansistr;
18295:   Func OctalBL( B : Byte) : Ansistr;
18296:   Func OctalWL( W : Word) : Ansistr;
18297:   Func OctalLL( L : LongInt) : Ansistr;
18298:   Func Str2Int16L( const S : Ansistr; var I : SmallInt) :Bool;
18299:   Func Str2WordL( const S : Ansistr; var I : Word) :Bool;
18300:   Func Str2LongL( const S : Ansistr; var I : LongInt) :Bool;
18301:   Func Str2RealL( const S : Ansistr; var R : Double) :Bool;
18302:   Func Str2RealL( const S : Ansistr; var R : Real) :Bool;
18303:   Func Str2ExtL( const S : Ansistr; var R : Extended) :Bool;
18304:   Func Long2StrL( L : LongInt) : Ansistr;
18305:   Func Real2StrL( R : Double; Width : Byte; Places : ShortInt) : Ansistr;
18306:   Func Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : Ansistr;
18307:   Func ValPrepL( const S : Ansistr) : Ansistr;
18308:   Func CharStrL( C : Char; Len :Card) : Ansistr;
18309:   Func PadChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18310:   Func PadLL( const S : Ansistr; Len :Card) : Ansistr;
18311:   Func LeftPadChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18312:   Func LeftPadL( const S : Ansistr; Len :Card) : Ansistr;
18313:   Func TrimLeadL( const S : Ansistr) : Ansistr;
18314:   Func TrimTrailL( const S : Ansistr) : Ansistr;
18315:   Func TrimL( const S : Ansistr) : Ansistr;
18316:   Func TrimSpacesL( const S : Ansistr) : Ansistr;
18317:   Func CenterChL( const S : Ansistr; C : AnsiChar; Len :Card) : Ansistr;
18318:   Func CenterL( const S : Ansistr; Len :Card) : Ansistr;
18319:   Func EntabL( const S : Ansistr; TabSize : Byte) : Ansistr;
18320:   Func DetabL( const S : Ansistr; TabSize : Byte) : Ansistr;
18321:   Func ScrambleL( const S, Key : Ansistr) : Ansistr;
18322:   Func SubstituteL( const S, FromStr, ToStr : Ansistr) : Ansistr;
18323:   Func FilterL( const S, Filters : Ansistr) : Ansistr;
18324:   Func CharExistsL( const S : Ansistr; C : AnsiChar) :Bool;
18325:   Func CharCountL( const S : Ansistr; C : AnsiChar) :Card;
18326:   Func WordCountL( const S, WordDelims : Ansistr) :Card;
18327:   Func WordPositionL(N:Card;const S,WordDelims:Ansistr;var Pos:Card):Bool;
18328:   Func ExtractWordL( N :Card; const S, WordDelims : Ansistr) : Ansistr;
18329:   Func AsciiCountL( const S, WordDelims : Ansistr; Quote : AnsiChar) :Card;
18330:   Func AsciiPositionL(N:Card;const S,WordDelims:Ansistr;Quote:AnsiChr;var Pos:Card):Bool;
18331:   Func ExtractAsciiL(N:Card;const S,WordDelims:Ansistr;Quote:AnsiChar):Ansistr;
18332:   Proc WordWrapL(const InSt:Ansistr;var OutSt,Overlap:Ansistr;Margin:Card;PadToMargin:Bool;
18333:   Proc WordWrap(const InSt:Ansistr;var OutSt,Overlap:Ansistr;Margin:Card;PadToMargin:Bool;
18334:   Func CompStringL( const S1, S2 : Ansistr) : Int;
18335:   Func CompUCStringL( const S1, S2 : Ansistr) : Int;
18336:   Func SoundexL( const S : Ansistr) : Ansistr;
18337:   Func MakeLetterSetL( const S : Ansistr) : Longint;
18338:   Proc BMMakeTableL( const MatchString : Ansistr; var BT : BTable);
18339:   Func BMSearchL(var Buffer,BufLength:Card;var BT:BTable;const MatchString:Ansistr;var Pos:Card):Bool;
18340:   Func BMSearchUCL( var Buffer, BufLength :Card; var BT: BTable; const MatchString: Ansistr; var Pos:Card)
:Bool;
18341:   Func DefaultExtensionL( const Name, Ext : Ansistr) : Ansistr;
18342:   Func ForceExtensionL( const Name, Ext : Ansistr) : Ansistr;
18343:   Func JustFilenameL( const PathName : Ansistr) : Ansistr;

```

```

18344: Func JustNameL( const PathName : Ansistr) : Ansistr;
18345: Func JustExtensionL( const Name : Ansistr) : Ansistr;
18346: Func JustPathNameL( const PathName : Ansistr) : Ansistr;
18347: Func AddBackSlashL( const DirName : Ansistr) : Ansistr;
18348: Func CleanPathNameL( const PathName : Ansistr) : Ansistr;
18349: Func HasExtensionL( const Name : Ansistr; var DotPos :Card) :Bool;
18350: Func CommaizeL( L : LongInt) : Ansistr;
18351: Func CommaizeChL( L : LongInt; Ch : AnsiChar) : Ansistr;
18352: Func FloatFormL(const Mask:Ansistr;R:TstFloat;const LtCurr,RtCurr:Ansistr;Sep,DecPt:Char):Ansistr;
18353: Func LongIntFormL(const Mask:Ansistr;L:LongInt;const LtCurr,RtCurr:Ansistr;Sep:Char):Ansistr;
18354: Func StrChPosL( const P : Ansistr; C : AnsiChar; var Pos :Card) :Bool;
18355: Func StrStPosL( const P, S : Ansistr; var Pos :Card) :Bool;
18356: Func StrStCopyL( const S : Ansistr; Pos, Count :Card) : Ansistr;
18357: Func StrChInsertL( const S : Ansistr; C : AnsiChar; Pos :Card) : Ansistr;
18358: Func StrStInsertL( const S1, S2 : Ansistr; Pos :Card) : Ansistr;
18359: Func StrChDeleteL( const S : Ansistr; Pos :Card) : Ansistr;
18360: Func StrStDeleteL( const S : Ansistr; Pos, Count :Card) : Ansistr;
18361: Func ContainsOnlyL( const S, Chars : Ansistr; var BadPos :Card) :Bool;
18362: Func ContainsOtherThanL(const S,Chars: Ansistr; var BadPos :Card):Bool;
18363: Func CopyLeftL( const S : Ansistr; Len :Card) : Ansistr;
18364: Func CopyMidL( const S : Ansistr; First, Len :Card) : Ansistr;
18365: Func CopyRightL( const S : Ansistr; First :Card) : Ansistr;
18366: Func CopyRightAbsL( const S : Ansistr; NumChars :Card) : Ansistr;
18367: Func CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Card;var SubString:AString):Bool;
18368: Func CopyFromToWordL(const S,WordDelims,Word1,Word2:Ansistr;N1,N2:Card;var SubString:Ansistr):Bool;
18369: Func CopyWithinL( const S, Delimiter : Ansistr; Strip :Bool) : Ansistr;
18370: Func DeleteFromNthWordL(const S, WordDelims:Ansistr;const AWord: Ansistr; N:Card; var SubString:
Ansistr): Bool;
18371: Func DeleteFromToWordL(const S,WordDelims,Word1,Word2:Ansistr;N1,N2:Card;var SubString:Ansistr):Bool;
18372: Func DeleteWithinL( const S, Delimiter : Ansistr) : Ansistr;
18373: Func ExtractTokensL(const S,Delims:Ansistr;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Card;
18374: Func IsChAlphaL( C : AnsiChar) :Bool;
18375: Func IsChNumericL( C : AnsiChar; const Numbers : Ansistr) :Bool;
18376: Func IsChAlphaNumericL( C : AnsiChar; const Numbers : Ansistr) :Bool;
18377: Func IsStrAlphaL( const S : Ansistr) :Bool;
18378: Func IsStrNumericL( const S, Numbers : Ansistr) :Bool;
18379: Func IsStrAlphaNumericL( const S, Numbers : Ansistr) :Bool;
18380: Func KeepCharsL( const S, Chars : Ansistr) : Ansistr;
18381: Func LastWordL(const S,WordDelims, AWord:Ansistr; var Position:Card):Bool;
18382: Func LastWordAbsL( const S, WordDelims:Ansistr;var Position:Card):Bool;
18383: Func LastStringL( const S, AString : Ansistr; var Position :Card):Bool;
18384: Func LeftTrimCharsL( const S, Chars : Ansistr) : Ansistr;
18385: Func ReplaceWordL(const S,WordDelims,OldWord,NewWord:Ansistr;N:Card;var Replacements:Card):Ansistr;
18386: Func ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:Ansistr;var Replacements:Card):Ansistr;
18387: Func ReplaceStringL(const S,OldString,NewString:Ansistr;N:Card;var Replacements:Card):Ansistr;
18388: Func ReplaceStringAllL(const S,OldString,NewString:Ansistr; var Replacements:Card):Ansistr;
18389: Func RepeatStringL(const RepeatString:Ansistr;var Repetitions:Card;MaxLen:Card):Ansistr;
18390: Func RightTrimCharsL( const S, Chars : Ansistr) : Ansistr;
18391: Func StrWithinL(const S,SearchStr:str;Start:Card;var Position:Card): bool;
18392: Func TrimCharsL( const S, Chars : Ansistr) : Ansistr;
18393: Func WordPosL(const S,WordDelims,AWord:Ansistr;N:Card;var Position:Card):Bool;
18394: Func WordPos( const S,WordDelims,AWord:Ansistr;N:Card;var Position:Card):Bool;
18395: end;
18396:
18397: Proc SIRegister_pwnative_out(CL: TPSPascalCompiler);
18398: begin
18399:   ConstantN('STDIN','LongInt').SetInt( 0);
18400:   ('STDOUT','LongInt').SetInt( 1);
18401:   ('STDERR','LongInt').SetInt( 2);
18402: Proc NativeWrite( s : astr);;
18403: Proc NativeWrite1( PString : PChar);;
18404: Proc NativeWrite2( Buffer : PChar; NumChars :Card);;
18405: Proc NativeWriteLn( s : astr);;
18406: Proc NativeWriteLn1;;
18407: end;
18408:
18409: Proc SIRegister_synwrap1(CL: TPSPascalCompiler);
18410: begin
18411:   TypeS(TSynwInfo', 'record Err : byte; UrlHtml : Ansistr; ErrResponse
: Int; UltimateURL : Ansistr; Headers : Ansistr; end;
18412:   TypeS('TUrlInfo', 'record Err : byte; UltimateURL :Str; end;
18413:   Func GetHttpFile(const Url,UserAgent,Outfile:str; verbose :Bool):TSynwInfo;
18414:   Func GetHttpFile1( const Url, UserAgent, Outfile :Str) : TSynwInfo;
18415:   Func GetHttpFile2( const Url, Outfile :Str) : TSynwInfo;;
18416:   Func GetHttpFile3( const Url, outfile :Str; verbose :Bool) : TSynwInfo;
18417:   Func GetHtm( const Url :Str) :Str;;
18418:   Func GetHtml( const Url, UserAgent :Str) :Str;;
18419:   Func GetUrl( const Url :Str; verbose :Bool) : TSynwInfo;;
18420:   Func GetUrl1( const Url, useragent :Str) : TSynwInfo;;
18421:   Func GetUrl2( const Url :Str) : TSynwInfo;;
18422:   Func GetUrl3(const Url:str;const http THTPSend; verbose :Bool): TUrlInfo;
18423:   Func GetUrl4( const Url :Str; const http : THTPSend) : TUrlInfo;;
18424:   Proc StrToStream( s :Str; strm : TMemoryStream);
18425:   Func StrLoadStream( strm : TStream) :Str;
18426: end;
18427:
18428:
18429: Proc SIRegister_HTMLUtil(CL: TPSPascalCompiler);
18430: begin
18431:   Func GetVal( const tag, attribname_ci :Str) :Str;

```

```

18432: Func GetTagName( const Tag :Str) :Str;
18433: Func GetUpTagName( const tag :Str) :Str;
18434: Func GetNameValPair( const tag, attribname_ci :Str) :Str;
18435: Func GetValFromNameVal( const namevalpair :Str) :Str;
18436: Func GetNameValPair_cs( const tag, attribname :Str) :Str;
18437: Func GetVal_JAMES( const tag, attribname_ci :Str) :Str;
18438: Func GetNameValPair_JAMES( const tag, attribname_ci :Str) :Str;
18439: Func CopyBuffer( StartIndex : PChar; Len : Int) :Str;
18440: Func Ucase( s :Str) :Str;
18441: end;
18442:
18443: Proc SIRegister_pwmain(CL: TPSPascalCompiler);
18444: begin
18445:   ConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT;
18446:   EXPIRED_COOKIE','String').SetString( 'Mon, 01 Jan 2001 12:00:00 GMT;
18447:   'SECURE_OFF','LongInt').SetInt( 0);
18448:   'SECURE_ON','LongInt').SetInt( 2);
18449:   'SECURE_FILTER','LongInt').SetInt( 3);
18450:   THandle or DWord!
18451:   // astr = Ansistr;
18452:   TypeS('pastr', 'Ansistr;
18453:   TypeS('TFilterFunc', 'function(const s: pastr): pastr;;
18454:   uses pwinit at begin
18455:   //type TFilterFunc = function(const s: astr): astr;
18456:   Demo: ..\maxbox3\examples2\519_powtils.txt
18457:   //ConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false);
18458:   //ConstantN('CASE_IGNORE','Boolean')BoolToStr( false);
18459:   Proc pwInit;
18460:   Proc OffReadln;
18461:   Func Lcase( const s : pastr) : pastr;
18462:   Func Ucase( const s : pastr) : pastr;
18463:   Func CountPostVars : longword;
18464:   Func GetPostVar( const name : pastr) : pastr;;
18465:   Func GetPostVarI( const name : pastr; filter : TFilterFunc) : pastr;;
18466:   Func GetPostVar_S( const name : pastr; Security : Int) : pastr;
18467:   Func GetPostVar_SF( const name : pastr; Security : Int) : pastr;
18468:   Func GetPostVarAsFloat( const name : pastr) : double;
18469:   Func GetPostVarAsInt( const name : pastr) : longint;
18470:   Func GetPostVar_SafeHTML( const name : pastr) : pastr;
18471:   Func FetchPostVarName( idx : longword) : pastr;
18472:   Func FetchPostVarVal( idx : longword) : pastr;;
18473:   Func FetchPostVarVall( idx : longword; filter : TFilterFunc) : pastr;;
18474:   Func FetchPostVarName_S( idx : longword; Security : Int) : pastr;
18475:   Func FetchPostVarVal_S( idx : longword; Security : Int) : pastr;
18476:   Func IsPostVar( const name : pastr) :Bool;
18477:   Func CountAny : longword;
18478:   Func GetAny( const name : pastr) : pastr;;
18479:   Func GetAnyI( const name : pastr; filter : TFilterFunc) : pastr;;
18480:   Func GetAny_S( const name : pastr; Security : Int) : pastr;
18481:   Func GetAnyAsFloat( const name : pastr) : double;
18482:   Func GetAnyAsInt( const name : pastr) : longint;
18483:   Func IsAny( const name : pastr) : byte;
18484:   Func CountCookies : longword;
18485:   Func FetchCookieName( idx : longword) : pastr;
18486:   Func FetchCookieVal( idx : longword) : pastr;;
18487:   Func FetchCookieVall( idx : longword; filter : TFilterFunc) : pastr;;
18488:   Func GetCookie( const name : pastr) : pastr;;
18489:   Func GetCookieI( const name : pastr; filter : TFilterFunc) : pastr;;
18490:   Func GetCookieAsFloat( const name : pastr) : double;
18491:   Func GetCookieAsInt( const name : pastr) : longint;
18492:   Func IsCookie( const name : pastr) :Bool;
18493:   Func SetCookie( const name, value : pastr) :Bool;
18494:   Func SetCookieAsFloat( const name : pastr; value : double) :Bool;
18495:   Func SetCookieAsInt( const name : pastr; value : longint) :Bool;
18496:   Func SetCookieEx( const name, value, path, domain, expiry : pastr) :Bool;
18497:   Func SetCookieAsFloatEx(const name:pastr;value:double;const path,domain,expiry:pastr):bool;
18498:   Func SetCookieAsIntEx(const name:pastr;value:longint;const path,domain,expiry:pastr):bool;
18499:   Func UnsetCookie( const name : pastr) :Bool;
18500:   Func UnsetCookieEx( const name, path, domain : pastr) :Bool;
18501:   Func FilterHtml( const input : pastr) : pastr;
18502:   Func FilterHtml_S( const input : pastr; security : Int) : pastr;
18503:   Func TrimBadChars( const input : pastr) : pastr;
18504:   Func TrimBadFile( const input : pastr) : pastr;
18505:   Func TrimBadDir( const input : pastr) : pastr;
18506:   Func TrimBad_S( const input : pastr; security : Int) : pastr;
18507:   Func CountHeaders : longword;
18508:   Func FetchHeaderName( idx : longword) : pastr;
18509:   Func FetchHeaderVal( idx : longword) : pastr;
18510:   Func GetHeader( const name : pastr) : pastr;
18511:   Func IsHeader( const name : pastr) :Bool;
18512:   Func SetHeader( const name, value : pastr) :Bool;
18513:   Func UnsetHeader( const name : pastr) :Bool;
18514:   Func PutHeader( const header : pastr) :Bool;
18515:   Proc OutI( const s : pastr);
18516:   Proc OutLn( const s : pastr);
18517:   Proc OutA( args : array of const);
18518:   Proc OutF( const s : pastr);
18519:   Proc OutLnF( const s : pastr);
18520:   Proc OutFF( const s : pastr);

```



```

18521: Proc OutF_FI( const s : pastr; HTMLFilter :Bool);
18522: Proc OutLnFF( const s : pastr);
18523: Proc OutLnF_FI( const s : pastr; HTMLFilter :Bool);
18524: Func FileOut( const fname : pastr) : word;
18525: Func ResourceOut( const fname : pastr) : word;
18526: Proc BufferOut( const buff, len : LongWord);
18527: Func TemplateOut( const fname : pastr; HtmlFilter :Bool) : word;;
18528: Func TemplateOut1( const fname : pastr) : word;;
18529: Func TemplateOut2( const fname : pastr; filter : TFilterFunc) : word;;
18530: Func TemplateOut3( const fname : pastr; HtmlFilter :Bool) : word;;
18531: Func TemplateRaw( const fname : pastr) : word;
18532: Func Fmt( const s : pastr) : pastr;;
18533: Func Fmt1( const s : pastr; filter : TFilterFunc) : pastr;;
18534: Func FmtFilter( const s : pastr) : pastr;
18535: Func Fmt_SF(const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecur,TrimSecurity:int):pastr;
18536: Func Fmt_SF1(const s:pastr;HTMLFilter:bool;FilterSecurity,TrimSecurity:Int):pastr;;
18537: Func CountRtiVars : longword;
18538: Func FetchRtiName( idx : longword) : pastr;
18539: Func FetchRtiVal( idx : longword) : pastr;
18540: Func GetRti( const name : pastr) : pastr;
18541: Func GetRtiAsFloat( const name : pastr) : double;
18542: Func GetRtiAsInt( const name : pastr) : longint;
18543: Func IsRti( const name : pastr) :Bool;
18544: Proc SetRTI( const name, value : pastr);
18545: Func FetchUpfileName( idx : longword) : pastr;
18546: Func GetUpfileName( const name : pastr) : pastr;
18547: Func GetUpfileSize( const name : pastr) : longint;
18548: Func GetUpfileType( const name : pastr) : pastr;
18549: Func CountUpfiles : longword;
18550: Func IsUpfile( const name : pastr) :Bool;
18551: Func SaveUpfile( const name, fname : pastr) :Bool;
18552: Func CountVars : longword;
18553: Func FetchVarName( idx : longword) : pastr;
18554: Func FetchVarVal( idx : longword) : pastr;;
18555: Func FetchVarVal1( idx : longword; filter : TFilterFunc) : pastr;;
18556: Func GetVar( const name : pastr) : pastr;;
18557: Func GetVar1( const name : pastr; filter : TFilterFunc) : pastr;;
18558: Func GetVar_S( const name : pastr; security : Int) : pastr;
18559: Func GetVarAsFloat( const name : pastr) : double;
18560: Func GetVarAsInt( const name : pastr) : longint;
18561: Proc SetVar( const name, value : pastr);
18562: Proc SetVarAsFloat( const name : pastr; value : double);
18563: Proc SetVarAsInt( const name : pastr; value : longint);
18564: Func IsVar( const name : pastr) : byte;
18565: Proc UnsetVar( const name : pastr);
18566: Func LineEndToBR( const s : pastr) : pastr;
18567: Func RandomStr( len : longint) : pastr;
18568: Func XorCrypt( const s : pastr; key : byte) : pastr;
18569: Func CountCfgVars : longword;
18570: Func FetchCfgVarName( idx : longword) : pastr;
18571: Func FetchCfgVarVal( idx : longword) : pastr;
18572: Func IsCfgVar( const name : pastr) :Bool;
18573: Func SetCfgVar( const name, value : pastr) :Bool;
18574: Func GetCfgVar( const name : pastr) : pastr;
18575: Proc ThrowErr( const s : pastr);
18576: Proc ThrowWarn( const s : pastr);
18577: Proc ErrWithHeader( const s : pastr);
18578: TypeS(TWebVar', 'record name : pastr; value : pastr; end;
18579: TypeS(TWebVars', 'array of TWebVar;
18580: Func iUpdateWebVar( var webv:TWebVars; const name,value:pastr; upcased:boolean):boolean;
18581: Func iAddWebCfgVar( const name, value : pastr) :Bool;
18582: Proc iAddWebVar( var webv : TWebVars; const name, value : pastr);
18583: Proc iSetRTI( const name, value : pastr);
18584: Func iCustomSessUnitSet :Bool;
18585: Func iCustomCfgUnitSet :Bool;
18586: end;
18587:
18588: Proc SIRegister_W32VersionInfo(CL: TPSPascalCompiler);
18589: begin
18590:   SIRegister_TProjectVersionInfo(CL);
18591:   Func MSLanguageToHex( const s :Str) :Str;
18592:   Func MSHexToLanguage( const s :Str) :Str;
18593:   Func MSCharacterSetToHex( const s :Str) :Str;
18594:   Func MSHexToCharacterSet( const s :Str) :Str;
18595:   Func MSLanguages : TStringList;
18596:   Func MSHexLanguages : TStringList;
18597:   Func MSCharacterSets : TStringList;
18598:   Func MSHexCharacterSets : TStringList;
18599: end;
18600:
18601: Proc SIRegister_IpUtils(CL: TPSPascalCompiler);
18602: begin
18603:   TIpHandle', 'Cardinal;
18604:   TIpMD5StateArray', 'array[0..3] of DWORD;;
18605:   TypeS('TIpMD5CountArray', 'array[0..1] of DWORD;;
18606:   TIpMD5ByteBuf', 'array[0..63] of Byte;;
18607:   TIpMD5LongBuf', 'array[0..15] of DWORD;;
18608:   TIpMD5Digest', 'array[0..15] of Byte;;
18609:   TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther ) ;

```

```

18610: TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5'
18611: + 'CountArray; ByteBuf : TIpMD5ByteBuf; end;
18612: ClassN(CL.FindClass('TOBJECT'),'EIpBaseException;
18613: ClassN(CL.FindClass('TOBJECT'),'EIpAccessException;
18614: ClassN(CL.FindClass('TOBJECT'),'EIpHtmlException;
18615: SIRegister_TIpBaseAccess(CL);
18616: SIRegister_TIpBasePersistent(CL);
18617: //TIpComponentClass', 'class of TIpBaseComponent;
18618: SIRegister_TIpBaseComponent(CL);
18619: SIRegister_TIpBaseWinControl(CL);
18620: Func InClassA( Addr : LongInt ) :Bool;
18621: Func InClassB( Addr : LongInt ) :Bool;
18622: Func InClassC( Addr : LongInt ) :Bool;
18623: Func InClassD( Addr : LongInt ) :Bool;
18624: Func InMulticast( Addr : LongInt ) :Bool;
18625: Func IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar ) : DWORD;
18626: Func IpCompStruct( const S1, S2, Size :Card ) : Int;
18627: Func IpMaxInt( A, B : Int ) : Int;
18628: Func IpMinInt( A, B : Int ) : Int;
18629: Proc IpSafeFree( var Obj: TObject);
18630: Func IpShortVersion :Str;
18631: Func IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD ) : DWORD;
18632: Func IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD ) : DWORD;
18633: Func IpInternetSumOfFile( const FileName :Str ) : DWORD;
18634: Func MD5SumOfFile( const FileName :Str ) :Str;
18635: Func MD5SumOfStream( Stream : TStream ) :Str;
18636: Func MD5SumOfStreamDigest( Stream : TStream ) : TIpMD5Digest;
18637: Func MD5SumOfString( const S :Str ) :Str;
18638: Func MD5SumOfStringDigest( const S :Str ) : TIpMD5Digest;
18639: Func SafeYield : LongInt;
18640: Func AllTrimSpaces( Strng :Str ) :Str;
18641: Func IpCharPos( C : AnsiChar; const S :Str ) : Int;
18642: Func CharPosIdx( C : AnsiChar; const S :Str; Idx : Int ) : Int;
18643: Func NthCharPos( C : AnsiChar; const S :Str; Nth : Int ) : Int;
18644: Func RCharPos( C : AnsiChar; const S :Str ) : Int;
18645: Func RCharPosIdx( C : AnsiChar; const S :Str; Idx : Int ) : Int;
18646: Func RNthCharPos( C : AnsiChar; const S :Str; Nth : Int ) : Int;
18647: Func IpRPos( const Substr :Str; const S :Str ) : Int;
18648: Func IpPosIdx( const SubStr, S :Str; Idx : Int ) : Int;
18649: ACharSet', 'set of AnsiChar;
18650: TIpAddrRec', 'record Scheme:str; UserName :Str; Password string; Authority :Str;
18651: Port:str;Path:str; Fragment:Str; Query :Str; QueryDelim : AnsiChar; end;
18652: Proc Initialize( var AddrRec : TIpAddrRec);
18653: Proc Finalize( var AddrRec : TIpAddrRec);
18654: Func ExtractEntityName( const NamePath :Str ) :Str;
18655: Func ExtractEntityPath( const NamePath :Str ) :Str;
18656: Func IpParseURL( const URL :Str; var Rslt : TIpAddrRec ) :Bool;
18657: Func BuildURL( const OldURL, NewURL :Str ) :Str;
18658: Func PutEscapes( const S :Str; EscapeSet : ACharSet ) :Str;
18659: Func RemoveEscapes( const S :Str; EscapeSet : ACharSet ) :Str;
18660: Proc SplitParams( const Params :Str; Dest : TStrings);
18661: Func NetToDOSPath( const PathStr :Str ) :Str;
18662: Func DOSToNetPath( const PathStr :Str ) :Str;
18663: Proc SplitHttpResponse( const S :Str; var V, MsgID, Msg :Str);
18664: Proc FieldFix( Fields : TStrings);
18665: Func AppendSlash( APath :Str ) :Str;
18666: Func RemoveSlash( APath :Str ) :Str;
18667: Func GetParentPath( const Path :Str ) :Str;
18668: Func GetLocalContent( const TheFileName :Str ) :Str;
18669: Func IPDirExists( Dir :Str ) :Bool;
18670: Func GetTemporaryFile( const Path :Str ) :Str;
18671: Func GetTemporaryPath :Str;
18672: Func AppendBackSlash( APath :Str ) :Str;
18673: Func IpRemoveBackSlash( APath :Str ) :Str;
18674: Func INetDateStrToDateTime( const DateStr :Str ) : TDateTime;
18675: Func DateTimeToINetDateTimeStr( DateTime : TDateTime ) :Str;
18676: Func IpTimeZoneBias : Int;
18677: Proc SplitCookieFields( const Data :Str; Fields : TStrings);
18678: end;
18679:
18680: Proc SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18681: begin
18682:   TypeS('TPOStyle', '( postStandard, postPropName, postFull );
18683:   Proc Lrt2Po( const LRTFile :Str; POStyle : TPOStyle);
18684:   Proc CombinePoFiles( SL : TStrings; const FName :Str);
18685: end;
18686:
18687: Proc SIRegister_GPS(CL: TPSPascalCompiler);
18688: begin
18689:   ConstantN('MAX_SATS','LongInt').SetInt( 12);
18690:   GPMSG_START','String').SetString( '$;
18691:   GPMSG_STOP','String').SetString( '*;
18692:   SEC_BETWEEN_SEG','LongInt').SetInt( 5);
18693:   TypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor
18694:   + 'tint; Azimut : Smallint; SignLevel : Smallint; end;
18695:   TypeS('TSatellites', 'array[1..12] of TSatellite;
18696:   //TSatellites = array[1..MAX_SATS] of TSatellite;
18697:   TGPSSatEvent,Procedure(Sender: TObject; NbSat, NbSatUse: Shortint; Sats: TSatellites);
18698:   TGPSSatData', 'record Latitude : Double; Longitude : Double; Heigh

```

```

18699:   tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid : Bool; '
18700:   NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end;
18701:   TypeS('TGPSSDataEvent', Proc ( Sender : TObject; GPSSData : TGPSSData);
18702:   TypeS('TMsgGP', ( msgGP,msgGPGA,msgGPGLL,msgGPGRMA, msgGPGRMC, msgGPZDA );
18703:   TypeS('TSpeedUnit', ( suKilometre, suMile, suNauticalMile );
18704:   SIRegister_TGPSLink(CL);
18705:   SIRegister_TCustomGPS(CL);
18706:   SIRegister_TGPS(CL);
18707:   SIRegister_TGPStoGPX(CL);
18708:   SIRegister_TGPSSpeed(CL);
18709:   SIRegister_TGPSSatellitesPosition(CL);
18710:   SIRegister_TGPSSatellitesReception(CL);
18711:   SIRegister_TGPSSCompass(CL);
18712: //Proc Register( );
18713: Func IndexMsgGP( StrMsgGP :Str) : TMsgGP;
18714: Func StrCoordToAngle( Point : Char; Angle :Str) : Double;
18715: Func StrTimeToTime( const Time :Str) : TDateTime;
18716: Func StrToInt( const Str :Str) : Int;
18717: Func StrToReal( const Str :Str) : Extended;
18718: Func GPSRotatePoint( Angle : Double; Ct, Pt : TPoint) : TPoint;
18719: Proc LoadResource( ResourceName :Str; ImageList : TImageList);
18720: end;
18721:
18722: Proc SIRegister_NMEA(CL: TPSPascalCompiler);
18723: begin
18724:   NMEADatArray, 'array of string;
18725:   Proc TrimNMEA( var S :Str);
18726:   Proc ExpandNMEA( var S :Str);
18727:   Func ParseNMEA( S :Str) : NMEADatArray;
18728:   Func ChkValidNMEA( S :Str) :Bool;
18729:   Func IdNMEA( S :Str) :Str;
18730:   Func ChkSumNMEA( const S :Str) :Str;
18731:   Func PosInDeg( const PosStr :Str) : Double;
18732:   Func DateTimeNMEA( const StrD, StrT :Str) : TDateTime;
18733:   Func SysClockSet( const StrD, StrT :Str) :Bool;
18734:   Func Ticks2Secs(Ticks : LongInt) : LongInt;;
18735:   Func Secs2Ticks(Secs : LongInt) : LongInt;;
18736:   Func MSecs2Ticks(MSecs : LongInt) : LongInt;;
18737: end;
18738:
18739: Proc SIRegister_SortUtils(CL: TPSPascalCompiler);
18740: begin
18741:   TypeS('SortType1', 'Byte;
18742:   TypeS('SortType2', 'Double;
18743:   TypeS('SortType3', 'DWord;
18744:   //TypeS('PDWordArray', '^DWordArray // will not work;
18745:   TypeS('TDataRecord4', 'record Value : Int; Data : Int; end;
18746:   Function('Proc QuickSort( var List : array of SortType1; Min, Max : Int);
18747:   Proc QuickSortDWord( var List : array of SortType3; Min, Max : Int);
18748:   Proc QuickSortDataRecord4( var List : array of TDataRecord4; Count : Int);
18749:   Proc HeapSort( var List : array of SortType1; Count : DWord; FirstNeeded : DWord);
18750:   Func QuickSelect(var List : array of SortType1; Min, Max, Wanted : Int) : SortType1;
18751:   Func QuickSelectDouble( var List : array of SortType2; Min, Max, Wanted : Int) : SortType2;
18752:   Func QuickSelectDWord( var List : array of SortType3; Min,Max,Wanted: Int):SortType3;
18753: end;
18754:
18755: Proc SIRegister_BitmapConversion(CL: TPSPascalCompiler);
18756: begin
18757:   // TMatrix3x3 = array[1..3,1..3] of Double;
18758:   // TMatrix4x4 = array[1..4,1..4] of Double;
18759:   TypeS('TMatrix3x3', 'array[1..3] of Double;
18760:   TypeS('TMatrix3x3', 'array[1..3] of TMatrix3x3;
18761:   TypeS('TMatrix4x4', 'array[1..4] of Double;
18762:   TypeS('TMatrix4x4', 'array[1..4] of TMatrix4x4;
18763:   Proc ColorTransform( A, B, C : Byte; out X, Y, Z : Byte; const T : TMatrix4x4);
18764:   Proc ColorTransform1( A, B, C : Byte; out X, Y, Z : Float; const T : TMatrix4x4);
18765:   Proc ColorTransform2(const A,B,C:Float; out X, Y, Z : Byte; const T: TMatrix4x4);
18766:   Proc ColorTransformHSI2RGB( H, S, I : Byte; out R, G, B : Byte);
18767:   Proc ColorTransformRGB2HSI( R, G, B : Byte; out H, S, I : Byte);
18768:   Proc ColorTransformRGB2Lab( R, G, B : Byte; out L, a, b : Byte);
18769:   Proc ColorTransformLab2RGB( L, a, b : Byte; out R, G, B : Byte);
18770:   Proc ColorTransformRGB2LOCO( R, G, B : Byte; out S0, S1, S2 : Byte);
18771:   Proc ColorTransformLOCO2RGB( S0, S1, S2 : Byte; out R, G, B : Byte);
18772:   Proc ConvertColorSpace(Image:TLinearBitmap;const T:TMatrix4x4;NewImage:TLinearBitmap);
18773: //Proc ConvertColorSpace1(Image:TLinearBitmap;ColorTransform:TColorTransformProc;NewImage:TLinearBitmap);
18774:   Proc ConvertToGrayscale( const Image, GrayImage : TLinearBitmap);
18775:   Proc ConvertToGrayscale1( const Image : TLinearBitmap);
18776: end;
18777:
18778: Proc SIRegister_ZDBCUtils(CL: TPSPascalCompiler);
18779: begin
18780:   { TZSQLType = (zsqlstUnknown, zsqlstBoolean, zsqlstByte, zsqlstShort, zsqlstInt, zsqlstLong,
18781:   zsqlstFloat,zsqlstDouble,zsqlstBigDecimal, zsqlstString, zsqlstUnicodeString, zsqlstBytes,
18782:   zsqlstDate, zsqlstTime, zsqlstTimestamp, zsqlstDataSet, zsqlstGUID,
18783:   stAsciiStream, stUnicodeStream, stBinaryStream);}
18784:   Func ResolveConnectionProtocol(Url:str; SupportedProtocols:TStringDynArray):Str;
18785: //Proc ResolveDatabaseUrl( const Url:Str; Info:TStrings; var HostName:Str; var Port:Int;var
Database:str; var UserName:Str; var Password :Str; ResultInfo : TStrings);
18786:   Func CheckConversion( InitialType : TZSQLType; ResultType : TZSQLType) :Bool;

```

```

18787: Func DefineColumnName( ColumnType : TZSQLType) :Str;
18788: Proc RaiseSQLException( E : Exception);
18789: //Proc CopyColumnsInfo( FromList : TObjectList; ToList : TObjectList);
18790: Func ToLikeString( const Value :Str) :Str;
18791: Func GetSQLHexWideString( Value : PChar; Len : Int; ODBC :Bool) : WideString;
18792: Func GetSQLHexAnsiStr( Value : PChar; Len : Int; ODBC :Bool) : RawByteString;
18793: Func GetSQLHexString( Value : PChar; Len : Int; ODBC :Bool) :Str;
18794: Func WideStringStream( const AString : WideString) : TStream;
18795: Func ConvertAdoToTypeName( FieldType : SmallInt) :Str;
18796: Func GetTableName( const AField: TField) :Str;;
18797: Func GetFieldName( const AField: TField) :Str;;
18798: end;
18799:
18800: Proc SIRegister_JclTD32( CL: TPSPascalCompiler);
18801: TypeS('TSymbolInfo', 'record Size : Word; SymbolType : Word; end;
18802: //TypeS('PSymbolInfos', '^TSymbolInfos // will not work;
18803: SIRegister_TJclModuleInfo( CL);
18804: SIRegister_TJclLineInfo( CL);
18805: SIRegister_TJclSourceModuleInfo( CL);
18806: SIRegister_TJclSymbolInfo( CL);
18807: SIRegister_TJclProcSymbolInfo( CL);
18808: ClassN( CL.FindClass('TOBJECT'), 'TJclLocalProcSymbolInfo;
18809: ClassN( CL.FindClass('TOBJECT'), 'TJclGlobalProcSymbolInfo;
18810: SIRegister_TJclTD32InfoParser( CL);
18811: SIRegister_TJclTD32InfoScanner( CL);
18812: SIRegister_TJclPeBorTD32Image( CL);
18813: end;
18814:
18815: Proc SIRegister_JvIni( CL: TPSPascalCompiler);
18816: begin
18817:   TypeS('TReadObjectEvent', 'Func ( Sender : TObject; const Section, '
18818:     +'Item, Value :Str) : TObject;
18819:   TWriteObjectEvent, Procedure( Sender: TObject; const Section, Item: Str; Obj: TObject);
18820:   Func StringToFontStyles( const Styles :Str) : TFontStyles;
18821:   Func FontStylesToString( Styles : TFontStyles) :Str;
18822:   Func FontToString( Font : TFont) :Str;
18823:   Proc StringToFont( const Str :Str; Font : TFont);
18824:   Func RectToStr( Rect : TRect) :Str;
18825:   Func StrToRect( const Str :Str; const Def : TRect) : TRect;
18826:   Func JPointToStr( P : TPoint) :Str;
18827:   Func JStrToPoint( const Str :Str; const Def : TPoint) : TPoint;
18828:   Func DefProfileName :Str;
18829:   Func DefLocalProfileName :Str;
18830:   ConstantN('IdnListItem', 'String').SetString( 'Item;
18831: end;
18832:
18833: Proc SIRegister_JvHtControls( CL: TPSPascalCompiler);
18834: begin
18835:   Proc ItemHtDrawEx( Canvas : TCanvas; Rect : TRect; const State: TOwnerDrawState; const Text :Str; const
18836:     HideSelColor: Boolean; var PlainItem: Str; var Width: Int; CalcWidth: Boolean);
18837:   Func ItemHtDraw( Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: str; const
18838:     HideSelColor: Booln) : str;
18839:   Func ItemHtWidth( Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: str; const
18840:     HideSelColor: Bool) : Int;
18841:   Func ItemHtPlain( const Text :Str) : Str;
18842:   Proc ExecuteHyperlink( Sender: TObject; HyperLinkClick: TJvHyperLinkClickEvent; const LinkName: str);
18843:   Func IsHyperLink( Canvas: TCanvas; Rect: TRect; const Text: str; MouseX, mouseY: Int; var HyperLink: str) : Bool;
18844: end;
18845:
18846: Proc SIRegister_NeuralNetwork( CL: TPSPascalCompiler);
18847: begin
18848:   ClassN( CL.FindClass('TOBJECT'), 'TNeuron;
18849:   TypeS('TSynapse', 'record W : Real; Connection : TNeuron; end;
18850:   TypeS('TAcson', 'record Alfa : Real; Beta : Real; Gama : Real; end;
18851:   SIRegister_TNeuron( CL);
18852:   SIRegister_TNeuronLayer( CL);
18853:   SIRegister_TNeuralNet( CL);
18854: end;
18855:
18856: Proc SIRegister_StExpr( CL: TPSPascalCompiler);
18857: begin
18858:   //TypeS('PStFloat', '^TStFloat // will not work;
18859:   TStMethod0Param', 'Func ( Valuel : TStFloat) : TStFloat;
18860:   TStMethod1Param', 'Func ( Valuel : TStFloat) : TStFloat;
18861:   TStMethod2Param', 'Func ( Valuel, Value2 : TStFloat) : TStFloat;
18862:   TStMethod3Param', 'Func ( Valuel, Value2, Value3 : TStFloat) : TStFloat;
18863:   TStGetIdentValueEvent', 'Proc ( Sender : TObject; const Ide'
18864:     + 'ntifier : AnsiStr; var Value : TStFloat);
18865:   TStToken', '( ssStart, ssInIdent, ssInNum, ssInSign, ssInExp, ss'
18866:     + 'Eol, ssNum, ssIdent, ssLPar, ssRPar, ssComma, ssPlus, ssMinus, ssTimes, ssDiv, ssEqual, ssPower);
18867:   SIRegister_TStExpression( CL);
18868:   TStExprErrorEvent', 'Proc ( Sender : TObject; ErrorNumber : '
18869:     + 'LongInt; const ErrorStr : AnsiStr);
18870:   SIRegister_TStExpressionEdit( CL);
18871:   Func AnalyzeExpr( const Expr : AnsiStr) : Double;
18872:   Proc TpVal( const S : AnsiStr; var V : Extended; var Code : Int);
18873: end;
18874:
18875: Proc SIRegister_GR32_Containers( CL: TPSPascalCompiler);

```

```

18873: begin
18874:   ConstantN('BUCKET_MASK','LongWord').SetUInt( $FF);
18875:   ConstantN('BUCKET_COUNT','LongInt').SetInt( BUCKET_MASK + 1);
18876:   Proc SmartAssign( Src, Dst : TPersistent; TypeKinds : TTypeKinds);
18877:   Proc Advance( var Node : TLinkedNode; Steps : Int);
18878: end;
18879:
18880: Proc SIRegister_StSaturn(CL: TPSPascalCompiler);
18881: begin
18882:   TStJupSatPos, 'record X: double; Y: Double; end;
18883:   TStJupSats record Io:TStJupSatPos;Europa:TStJupSatPos;Ganymede:TStJupSatPos;Callisto:TStJupSatPos;end;
18884:   Func ComputeSaturn( JD : Double) : TStEclipticalCord;
18885:   Func ComputePluto( JD : Double) : TStEclipticalCord;
18886:   Func ComputeVenus( JD : Double) : TStEclipticalCord;
18887:   Func ComputeMars( JD : Double) : TStEclipticalCord;
18888:   Func ComputeMercury( JD : Double) : TStEclipticalCord;
18889:   Func ComputeJupiter( JD : Double) : TStEclipticalCord;
18890:   Func ComputeUranus( JD : Double) : TStEclipticalCord;
18891:   Func ComputeNeptune( JD : Double) : TStEclipticalCord;
18892:   Func GetJupSats(JD : TDateTime; HighPrecision, Shadows :Bool) : TStJupSats;;
18893: end;
18894:
18895: Proc SIRegister_JclParseUses(CL: TPSPascalCompiler);
18896: begin
18897:   ClassN(CL.FindClass('OBJECT'),'EUsesListError;
18898:   Func CreateGoal( Text : PChar) : TCustomGoal;
18899: end;
18900:
18901: Proc SIRegister_JvFinalize(CL: TPSPascalCompiler);
18902: begin
18903:   //type
18904:   //TFinalizeProc = procedure;
18905:   TypeS('TFinalizeProc', 'procedure;
18906:   Proc AddFinalizeProc( const UnitName :Str; FinalizeProc : TFinalizeProc);
18907:   Func AddFinalizeObject( const UnitName :Str; Instance : TObject) : TObject;
18908:   Func AddFinalizeObjectNil(const UnitName:str; var Reference: TObject):TObject;
18909:   Func AddFinalizeFreeAndNil(const UnitName:str; var Reference: TObject) : TObject;
18910:   Func AddFinalizeMemory( const UnitName :Str; Ptr : __Pointer) : __Pointer;
18911:   Func AddFinalizeMemoryNil(const UnitName:Str;var Ptr: __Pointer) : __Pointer;
18912:   Proc FinalizeUnit( const UnitName :Str);
18913: end;
18914:
18915: Proc SIRegister_BigIni(CL: TPSPascalCompiler);
18916: begin
18917:   ConstantN('IniTextBufferSize','LongWord').SetUInt( $7000);
18918:   cIniCount,'String').SetString( 'Count;
18919:   TEraseSectionCallback', 'Func ( const sectionName :Str; '
18920:   const s11, s12 : TStringList) :Bool;
18921:   SIRegister_TCommaSeparatedInfo(CL);
18922:   SIRegister_TSectionList(CL);
18923:   SIRegister_TBigIniFile(CL);
18924:   SIRegister_TBiggerIniFile(CL);
18925:   SIRegister_TAppIniFile(CL);
18926:   SIRegister_TLibIniFile(CL);
18927:   Func ModuleName( getLibraryName :Bool) :Str;
18928: end;
18929:
18930: Proc SIRegister_ShellCtrls(CL: TPSPascalCompiler);
18931: begin
18932:   TypeS(TRoot', 'string;
18933:   TypeS(TRootFolder', '( rfDesktop, rfMyComputer, rfNetwork, rfRecycleBi'
18934:   +n, rfAppData, rfCommonDesktopDirectory, rfCommonPrograms, rfCommonStartMen'
18935:   +u, rfCommonStartup, rfControlPanel, rfDesktopDirectory, rfFavorites, rfFon'
18936:   +ts, rfInternet, rfPersonal, rfPrinters, rfPrintHood, rfPrograms, rfRecent,'
18937:   + rfSendTo, rfStartMenu, rfStartup, rfTemplates );
18938:   TShellFolderCapability', '( fcCanCopy, fcCanDelete, fcCanLink, f'
18939:   +'cCanMove, fcCanRename, fcDropTarget, fcHasPropSheet );
18940:   TShellFolderCapabilities', 'set of TShellFolderCapability;
18941:   TShellFolderProperty', '( fpCut, fpIsLink, fpReadOnly, fpShared,'
18942:   + fpFileSystem, fpFileSystemAncestor, fpRemovable, fpValidate );
18943:   TShellFolderProperties', 'set of TShellFolderProperty;
18944:   TShellObjectType', '( otFolders, otNonFolders, otHidden );
18945:   TShellObjectTypes', 'set of TShellObjectType;
18946:   ClassN(CL.FindClass('OBJECT'),'EInvalidPath;
18947:   SIRegister_IShellCommandVerb(CL);
18948:   SIRegister_TShellFolder(CL);
18949:   TNotifyFilter', '( nfFileNameChange, nfDirNameChange, nfAttribut'
18950:   +eChange, nfSizeChange, nfWriteChange, nfSecurityChange );
18951:   TNotifyFilters', 'set of TNotifyFilter;
18952:   SIRegister_TShellChangeThread(CL);
18953:   SIRegister_TCustomShellChangeNotifier(CL);
18954:   SIRegister_TShellChangeNotifier(CL);
18955:   ClassN(CL.FindClass('OBJECT'),'TCustomShellComboBox;
18956:   ClassN(CL.FindClass('OBJECT'),'TCustomShellListView;
18957:   TAddFolderEvent', 'Proc ( Sender : TObject; AFolder: TShel'
18958:   +lFolder; var CanAdd :Bool);
18959:   TGetImageIndexEvent', 'Proc ( Sender : TObject; Index : Int'
18960:   +'eger; var ImageIndex : Int);
18961:   SIRegister_TCustomShellTreeView(CL);

```



```

18962:   SIRegister_TShellTreeView(CL);
18963:   SIRegister_TCustomShellComboBox(CL);
18964:   SIRegister_TShellComboBox(CL);
18965:   SIRegister_TCustomShellListView(CL);
18966:   SIRegister_TShellListView(CL);
18967: Proc InvokeContextMenu(Owner : TWinControl; AFolder : TShellFolder; X, Y : Int);
18968: end;
18969:
18970: Proc SIRegister_fmth(CL: TPSPascalCompiler);
18971: begin
18972:   TypeS('Float', 'Double; !
18973:   'FN_OK', 'LongInt').SetInt( 0);
18974:   ConstantN('FN_DOMAIN', 'LongInt').SetInt( - 1);
18975:   'FN_SING', 'LongInt').SetInt( - 2);
18976:   'FN_OVERFLOW', 'LongInt').SetInt( - 3);
18977:   'FN_UNDERFLOW', 'LongInt').SetInt( - 4);
18978:   'FN_TLOSS', 'LongInt').SetInt( - 5);
18979:   'FN_PLOSS', 'LongInt').SetInt( - 6);
18980:   //ConstantN('NFACT', 'LongInt').SetInt( 33);
18981: Func MathError : Int;
18982: Func FMin( X, Y : Float) : Float;
18983: Func FMax( X, Y : Float) : Float;
18984: Func IMin( X, Y : Int) : Int;
18985: Func IMax( X, Y : Int) : Int;
18986: Func FSgn( X : Float) : Int;
18987: Func Sgn0( X : Float) : Int;
18988: Func DSgn( A, B : Float) : Float;
18989: Proc FSwap( var X, Y : Float);
18990: Proc ISwap( var X, Y : Int);
18991: Func fExpo( X : Float) : Float;
18992: Func fExp2( X : Float) : Float;
18993: Func fExp10( X : Float) : Float;
18994: Func fLog( X : Float) : Float;
18995: Func fLog2( X : Float) : Float;
18996: Func fLog10( X : Float) : Float;
18997: Func fLogA( X, A : Float) : Float;
18998: Func fIntPower( X : Float; N : Int) : Float;
18999: Func fPower( X, Y : Float) : Float;
19000: Func Pythag( X, Y : Float) : Float;
19001: Func FixAngle( Theta : Float) : Float;
19002: Func fTan( X : Float) : Float;
19003: Func fArcSin( X : Float) : Float;
19004: Func fArcCos( X : Float) : Float;
19005: Func fArcTan2( Y, X : Float) : Float;
19006: Proc fSinCos( X : Float; var SinX, CosX : Float);
19007: Func fSinh( X : Float) : Float;
19008: Func fCosh( X : Float) : Float;
19009: Func fTanh( X : Float) : Float;
19010: Func fArcSinh( X : Float) : Float;
19011: Func fArcCosh( X : Float) : Float;
19012: Func fArcTanh( X : Float) : Float;
19013: Proc fSinhCosh( X : Float; var SinhX, CoshX : Float);
19014: Func fFact( N : Int) : Float;
19015: Func fBinomial( N, K : Int) : Float;
19016: Func fGamma( X : Float) : Float;
19017: Func fSgnGamma( X : Float) : Int;
19018: Func LnGamma( X : Float) : Float;
19019: Func fGamma( A, X : Float) : Float;
19020: Func fJGamma( A, X : Float) : Float;
19021: Func fBeta( X, Y : Float) : Float;
19022: Func fIBeta( A, B, X : Float) : Float;
19023: Func fErf( X : Float) : Float;
19024: Func fErfc( X : Float) : Float;
19025: Func fPBinom(N: Int; P: Float; K : Int) : Float;
19026: Func FBinom(N: Int; P: Float; K : Int) : Float;
19027: Func PPoisson( Mu : Float; K : Int) : Float;
19028: Func FPoisson( Mu : Float; K : Int) : Float;
19029: Func fDNorm( X : Float) : Float;
19030: Func FNorm( X : Float) : Float;
19031: Func PNorm( X : Float) : Float;
19032: Func InvNorm( P : Float) : Float;
19033: Func fDStudent( Nu : Int; X : Float) : Float;
19034: Func FStudent( Nu : Int; X : Float) : Float;
19035: Func PStudent( Nu : Int; X : Float) : Float;
19036: Func fDKhi2( Nu : Int; X : Float) : Float;
19037: Func FKhi2( Nu : Int; X : Float) : Float;
19038: Func PKhi2( Nu : Int; X : Float) : Float;
19039: Func fDSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
19040: Func FSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
19041: Func PSnedecor( Nu1, Nu2 : Int; X : Float) : Float;
19042: Func fDExpo( A, X : Float) : Float;
19043: Func FExpo( A, X : Float) : Float;
19044: Func fDBeta( A, B, X : Float) : Float;
19045: Func FBeta( A, B, X : Float) : Float;
19046: Func fDGamma( A, B, X : Float) : Float;
19047: Func FGamma( A, B, X : Float) : Float;
19048: Proc RMarIn( Seed1, Seed2 : Int);
19049: Func IRanMar : LongInt;
19050: Func RanMar : Float;

```

```

19051: Func RanGaussStd : Float;
19052: Func RanGauss( Mu, Sigma : Float ) : Float;
19053: end;
19054:
19055: Proc SIRegister_fcomp(CL: TPSPascalCompiler);
19056: begin
19057:   TypeS('ComplexForm', '( Rec, Pol );
19058:   TypeS(TComplex', 'record Form:ComplexForm; X:Float;Y: Float; R:Float; Theta:Float; end;
19059:   Proc CSet( var Z : TComplex; A, B : Float; F : ComplexForm);
19060:   Proc CConvert( var Z : TComplex; F : ComplexForm);
19061:   Proc CSwap( var X, Y : TComplex);
19062:   Func CReal( Z : TComplex) : Float;
19063:   Func CImag( Z : TComplex) : Float;
19064:   Func CAbs( Z : TComplex) : Float;
19065:   Func CArg( Z : TComplex) : Float;
19066:   Func CSgn( Z : TComplex) : Int;
19067:   Proc CNeg( A : TComplex; var Z : TComplex);
19068:   Proc CConj( A : TComplex; var Z : TComplex);
19069:   Proc CAdd( A, B : TComplex; var Z : TComplex);
19070:   Proc CSub( A, B : TComplex; var Z : TComplex);
19071:   Proc CDiv( A, B : TComplex; var Z : TComplex);
19072:   Proc CMult( A, B : TComplex; var Z : TComplex);
19073:   Proc CLn( A : TComplex; var Z : TComplex);
19074:   Proc CExp( A : TComplex; var Z : TComplex);
19075:   Proc CPower( A, B : TComplex; var Z : TComplex);
19076:   Proc CIntPower( A : TComplex; N : Int; var Z : TComplex);
19077:   Proc CRealPower( A : TComplex; X : Float; var Z : TComplex);
19078:   Proc CSqrt( A : TComplex; var Z : TComplex);
19079:   Proc CRoot( A : TComplex; K, N : Int; var Z : TComplex);
19080:   Proc CCSin( A : TComplex; var Z : TComplex);
19081:   Proc CCos( A : TComplex; var Z : TComplex);
19082:   Proc CTan( A : TComplex; var Z : TComplex);
19083:   Proc CArcSin( A : TComplex; var Z : TComplex);
19084:   Proc CArcCos( A : TComplex; var Z : TComplex);
19085:   Proc CArcTan( A : TComplex; var Z : TComplex);
19086:   Proc CSinh( A : TComplex; var Z : TComplex);
19087:   Proc CCosh( A : TComplex; var Z : TComplex);
19088:   Proc CTanh( A : TComplex; var Z : TComplex);
19089:   Proc CArcSinh(A : TComplex; var Z : TComplex);
19090:   Proc CArcCosh(A : TComplex; var Z : TComplex);
19091:   Proc CArcTanh(A : TComplex; var Z : TComplex);
19092:   Proc CLnGamma(A : TComplex; var Z : TComplex);
19093: end;
19094:
19095:   XSDd:= DateTimeToXSDateTime(now,true)
19096:   xsdd.nativetoxs;
19097:   writeln(datetimetostr(xsdd.asUTCdatetime));
19098:   xsdd.free;
19099:
19100: Proc SIRegister_XSBuiltIns(CL: TPSPascalCompiler);
19101: begin
19102:   ConstantN('SHexMarker','String').SetString('$';
19103:   Func DateTimeToXMLTime( Value : TDateTime; ApplyLocalBias :Bool) : WideString;
19104:   Func XMLTimeToDateTime(const XMLDateTime : WideString; AsUTCtime:Boolean): TDateTime;
19105:   Func DateTimeToXSDateTime(const Value:TDateTime;ApplyLocalBias:Bool):TXSDateTime;
19106:   Func GetDataFromFile( AFileName :Str) :Str;
19107:   Func SoapFloatToStr( Value : double) :Str;
19108:   Func SoapStrToFloat( Value :Str) : double;
19109:   Proc InitXSTypes;
19110: end;
19111:
19112: Proc SIRegister_CompFileIo(CL: TPSPascalCompiler);
19113: begin
19114:   SIRegister_TComponentStream(CL);
19115:   TypeS('TComponentStream', 'TMemoryStream;
19116:   TComponentFileFormat', '( cffText, cffBinary );
19117:   TComponentArray', 'array of TComponent;
19118:   TResourceNaming,rnClassNameTag,rnClassTag,rnNameTag,rnClass,rnName, rnTag );
19119:   Func VOID_COMP : __Pointer;
19120:   Func VOID_FORM : __Pointer;
19121:   Func CreateIoForm( AForm : TCustomForm; ClassType : TFormClass) : TCustomForm;
19122:   Func GetComponentTree(Component:TComponent;pComponents:TComponentArray;bAddSelf:Bool):LongInt;
19123:   Func pPosEx( SearchStr: PChar; Str : PChar; var Pos : LongInt) : PChar;
19124:   Func pGetTextBetween(pBuff:PChar;bSearchCode:str;eSearchCode:str;Container:TStrings): LongInt;
19125:   Func pComponentToString( Component : TComponent) :Str;
19126:   //Func StringToComponent(Value:str; ComponentClass:TComponentClass): TComponent;
19127:   Func StringToObjectBinaryStream(Value:str;BinStream:TMemoryStream;ResName:str):Bool;
19128:   Func ObjectBinaryStreamToString(BinStream:TMemoryStream;var sResult:str;bResource:Bool):Bool
19129:   Func ObjectTextToBinaryStream(StrStream,BinStream:TComponentStream;ObjCount:LongInt): LongInt;
19130:   Func
   ObjectBinaryStreamToObjectTextStream(BinStream:TMemoryStream;StrStream:TMemoryStream;bResource:Bool):Bool;
19131:   Func GetResHeaderInfo( Stream : TMemoryStream; sl : TStrings) :Bool;
19132:   Func pGetResourceName(Component:TComponent;NamingMethod: TResourceNaming) :Str;
19133:   Func GetFormResourceStream(Form:TCustomForm;ResName:str;var ResourceStream:TMemoryStream):Bool;
19134:   Func WriteComponentsToFile(FormsAndComponents:array of
   TComponent;FileName:str;Format:TComponentFileFormat; StoreComponentNames:Boolean):Boolean;
19135:   Func ReadComponentsFromFile(FormsAndComponents:array of TComponent;FileName:str):Bool;
19136:   //Func ReadComponentsFromFile1(FileName:str;pComponents: array of TPCOMPONENT;ComponentClasses:array of
   TComponentClass):Bool;

```

```

19137: Func ReadComponentTreeFromFile(FormOrComponent:TComponent;FileName:Str):Boolean;
19138: Func
WriteComponentToFile(Component:TComponent;FileName:Str;Format:TComponentFileFormat;StoreComponentName:Bool):Bool;
19139: Func
WriteComponentTreeToFile(FormOrComponent:TComponent;FileName:Str;Format:TComponentFileFormat;StoreComponentName:Bool):
19140: Func ReadComponentFromFile4(FormOrComponent:TComponent; FileName :Str):Bool;;
19141: Func ReadFormFromFile(pInstance:TComponent;FormClass:TFormClass;FileName:Str): Bool;
19142: Func ReadComponentResourceFile5( Instance : TObject; FileName :Str) :Bool;;
19143: Func WriteComponentResourceFile(nstance:TObj;FileName:Str;StoreFormAsVisible:Bool):Bool;
19144: Func ReadComponentsResourceFile(Components:array of TComponent;FileName:Str;NamingMethod:TResourceNaming;
bLoadTotalForm:Boolean):Boolean;
19145: Func WriteComponentsResourceFile( Components : array of TComponent; FileName:Str; NamingMethod :
TResourceNaming) :Bool;
19146: Func ReadComponentResourceHeader(sHeaderInfo:TStrings; pSize : Int; FileName:Str; NamingMethod :
TResourceNaming) :Bool;
19147: Func ConvertComponentResourceToTextFile(SourceFileName:Str;TargetFileName:Str;bStoreResNames:Bool):Bool;
19148: Func CheckComponentInResourceFile(Component:TComponent;FileName:Str;NamingMethod:TResourceNaming):Boolean;
19149: Func DeleteComponentFromResourceFile(ResourceName:Str; FileName:Str):Bool;;
19150: Func DeleteComponentFromResourceFile8(Component:TComponent;FileName
string;NamingMethod:TResourceNaming):Bool;
19151: //Func CreateFormFromResFile(AOwner:TComponent;NewClassType:TFormClass;FileName:Str): Pointer;
19152: //Func CreateComponentFromResFile( AOwner : TComponent; NewClassType : TComponentClass; FileName :Str) :
Pointer;
19153: end;
19154:
19155: Proc SIRegister_SMScript(CL: TPSPascalCompiler);
19156: begin
19157: CL.FindClass('TObject'),'TSMSEScriptExecutor;
19158: SIRegister_TSMSEError(CL);
19159: ClassN(CL.FindClass('TObject'),'TSMSEModule;
19160: TypeS('TSMSEProcedureType','( ptProcedure, ptFunc );
19161: SIRegister_TSMSEProcedure(CL);
19162: SIRegister_TSMSEProcedures(CL);
19163: SIRegister_TSMSEModule(CL);
19164: SIRegister_TSMSEModules(CL);
19165: TypeS('TSMScriptLanguage', ( slCustom, slVBScript, slJavaScript );
19166: SIRegister_TSMSEScriptExecutor(CL);
19167: //DelphiFunction('Proc Register;
19168: end;
19169:
19170: Proc SIRegister_geometry2(CL: TPSPascalCompiler);
19171: begin
19172: TPointF2', 'record X : Double; Y : Double; end;
19173: TPoint3', 'record X : Int; Y : Int; Z : Int; end;
19174: //TypeS('PPointF3', '^TPointF3 // will not work;
19175: TPointF3', 'record X : double; Y : double; Z : Double; end;
19176: Func AreLinesParallel( p1, p2, p3, p4 : TPoint) :Bool;;
19177: Func AreLinesParallel1( p1, p2, p3, p4 : TPointF2; e : Double) :Bool;;
19178: Func AreLinesParallel2( p1, p2, p3, p4, p5, p6 : TPoint3) :Bool;;
19179: Func AreLinesParallel3( p1, p2, p3, p4, p5, p6 : TPointF3; e : Double) :Bool;;
19180: Func IntersectLines( p1, p2, p3, p4 : TPoint) : TPoint;;
19181: Func IntersectLines1( p1, p2, p3, p4 : TPointF2; e : Double) : TPointF2;;
19182: Func AngleDifference( alpha, beta : Double) : Double;
19183: end;
19184:
19185: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
19186: New Functions /Classes mX4:
19187: Func DownloadJPGToBitmap(const URL :Str; ABitmap: TBitmap):Bool;;
19188: Proc GetImageLinks(AURL:Str; AList: TStrings);
19189: Proc SaveByteCode;;
19190: Proc ResetKeyPressed;;
19191: Proc SetKeyPressed;;
19192:
19193: RIRegister_HTTPProd_Routines(Exec);
19194: Func ContentFromScriptStream(AStream:TStream; AWebModuleContext : TWebModuleContext; AStripParamQuotes :
Bool; AHandleTag : THandleTagProc; AHandleScriptTag : THandledTagProc; const AScriptEngine :Str;
ALocateFileService : ILocateFileService) :Str;
19195: Func ContentFromScriptFile(const AFileName:TFileName; AWebModuleContext:TWebModuleContext;
AStripParamQuotes :Bool; AHandleTag:THandleTagProc;AHandleScriptTag:THandledTagProc; '+
const AScriptEngine:Str; ALocateFileService:ILocateFileService) :Str;
19196: Func FindComponentWebModuleContext( AComponent : TComponent) : TWebModuleContext;
19198: Func GetTagID( const TagString :Str) : TTag;
19199: Func ContentFromStream(AStream:TStream;
AStripParamQuotes:Boolean;AHandleTag:THandleTagProc;AHandledTag:THandledTagProc):Str;
19200: Func ContentFromString(const AValue:Str;
AStripParamQuotes:Bool;AHandleTag:THandleTagProc;AHandledTag:THandledTagProc):Str;
19201: RIRegister_synacrypt_Routines(Exec);
19202: Func TestDes:Bool;
19203: {Call internal test of all 3DES encryptions. Returns @true if all is OK.}
19204: Func Test3Des:Bool;
19205: {Call internal test of all AES encryptions. Returns @true if all is OK.}
19206: Func TestAes:Bool;
19207: Proc LogMessage( const Fmt :Str; const Params : array of const);
19208: Func UnixPathToDosPath2( const Path :Str) :Str;
19209: Func DosPathToUnixPath2( const Path :Str) :Str;
19210: Proc InitISAPIApplicationList;
19211: Proc DoneISAPIApplicationList;
19212:
19213: RIRegister_xmlDOM_Routines(Exec);

```

```

19214: Func IsPrefixed( const AName : DOMString ) : Bool;
19215: Func IsPrefixedW( const AName : DOMStringW ) : Bool;
19216: Func ExtractLocalName( const AName : DOMString ) : DOMString;
19217: Func ExtractLocalNameW( const AName : DOMStringW ) : DOMStringW;
19218: Func ExtractPrefixW( const AName : DOMStringW ) : DOMStringW;
19219: Func MakeNodeNameW( const Prefix, LocalName : DOMStringW ) : DOMStringW;
19220: Func ExtractPrefix( const AName : DOMString ) : DOMString;
19221: Func MakeNodeName( const Prefix, LocalName : DOMString ) : DOMString;
19222: Func SameNamespace( const Node:IDOMNode; const namespaceURI:WideString ) : Boolean;;
19223: Func SameNamespace2( const URI1, URI2 : WideString ) : Bool;;
19224: Func NodeMatches( const Node:IDOMNode; const TagName, NamespaceURI:DOMString ) : Bool;;
19225: Func GetDOMNodeEx( const Node : IDOMNode ) : IDOMNodeEx;
19226: Proc RegisterDOMVendor( const Vendor : TDOMVendor );
19227: Proc UnRegisterDOMVendor( const Vendor : TDOMVendor );
19228: Func GetDOMVendor( VendorDesc : Str ) : TDOMVendor;
19229: Func GetDOM( const VendorDesc : Str ) : IDOMImplementation;
19230: Proc DOMVendorNotSupported( const PropOrMethod, VendorName : Str );
19231: Func IsValidLocale( Locale : LCID; dwFlags : DWORD ) : BOOL;
19232: Func ConvertDefaultLocale( Locale : LCID ) : LCID;
19233: Func GetThreadLocale : LCID;
19234: Func SetThreadLocale( Locale : LCID ) : BOOL;
19235: Func GetSystemDefaultLangID : LANGID;
19236: Func GetUserDefaultLangID : LANGID;
19237: Func GetSystemDefaultLCID : LCID;
19238: Func GetUserDefaultLCID : LCID;
19239: Func AbsInt( const B : Int ) : Int;
19240: Func AbsFloat( const B : double ) : extended;
19241:
19242: Proc ReconcileDeltas( const cdsArray:array of TClientDataset; vDeltaArray:OleVariant );
19243: Proc CDSApplyUpdates( ADatabase:TDatabase; var vDeltaArray:OleVariant; const
vProviderArray:OleVariant; Local:Bool );
19244: Function( @CheckMemory, 'Proc CheckMemory;;
19245: Function( @GetMemoryInfo, 'Func getMemoryInfo;;
19246: Function( @GetMemoryInfo, 'Func getMemInf;;
19247: Proc RaiseLastWin32_2( const Text : Str );
19248:
19249: RIRegister_Gameboard_Routines( Exec );
19250: Func Opponent( Player : TPlayer ) : TPlayer;
19251: Proc InitZobristsNumbers( var ZobristNumbers, Count : Int );
19252: Proc SaveStringToStream( const Str : Str; Stream : TStream );
19253: Func LoadStringFromStream( Stream : TStream ) : Str;
19254: Func WaitForSyncObject( SyncObject:THandle; Timeout:Card; BlockInput:Boolean ):Card;
19255: Func ProcessMessage : Bool;
19256: Proc ProcessMessages( Timeout : DWORD ); //without application!
19257:
19258: RIRegister_PersistSettings_Routines( Exec );
19259: Proc SetStorageHandler( AFunc : TStorageHandlerFunction );
19260: Proc SetStorageHandler1( AMethod : TStorageHandlerMethod );
19261: Func GetStorage : TPersistStorage;
19262: Proc SaveComponents( Root : TComponent; Storage : TPersistStorage );
19263: Proc LoadComponents( Root : TComponent; Storage : TPersistStorage );
19264: Proc AutoSave( Root : TComponent );
19265: Proc AutoLoad( Root : TComponent );
19266:
19267: Proc FloatToDecimalE( var Result:TFloatRec; const Value:extended; ValueType:TFloatValue; Precision,
Decimals: Int );
19268: Func FloatToTextE( BufferArg:PChar; const Value:extended; ValueType:FloatValue; Format:TFloatFormat;
Precision, Digits: Int ) : Int;
19269: Proc FloatToDecimalE( var Result:TFloatRec; const Value:extended; ValueType:TFloatValue; Precision,
Decimals: Int );
19270: Func FloatToTextE( BufferArg:PChar; const
Value:extended; ValueType:TFloatValue; Format:TFloatFormat; Precision, Digits: Int ) : Int;
19271: Func GetSystemDefaultLCID : LCID;
19272: Func GetUserDefaultLCID : LCID;
19273: Func CreateMutex2( lpMutexAttributes:TObject; bInitialOwner:BOOL; lpName:PChar ) : THandle;
19274: Func CreateSemaphore2( lpSemaphoreAttributes:TObject; lInitialCount,
lMaximumCount:Longint; lpName:PChar ):THandle;
19275: Func GetUserNameAPI( lpBuffer : PChar; var nSize : DWORD ) : BOOL;
19276: Createmutex2 fhand:= OpenFileHandle(exepath+'maxbox4.exe')
19277: Func getMatchString( arex, atext:Str ):Str;
19278: Func getLastInput: DWord;
19279: Proc GetKLList( List: TStrings ); //Keyboardlist2
19280: Proc EnableCTRLALTDEL( YesNo : Bool );
19281: Func LocalIP:Str;
19282: Func IPAddrToName( IPAddr:Str ):Str;
19283: Func GetIPFromHost( const HostName:Str ):Str;
19284: function getIPFromHostName( hostName: string ): string;
19285: Func FindComputers( Computers : TStringList ) : DWORD;
19286: Func GetWin32TypeLibList( var Lines: TStringList ) : Bool;
19287: Func RecurseWin32( const R: TRegistry; const ThePath:Str;
const TheKey:Str ):Str;
19288:
19289: Func RunAsAdmin( hWnd: HWND; filename:Str; Parameters:Str ):Bool;
19290: Ex.: RunAsAdmin2( hinstance, 'cmd.exe', +
19291: 'Set OPENSsl_CONF='+OpenSSLPathF+'openssl.cfg' );
19292: writeln( getDosOutput( 'openssl.exe version', OpenSSLPath ) );
19293: Proc SaveGraphicToStream( Graphic: TGraphic; Stream: TStream );
19294: Func LoadGraphicFromStream( Stream: TStream ) : TGraphic;
19295: Proc CopyStreamToFile( S: TStream; F: THandle );
19296: Func BigPow( aone, atwo: Int ):Str;

```

```

19297: Proc GetPelsPerMeter(CONST Bitmap: TBitmap;VAR xPelsPerMeter,yPelsPerMeter: Int);
19298: Proc RainbowColor(CONST fraction: Double; VAR R,G,B: BYTE);
19299: RIRRegister_JclPCRE_Routines(S: TPSExec);
19300: Proc InitializeLocaleSupport;
19301: Proc TerminateLocaleSupport;
19302: Func StrReplaceRegEx(const Subject, Pattern:Ansistr;Args:array of const):Ansistr;
19303:
19304: Proc SIRegister_VariantRtn(CL: TPSPascalCompiler);
19305: begin
19306: TProcReadElementValue='procedure(Value:Variant; IndexValue:Intarray; const HighBoundInd:Int; Var
Continue:bool);
19307: TProcWriteElementValue='procedure(OldValue:Variant;IndexValue:Intarray;Var NewValue:Variant; Var
Continue:bool);
19308: //TProcReadElementValue=procedure(Value:Variant;IndexValue:array of Int; const HighBoundInd:Int;Var
Continue:bool);
19309: //TProcWriteElementValue=procedure(OldValue:Variant;IndexValue:array of Int;Var NewValue:Variant;Var
Continue:bool); //
19310: Func SafeVarArrayCreate(const Bounds:array of Int;VarType, DimCount: Int): Variant;
19311: Func VarArrayGet2(constA:Variant;const Indice:array of Int;const HighBound:Int):Variant);
19312: Proc VarArrayPut2(var A:Variant;const Value:Variant;const Indices:array of Int;const HighBound:Int);
19313: Func CycleReadArray( vArray : Variant; CallBackProc : TProcReadElementValue ) :Bool;
19314: Func CycleWriteArray(var vArray:Variant;CallBackProc: TProcWriteElementValue):Bool;
19315: Func CompareVarArray1( vArray1, vArray2 : Variant ) :Bool;
19316: Func EasyCompareVarArray1(vArray1,vArray2:Variant;HighBound:Int):Bool;
19317: end;
19318:
19319: Proc SIRegister_StdFuncs(CL: TPSPascalCompiler);
19320: begin
19321: ClassN(CL.FindClass('TOBJECT'),'EParserError;
19322: //TypeS('TCharSet', 'set of Char;
19323: Func ConvertFromBase(sNum:Str; iBase: Int; cDigits:Str): Int;
19324: Func ConvertToBase( iNum, iBase : Int; cDigits :Str) :Str;
19325: Func EnsureSentenceTerminates( Sentence :Str; Terminator: Char) :Str;
19326: Func FindTokenStartingAt(st:Str;var i:Int;TokenChars:TCharSet;TokenCharsInToken:Bool):Str;
19327: Func GetDirectoryOfFile( FileName :Str) :Str;
19328: Func GetDirOfFile( FileName :Str) :Str;
19329: Func GetTempFile( FilePrefix :Str) :Str;
19330: Func Icon2Bitmap( Icon : HICON) : HBITMAP;
19331: Func Maxfib( n1, n2 : Int) : Int;
19332: Func MaxD( n1, n2 : Double) : Double;
19333: Func Minfib( n1, n2 : Int) : Int;
19334: Func MinD( n1, n2 : Double) : Double;
19335: Func RandomStringfib( iLength : Int) :Str;
19336: Func RandomIntfib( iLow, iHigh : Int) : Int;
19337: Func Soundexfib( st :Str) :Str;
19338: Func StripStringfib( st :Str; CharsToStrip :Str) :Str;
19339: Func ClosestWeekday( const d : TDateTime) : TDateTime;
19340: Func Yearfib( d : TDateTime) : Int;
19341: Func Monthfib( d : TDateTime) : Int;
19342: Func DayOfYearfib( d : TDateTime) : Int;
19343: Func DayOfMonth( d : TDateTime) : Int;
19344: Func VarCoalesce( V1, V2 : Variant) : Variant;
19345: Func VarEqual( V1, V2 : Variant) :Bool;
19346: Proc WeekOfYearfib( d : TDateTime; var Year, Week : Int);
19347: Func Degree10( Degree : Int) : double;
19348: Func CompToStr( Value : comp) :Str;
19349: Func StrToComp( const Value :Str) : comp;
19350: Func CompDiv( A, B : comp) : comp;
19351: Func CompMod( A, B : comp) : comp;
19352: // TypeS('PComp', '^Comp // will not work;
19353: end;
19354:
19355: Proc SIRegister_RegUtils(CL: TPSPascalCompiler);
19356: begin
19357: Proc DefWriteToRegistry(const OtherKeys,ParamNams:array of string;const Values:array of Variant);
19358: Proc WriteToRegistry(aRootKey:HKEY;const OtherKeys,ParamNames:array of string;const Values:array of
Variant);
19359: Func ReadFromRegistry(aRootKey:HKEY;const OtherKeys,ParamNames: array of string): Variant;
19360: Func DefReadFromRegistry( const OtherKeys, ParamNames : array of string) : Variant;
19361: Func AllSubKey( aRootKey : HKEY; const ForPath : array of string) : Variant;
19362: Func DefAllSubKey( const ForPath : array of string) : Variant;
19363: Func SaveRegKey( const FileName :Str; const ForKey : array of string) :Bool;
19364: Func LoadRegKey( const FileName :Str; const ForKey : array of string) :Bool;
19365: Func AltSaveRegKey(const FileName:str;const ForKey: array of string) :Bool;
19366: Func AltLoadRegKey(const FileName:str;const ForKey: array of string) :Bool;
19367: Func GetKeyForParValue( const aRootKey, ParName, ParValue :Str) :Str;
19368: end;
19369:
19370: Func GetEulerPhi( n : int64) : int64;
19371: Func isprime(f: int64):Bool;;
19372: Func DispositionFrom(const SQLText:str):TPoint;
19373: Proc AllTables(const SQLText:str;FTables:Tstrings);
19374: Func TableByAlias(const SQLText,Alias:str):str;
19375: Func FullFieldName(const SQLText,FieldName:str):str;
19376: Func AddToWhereClause(const SQLText,NewClause:str):str;
19377: Func GetWhereClause(SQLText:str;N:Int;var StartPos,EndPos:Int ):str;
19378: Func WhereCount(SQLText:str):Int;
19379: Func GetOrderInfo(SQLText:str):variant;
19380: Func OrderStringTxt(SQLText:str; var StartPos,EndPos:Int ):str;

```



```

19381: Func   PrepareConstraint(Src:Tstrings):str;
19382: Proc   DeleteEmptyStr(Src:Tstrings);
19383: Func   NormalizeSQLText(const SQL:Str;MacroChar:Char):Str;
19384: Func   CountSelect(const SrcSQL:str):str;
19385: Func   GetModifyTable(const SQLText:str;AlreadyNormal:boolean):str;
19386: Func   GetCharFromVKey(vkey: Word):Str;
19387: Func   Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile:Str):Bool;
19388: Func   IsObjectActive(ClassName:Str):Bool;
19389: Func   GetActiveObject(ClassID:TGUID; anil:TObject; aUnknown:IUnknown):HRESULT;;
19390: Func   RegisterOCX(FileName:Str):Bool;
19391: Func   UnRegisterOCX(FileName:Str):Bool;
19392: Func   RegisterServer2(const aDllFileName:Str; aRegister:Bool):Bool;
19393: Proc   mIRCDE(Service, Topic, Cmd:Str);
19394:         //mIRCDE('mIRC', 'COMMAND', '/say Hallo von SwissDelphiCenter.ch;
19395: Func   OpenIE(aURL:Str):Bool;
19396: Func   XRTLIsInMainThread:Bool;
19397: Func   IsInMainThread:Bool;
19398:
19399: Proc   SIRegister_SpectraLibrary(CL: TPSPascalCompiler);
19400: begin
19401:   TypeS('Nanometers', 'Double;
19402:   ConstantN('WavelengthMinimum','LongInt').SetInt( 380);
19403:   ConstantN('WavelengthMaximum','LongInt').SetInt( 780);
19404:   Proc WavelengthToRGB ( const Wavelength : Nanometers; var R, G, B : BYTE);
19405: end;
19406: Proc   SIRegister_DrawFigures(CL: TPSPascalCompiler);
19407: begin
19408:   Proc DrawCube( const PantoGraph : TPantoGraph; const color : TColor);
19409:   Proc DrawSphere(const PantoGraph:TPantoGraph;const LatitudeColor,LongitudeColor:TColor; const
LatitudeCircles,LongitudeSemicircles,PointsInCircle:WORD);
19410:   Proc DrawSurface ( const PantoGraph : TPantoGraph);
19411:   Proc DrawFootballField(const PantoGraph:TPantoGraph;const ColorField,ColorLetters,ColorGoals:TColor);
19412: end;
19413: Proc   SIRegister_synadbg(CL: TPSPascalCompiler);
19414: begin
19415:   SIRegister_TSynaDebug(CL);
19416:   Proc AppendToLog( const value : Ansistr);
19417: end;
19418:
19419: Proc   SIRegister_XmlRpcCommon(CL: TPSPascalCompiler);
19420: begin
19421:   SIRegister_TRC4(CL);
19422:   TypeS('TRPCDataType', '( rpNone, rpString, rpInt, rpBoolean, rpDoub'
19423:   +le, rpDate, rpBase64, rpStruct, rpArray, rpName, rpError );
19424:   Func   GetTempDirRPC :Str;
19425:   Func   FileIsExpired( const FileName :Str; Elapsed : Int) :Bool;
19426:   Func   EncodeEntities( const Data :Str) :Str;
19427:   Func   DecodeEntities( const Data :Str) :Str;
19428:   Func   ReplaceRPC(const Data:str;const Find:str;const Replace:str):Str;
19429:   Func   InStr( Start : Int; const Data :Str; const Find :Str) : Int;
19430:   Func   Mid( const Data :Str; Start : Int) :Str;
19431:   Func   DateTimeToISO( ConvertDate : TDateTime) :Str;
19432:   Func   IsoToDateTime( const ISOStringDate :Str) : TDateTime;
19433:   Func   ParseStringRPC(const SearchString:str; Delimiter:Char; Substrings: TStrings; const
AllowEmptyStrings:Bool; ClearBeforeParse:Boolean):Int;
19434:   Func   ParseStream(
SearchStream:TStream;Delimiter:Char;ubstrings:TStrings;AllowEmptyStrings:Bool;ClearBeforeParse:Bool): Int;
19435:   Func   FixEmptyString( const Value :Str) :Str;
19436:   Func   URLEncodeRPC( const Value :Str) :Str;
19437:   Func   StreamToStringRPC( Stream : TStream) :Str;
19438:   Proc   StringToStream( const Text :Str; Stream : TStream);
19439:   Func   StreamToVariant( Stream : TStream) : OleVariant;
19440:   Proc   VariantToStream( V : OleVariant; Stream : TStream);
19441:   Func   Hash128AsHex( const Hash128Value : T4x4LongWordRecord) :Str;
19442:   ConstantN(ValidURLChars','String').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$-_.&+~!'"(),;/#?;
19443: end;
19444:
19445: Proc   SIRegister_synafpc(CL: TPSPascalCompiler);
19446: begin
19447:   Func   LoadLibraryfpc( ModuleName : PChar) : TLibHandle;
19448:   Func   FreeLibraryfpc( Module : TLibHandle) : LongBool;
19449:   Func   GetProcAddressfpc( Module : TLibHandle; Proc : PChar) : Pointer;
19450:   Func   GetModuleFileNamefpc(Module:TLibHandle;Buffer:PChar; BufLen:Int):Int;
19451:   TypeS('TLibHandle', 'Int;
19452:   TypeS('TLibHandle2', 'HModule;
19453:   TypeS('LongWordfpc', 'DWord;
19454:   Proc   Sleepfpc( milliseconds :Card);
19455: end;
19456:
19457: Proc   SIRegister_Spring_Utilsmx(CL: TPSPascalCompiler);
19458: begin
19459:   TypeS('TOSPlatformType', '( ptUnknown, ptWin3x, ptWin9x, ptWinNT );
19460:   SIRegister_TOperatingSystem(CL);
19461:   SIRegister_TEnvironmentClass(CL);
19462:   TypeS('Environment', 'TEnvironment;
19463:   Func   ApplicationPath :Str;
19464:   Func   ApplicationVersion : TVersion;
19465:   Func   ApplicationVersionString :Str;
19466:   Func   GetLastErrorErrorMessage :Str;

```

```

19467: function getLastSysErrorMessage: string;
19468: Func CreateCallback( obj : TObject; methodAddress : Pointer) : TCallbackFunc;
19469: Func ConvertFileTimeToDateTime(const fileTime:TFileTime;useLocalTimeZone:Bool):TDateTime;;
19470: Func ConvertDateTimeToFileTime(const datetime:TDateTime;useLocalTimeZone:Bool):TFileTime;
19471: {Proc Synchronize( threadProc : TThreadProcedure);
19472: Proc Queue( threadProc : TThreadProcedure); }
19473: Func TryGetPropertyInfo(instance:TObject;const propertyName:str;out propInfo:PPropInfo):Bool;
19474: Func IsCtrlPressed :Bool;
19475: Func IsShiftPressed :Bool;
19476: Func IsAltPressed :Bool;
19477: Proc CheckFileExists( const fileName :Str);
19478: Proc CheckDirectoryExists( const directory :Str);
19479: ConstantN('COneKB','Int64').SetInt64( 1024);
19480: COneMB,'Int64').SetInt64( 1048576);
19481: COneGB,'Int64').SetInt64( 1073741824);
19482: COneTB,'Int64').SetInt64( 1099511627776);
19483: Func TryConvertStrToDateTime(const s,format:str;out value:TDateTime):Boolean;;
19484: Func ConvertStrToDateTime(const s, format:Str): TDateTime;;
19485: end;
19486:
19487: Proc SIRegister_rxOle2Auto(CL: TPSPascalCompiler);
19488: begin
19489: ConstantN('MaxDispArgs','LongInt').SetInt( 64);
19490: ConstantN('MaxDispArgs','LongInt').SetInt( 32);
19491: ClassN(CL.FindClass('TOBJECT'),'EPropReadOnly);
19492: ClassN(CL.FindClass('TOBJECT'),'EPropWriteOnly);
19493: SIRegister_ToleController(CL);
19494: Proc InitOLE2;
19495: Proc DoneOLE;
19496: Func OleInitialized :Bool;
19497: Func MakeLangID( PrimaryLangID, SubLangID : Word) : Word;
19498: Func MakeLCID( LangID : Word) : TLCID;
19499: Func CreateLCID( PrimaryLangID, SubLangID : Word) : TLCID;
19500: Func ExtractLangID( LCID : TLCID) : Word;
19501: Func ExtractSubLangID( LCID : TLCID) : Word;
19502: Func CLSIDFromProgID(pszProgID: widestring; out clsid: TGUID): Longint); //stdcall;
19503: Func ProgIDFromCLSID(const clsid: TGUID; out pszProgID: widestring): Longint); //stdcall;
19504: Func StringFromCLSID(const clsid: TGUID; out psz: widestring): longint); //stdcall;
19505: //{$EXTERNALSYM CLSIDFromString}
19506: Func CLSIDFromString(psz: widestring; out clsid: TGUID): longint); //stdcall;
19507: Func StringFromGUID2(const guid: TGUID;psz: widestring;cbMax:Integer):Integer);//stdcall;
19508: Func ProgIDExists(const ProgID:WideString):Boolean);
19509: Proc CoUninitialize);
19510: Proc OleUninitialize);
19511: end;
19512:
19513: Proc SIRegister_ologifit(CL: TPSPascalCompiler);
19514: begin
19515: Proc LogiFit(X,Y:TVector;Lb,
Ub:Int;ConsTerm:Boolean;General:Bool;MaxIter:Int;Tol:Float;B:TVector;V:TMatrix);
19516: Proc WLogiFit(X,Y,S:TVector;Lb,Ub:Int;ConsTerm Bool General:Bool;MaxIter
Int;Tol:Float;B:TVector;V:TMatrix);
19517: Func LogiFit_Func( X : Float; B : TVector) : Float;
19518: end;
19519:
19520: Proc FormattedTextOut(TargetCanvas: TCanvas; const Rect : TRect; const Text :Str; Selected :Bool; Columns
: TProposalColumns; Images : TImageList);
19521: Func FormattedTextWidth(TargetCanvas:TCanvas;const
Text:str;Columns:TProposalColumns;Images:TImageList):Int;
19522: Func PrettyTextToFormattedString(const APrettyText:Str; AlternateBoldStyle:Boolean):str;
19523:
19524: Proc SIRegister_ulinfitt(CL: TPSPascalCompiler);
19525: begin
19526: Proc LinFit( X, Y : TVector; Lb, Ub : Int; B : TVector; V : TMatrix);
19527: Proc WLinFit( X, Y, S : TVector; Lb, Ub : Int; B : TVector; V : TMatrix);
19528: Proc SVDLinFit(X,Y:TVector; Lb, Ub:Int; SVDTol : Float; B: TVector; V : TMatrix);
19529: Proc WSVDLinFit(X,Y,S: TVector; Lb,Ub:Int;SVDTol: Float; B: TVector; V : TMatrix);
19530: Proc SVDFit(X:TMatrix;Y:TVector;Lb,Ub Nvar:Int;ConsTerm:Bool; SVDTol: Float;B: TVector; V: TMatrix);
19531: Proc WSVDFit(X: TMatrix;Y,S:TVector;Lb Ub,Nvar:Int;ConsTerm:Boolean;SVDTol:Float; B: TVector;V : TMatrix);
19532: end;
19533:
19534: Proc SIRegister_MaxUtils(CL: TPSPascalCompiler);
19535: begin
19536: TypeS('MaxCharSet', 'set of Char;
19537: Func GetMachineNamemax :Str;
19538: Func GetModuleNameamax( HModule : THandle) :Str;
19539: Func TrimChars( const S :Str; Chars : MaxCharSet) :Str;
19540: Func TickCountToDateTime( Ticks :Card) : TDateTime;
19541: Proc OutputDebugStringmax( const S :Str);
19542: Proc OutputDebugFormat( const FmtStr :Str; Args : array of const);
19543: Func IsAppRunningInDelphi :Bool;
19544: Proc ParseFields(Separators,WhiteSpace:TSysCharSet;Content:PChar;Strigs:TStrings;Decode: Bool;
19545: Func HTTPDecodemax( const AStr :Str) :Str;
19546: Func HTTPEncodemax( const AStr :Str) :Str;
19547: Func FormatDate( const DateString :Str) :Str;
19548: Func FormatListMasterDate(const DateStr,FormatDefStr:str; Len:Int):str;
19549: Func InvertCase( const S :Str) :Str;
19550: Func CommentLinesWithSlashes( const S :Str) :Str;
19551: Func UncommentLinesWithSlashes( const S :Str) :Str;

```

```

19552: Func StripChars( const S :Str; Strip : CharSet ) :Str;
19553: Func TrimChars( const S :Str; Chars : CharSet ) :Str;
19554: Func TrimLeftChars( const S :Str; Chars : CharSet ) :Str;
19555: Func TrimRightChars( const S :Str; Chars : CharSet ) :Str;
19556: Func ContainsChars( const S :Str; Strip : CharSet ) :Bool;
19557: Func DequotedStrmax( const S :Str; AQuoteChar : Char ) :Str;
19558: Proc LeftPadStr( var S :Str; toLength : Int; withChar : Char);
19559: Proc RightPadStr( var S :Str; toLength : Int; withChar : Char);
19560: Func RemoveChars( S :Str; Chars : CharSet ) :Str;
19561: Func FilterChars( S :Str; Chars : CharSet ) :Str;
19562: Func RemoveNonNumericChars( S :Str ) :Str;
19563: Func RemoveNonAlphanumChars( S :Str ) :Str;
19564: Func RemoveNonAlphaChars( S :Str ) :Str;
19565: Func HasAlphaChars( S :Str ) :Bool;
19566: Func ReplaceChars( S :Str; Chars : CharSet; ReplaceWith : Char ) :Str;
19567: Func DomainOfEMail( const EMailAddress :Str ) :Str;
19568: Func IPToHexIP( const IP :Str ) :Str;
19569: Proc CmdLineToStrings( S : Ansistr; const List : TStrings);
19570: BASE2('String').SetString( '01';
19571: ConstantN('BASE10','String').SetString(0123456789;
19572: ConstantN('BASE16','String').SetString(0123456789ABCDEF;
19573: ConstantN('BASE36','String').SetString(0123456789ABCDEFGHIJKLMNOSTUVWXYZ;
19574: ConstantN('BASE62','String').SetString(0123456789ABCDEFGHIJKLMNOSTUVWXYZabcdefghijklmnopqrstuvwxyz;
19575: Func BaseConvert( Number, FromDigits, ToDigits :Str) :Str;
19576: Func ValidXmlName( AName : PChar; ASize : Int) :Bool;;
19577: Func ValidXmlName1( const AName :Str) :Bool;;
19578: Func EncodeXmlAttribute( const AStr : Ansistr) : Ansistr;;
19579: Proc EncodeXmlAttribute3( ABuff:PChar;ABuffSize:Int;var AStr:Ansistr;var ALen,AOffset:Int);
19580: Proc EncodeXmlAttribute4(const ASource:Str;var ADest:str;var ALen,AOffset:Int);
19581: Func EncodeXmlString5( const AStr :Str) :Str;;
19582: Proc EncodeXmlString6(ABuff:PChar;ABuffSize:Int;var AStr:Ansistr;var ALen,AOffset:Int;
19583: Proc EncodeXmlComment(const ASource:Ansistr;var ADest:Ansistr;var ALen,AOffset:Int);
19584: Func HasEncoding( const AStr : Ansistr) :Bool;
19585: Func DecodeXmlAttribute( const AStr :Str) :Str;
19586: Proc ReallocateString(var AStr:Ansistr; var ALen : Int; AReqLen : Int);
19587: Proc AttrFillXMLString(AnAttr:IAttribute;var aString:Ansistr;var aOffset,aLen : Int);
19588: Proc FillXMLString(ANode:INode;var AStr:Str;var AOffset,ALen:Int;ASiblig:INode;ALevel:Int)
19589: Func NodeToXML( ANode : INode ) :Str;
19590: Proc XMLSaveToFile( ANode : INode; const AFileName :Str);
19591: Func XMLLoadFromFile( const AFileName :Str) : INode;
19592: Func Hashmax( const ASource : Ansistr) :Card;
19593: ConstantN('GXMLIndentSpaces','Int').SetInt( 2);
19594: //ConstantN('GXMLMultiLineAttributes','Boolean')BoolToStr( True);
19595: GXMLMultiLineAttributeThreshold,'Int').SetInt( 7);
19596: end;
19597:
19598: Proc SIRegister_MaxDOMDictionary(CL: TPSPascalCompiler);
19599: begin
19600:   SIRegister_IDictionary(CL);
19601:   SIRegister_TDictionary(CL);
19602:   Func HashFast( const AKey :Str) :Card;
19603:   Func HashCarlos( const AKey :Str) :Card;
19604:   Func BorlandHashOf( const AKey :Str) :Card;
19605:   Func HashSumOfChars( const AKey :Str) :Card;
19606: end;
19607:
19608: Proc SIRegister_MaxDOM(CL: TPSPascalCompiler);
19609: begin
19610:   TypeS('TNodeType', '( ntElement, ntText, ntCDATA, ntComment );
19611:   Interface(CL.FindInterface('IUNKNOWN'),INode, 'INode;
19612:   Interface(CL.FindInterface('IUNKNOWN'),IAttribute, 'IAttribute;
19613:   Interface(CL.FindInterface('IUNKNOWN'),IAttributeCollection, 'IAttributeCollection;
19614:   Interface(CL.FindInterface('IUNKNOWN'),INodeCollection, 'INodeCollection;
19615:   SIRegister_INode(CL);
19616:   SIRegister_IAttribute(CL);
19617:   SIRegister_IAttributeCollection(CL);
19618:   SIRegister_INodeCollection(CL);
19619: Func NodeCreate( const ANodeName :Str; ANodeType : TNodeType) : INode;
19620: SIRegister_TAttribute(CL);
19621: TypeS(TNodes', 'array of INode;
19622: TypeS(TAttributes2', 'array of IAttribute;
19623: TypeS(THashedAttributes', 'record Attributes : TAttributes2; AttrCount '
19624:   +': Int; AttrCapacity : Int; end;
19625: TypeS(TAttrHashTable', 'array of THashedAttributes;
19626: SIRegister_TNode(CL);
19627: //TypeS('TNodeClass', 'class of TNode;
19628: //TypeS('TAttributeClass', 'class of TAttribute;
19629: Func PointerToStr( P : __Pointer) :Str;
19630: Func StrToPointer( const S :Str) : __Pointer;
19631: Func INodeToStr( ANode : INode) :Str;
19632: Func StrToINode( const S :Str) : INode;
19633: Func CompareByNodeName( N1, N2 : INode) : Int;
19634: Func CompareByNameAttr( N1, N2 : INode) : Int;
19635: end;
19636:
19637: Proc SIRegister_cASN1(CL: TPSPascalCompiler);
19638: begin
19639:   ClassN(CL.FindClass('TOBJECT'),'EASN1;
19640:   'ASN1_ID_END_OF_CONTENT','LongWord').SetUInt( $00);

```

```

19641: ConstantN('ASN1_ID_BOOLEAN','LongWord').SetUInt( $01);
19642: 'ASN1_ID_Int','LongWord').SetUInt( $02);
19643: 'ASN1_ID_BIT_STRING','LongWord').SetUInt( $03);
19644: 'ASN1_ID_OCTET_STRING','LongWord').SetUInt( $04);
19645: 'ASN1_ID_NULL','LongWord').SetUInt( $05);
19646: 'ASN1_ID_OBJECT_IDENTIFIER','LongWord').SetUInt( $06);
19647: 'ASN1_ID_OBJECT_DESCRIPTOR','LongWord').SetUInt( $07);
19648: 'ASN1_ID_EXTERNAL','LongWord').SetUInt( $08);
19649: 'ASN1_ID_REAL','LongWord').SetUInt( $09);
19650: 'ASN1_ID_ENUMERATED','LongWord').SetUInt( $0A);
19651: 'ASN1_ID_EMBEDDED_PDV','LongWord').SetUInt( $0B);
19652: 'ASN1_ID_UTF8STRING','LongWord').SetUInt( $0C);
19653: 'ASN1_ID_RELATIVE_OID','LongWord').SetUInt( $0D);
19654: 'ASN1_ID_NUMERICSTRING','LongWord').SetUInt( $12);
19655: 'ASN1_ID_PRINTABLESTRING','LongWord').SetUInt( $13);
19656: 'ASN1_ID_T61STRING','LongWord').SetUInt( $14);
19657: 'ASN1_ID_VIDEOTEXSTRING','LongWord').SetUInt( $15);
19658: 'ASN1_ID_IA5STRING','LongWord').SetUInt( $16);
19659: 'ASN1_ID_UTCTIME','LongWord').SetUInt( $17);
19660: 'ASN1_ID_GENERALIZEDTIME','LongWord').SetUInt( $18);
19661: 'ASN1_ID_GRAPHICSTRING','LongWord').SetUInt( $19);
19662: 'ASN1_ID_VISIBLESTRING','LongWord').SetUInt( $1A);
19663: 'ASN1_ID_GENERALSTRING','LongWord').SetUInt( $1B);
19664: 'ASN1_ID_UNIVERSALSTRING','LongWord').SetUInt( $1C);
19665: 'ASN1_ID_CHARACTERSTRING','LongWord').SetUInt( $1D);
19666: 'ASN1_ID_BITMAPSTRING','LongWord').SetUInt( $1E);
19667: 'ASN1_ID_SEQUENCE','LongWord').SetUInt( $30);
19668: 'ASN1_ID_SET','LongWord').SetUInt( $31);
19669: 'ASN1_ID_CONSTRUCTED','LongWord').SetUInt( $20);
19670: 'ASN1_ID_APPLICATION','LongWord').SetUInt( $40);
19671: 'ASN1_ID_CONTEXT_SPECIFIC','LongWord').SetUInt( $80);
19672: 'ASN1_ID_PRIVATE','LongWord').SetUInt( $C0);
19673: TASN1ObjectIdentifier, 'array of Int;
19674: TASN1ParseProc', 'procedure(const TypeID:Byte; const DataBuf:Str; const DataSize: Int; const ObjectIdx:
Int; const CallerData: Int);;
19675: {TASN1ParseProc =
19676: Proc (const TypeID: Byte; const DataBuf; const DataSize: Int;
19677: const ObjectIdx: Int; const CallerData: Int); }
19678:
19679: Proc ASN1OIDInit( var A : TASN1ObjectIdentifier; const B : array of Int);
19680: Func ASN1OIDToStr( const A : TASN1ObjectIdentifier) : Ansistr;
19681: Func ASN1OIDEqual(const A:TASN1ObjectIdentifier; const B : array of Int) :Bool;
19682: Func ASN1EncodeLength( const Len : Int) : Ansistr;
19683: Func ASN1EncodeObj( const TypeID : Byte; const Data : Ansistr) : Ansistr;
19684: Func ASN1EncodeEndOfContent : Ansistr;
19685: Func ASN1EncodeNull : Ansistr;
19686: Func ASN1EncodeBoolean( const A :Bool) : Ansistr;
19687: Func ASN1EncodeDataInt8( const A : ShortInt) : Ansistr;
19688: Func ASN1EncodeDataInt16( const A : SmallInt) : Ansistr;
19689: Func ASN1EncodeDataInt24( const A : LongInt) : Ansistr;
19690: Func ASN1EncodeDataInt32( const A : LongInt) : Ansistr;
19691: Func ASN1EncodeDataInt64( const A : Int64) : Ansistr;
19692: Func ASN1EncodeInt8( const A : ShortInt) : Ansistr;
19693: Func ASN1EncodeInt16( const A : SmallInt) : Ansistr;
19694: Func ASN1EncodeInt24( const A : LongInt) : Ansistr;
19695: Func ASN1EncodeInt32( const A : LongInt) : Ansistr;
19696: Func ASN1EncodeInt64( const A : Int64) : Ansistr;
19697: Func ASN1EncodeIntBuf( const A :Str; const Size : Int) : Ansistr;
19698: Func ASN1EncodeIntBufStr( const A : Ansistr) : Ansistr;
19699: Func ASN1EncodeEnumerated( const A : Int64) : Ansistr;
19700: Func ASN1EncodeBitString(const A: Ansistr; const UnusedBits : Byte): Ansistr;
19701: Func ASN1EncodeOctetString( const A : Ansistr) : Ansistr;
19702: Func ASN1EncodeInt32AsOctetString( const A : LongInt) : Ansistr;
19703: Func ASN1EncodeUTF8String( const A : Ansistr) : Ansistr;
19704: Func ASN1EncodeIA5String( const A : Ansistr) : Ansistr;
19705: Func ASN1EncodeVisibleString( const A : Ansistr) : Ansistr;
19706: Func ASN1EncodeNumericString( const A : Ansistr) : Ansistr;
19707: Func ASN1EncodePrintableString( const A : Ansistr) : Ansistr;
19708: Func ASN1EncodeTeletexString( const A : Ansistr) : Ansistr;
19709: Func ASN1EncodeUniversalString( const A : WideString) : Ansistr;
19710: Func ASN1EncodeBMPString( const A : WideString) : Ansistr;
19711: Func ASN1EncodeUTCTime( const A : TDateTime) : Ansistr;
19712: Func ASN1EncodeGeneralizedTime( const A : TDateTime) : Ansistr;
19713: Func ASN1EncodeOID( const OID : array of Int) : Ansistr;
19714: Func ASN1EncodeSequence( const A : Ansistr) : Ansistr;
19715: Func ASN1EncodeSet( const A : Ansistr) : Ansistr;
19716: Func ASN1EncodeContextSpecific( const I : Int; const A : Ansistr) : Ansistr;
19717: Func ASN1DecodeLength(const Buf:Str;const Size:Int;var Len:Int) : Int;
19718: Func ASN1DecodeObjHeader(const Buf:str;const Size:Int;var TypeID:Byte;var Len:Int;var Data:str):Int;
19719: Func ASN1TypeIsConstructedType( const TypeID : Byte) :Bool;
19720: Func ASN1TypeIsContextSpecific( const TypeID : Byte; var Idx : Int) :Bool;
19721: Func ASN1DecodeDataBoolean(const Buf:str;const Size:Int; var A:Boolean) : Int;
19722: Func ASN1DecodeDataInt32(const Buf:str;const Size:Int;var A:LongInt): Int;
19723: Func ASN1DecodeDataInt64(const Buf:str; const Size:Int; var A:Int64): Int;
19724: Func ASN1DecodeDataIntBuf(const Buf:str; const Size:Int;var A: Ansistr): Int;
19725: Func ASN1DecodeDataBitString(const Buf:str; const Size:Int;var A:Ansistr;var UnusedBits:Byte) : Int;
19726: Func ASN1DecodeDataRawAnsistr(const Buf:str;const Size:Int;var A:Ansistr):Int;
19727: Func ASN1DecodeDataOctetString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19728: Func ASN1DecodeDataIA5String(const Buf:str;const Size:Int;var A:Ansistr): Int;

```

```

19729: Func ASN1DecodeDataVisibleString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19730: Func ASN1DecodeDataNumericString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19731: Func ASN1DecodeDataPrintableString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19732: Func ASN1DecodeDataTeletexString(const Buf:str;const Size:Int; var A:Ansistr):Int;
19733: Func ASN1DecodeDataUTF8String(const Buf:str; const Size:Int;var A:Ansistr):Int;
19734: Func ASN1DecodeDataUniversalString(const Buf:str;const Size:Int;var A:Ansistr):Int;
19735: Func ASN1DecodeDataBMPString(const Buf:Str; const Size:Int; var A: Ansistr):Int;
19736: Func ASN1DecodeDataOID(const Buf:str;const Size:Int;var A:TASN1ObjectIdentifier):Int;
19737: Func ASN1DecodeDataUTCTime(const Buf:str; const Size:Int;var A:TDateTime): Int;
19738: Func ASN1DecodeDataGeneralizedTime(const Buf:str;const Size:Int;var A:TDateTime):Int;
19739: Func ASN1Parse(const Buf:str;const Size:Int;const ParseProc:TASN1ParseProc;const CallerData:Int):Int;
19740: Func ASN1DecodeBoolean(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var A:Bool):Int;
19741: Func ASN1DecodeInt32(const TypeID:Byte;const DataBuf:str;const DataSize: Int; var A : LongInt): Int;
19742: Func ASN1DecodeInt64(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var A:Int64):Int;
19743: Func ASN1DecodeIntBuf(const TypeID:Byte;const DataBuf:Str; const DataSize: Int;var A : Ansistr):Int;
19744: Func ASN1DecodeBitString(const TypeID:Byte;const DataBuf:str; const DataSize:Int;var A:Ansistr;var
UnusedBits:Byte):Int;
19745: Func ASN1DecodeString(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var A:Ansistr):Int;
19746: Func ASN1DecodeOID(const TypeID:Byte;const DataBuf:str;const DataSize:Int;var
A:TASN1ObjectIdentifier):Int;
19747: Func ASN1DecodeTime const TypeID:Byte;const DataBuf:str; const DataSize:Int;var A:TDateTime): Int;
19748: Proc SelfTestASN1;
19749: end;
19750:
19751: Proc SRegister_cX509Certificate(CL: TPSPascalCompiler);
19752: begin
19753:   ClassN(CL.FindClass('TOBJECT'),'EX509;
19754:   TypeS(TX509AttributeType','TASN1ObjectIdentifier;
19755:   TypeS(TX509AttributeValue','Ansistr;
19756:   TypeS(TX509AttributeTypeAndValue','record AType : TX509AttributeType;
19757:   + ' Value : TX509AttributeValue; _Decoded :Bool; end;
19758:   //TypeS('PX509AttributeTypeAndValue','TX509AttributeTypeAndValue // wil not work;
19759:   Proc InitX509AttributeTypeAndValue(var A : TX509AttributeTypeAndValue; const AType : TX509AttributeType;
const Value : TX509AttributeValue);
19760:   Proc InitX509AtName( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19761:   Proc InitX509AtSurname( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19762:   Proc InitX509AtGivenName( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19763:   Proc InitX509AtInitials( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19764:   Proc InitX509AtGenerationQuailifier(var A:TX509AttributeTypeAndValue; const B:Ansistr);
19765:   Proc InitX509AtCommonName(var A: TX509AttributeTypeAndValue; const B : Ansistr);
19766:   Proc InitX509AtLocalityName(var A:TX509AttributeTypeAndValue;const B: Ansistr);
19767:   Proc InitX509AtStateOrProvince(var A:TX509AttributeTypeAndValue;const B:Ansistr);
19768:   Proc InitX509AtOriganizationName(var A: TX509AttributeTypeAndValue; const B:Ansistr);
19769:   Proc InitX509AtOriganizationUnitName(var A:TX509AttributeTypeAndValue;const B:Ansistr);
19770:   Proc InitX509AtTitle( var A : TX509AttributeTypeAndValue; const B : Ansistr);
19771:   Proc InitX509AtDnQualifier(var A: TX509AttributeTypeAndValue; const B: Ansistr);
19772:   Proc InitX509AtCountryName(var A :TX509AttributeTypeAndValue; const B : Ansistr);
19773:   Proc InitX509AtEmailAddress(var A: TX509AttributeTypeAndValue; const B: Ansistr);
19774:   Func EncodeX509AttributeTypeAndValue(const A:TX509AttributeTypeAndValue):Ansistr;
19775:   Func DecodeX509AttributeTypeAndValue(const TypeID Byte;const Buf:str; const Size:Int;var
A:TX509AttributeTypeAndValue): Int;
19776:   TypeS('TX509RelativeDistinguishedName','array of TX509AttributeTypeAndValue;
19777:   // TypeS('PX509RelativeDistinguishedName','TX509RelativeDistinguishedName //will not work;
19778:   Proc AppendX509RelativeDistinguishedName( var A : TX509RelativeDistinguishedName; const V :
TX509AttributeTypeAndValue);
19779:   Func EncodeX509RelativeDistinguishedName(const A:TX509RelativeDistinguishedName):AnsiStr;
19780:   Func DecodeX509RelativeDistinguishedName(const TypeID:Byte; const Buf:Str; const Size:Int;var
A:TX509RelativeDistinguishedName): Int;
19781:   TypeS('TX509RDNSSequence','array of TX509RelativeDistinguishedName;
19782:   //TypeS('PX509RDNSSequence','TX509RDNSSequence // will not work;
19783:   TypeS('TX509Name','TX509RDNSSequence;
19784:   Proc AppendX509RDNSSequence(var A:TX509RDNSSequence;const B:TX509RelativeDistinguishedName);
19785:   Func EncodeX509RDNSSequence( const A : TX509RDNSSequence) : Ansistr;
19786:   Func DecodeX509RDNSSequence(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RDNSSequence):Int;
19787:   Func EncodeX509Name( const A : TX509Name) : Ansistr;
19788:   Func DecodeX509Name(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Name):Int;
19789:   'CurrentX509Version','string').SetString(' X509v3;
19790:   type
19791:   TX509Version = (
19792:     X509v1 = 0,
19793:     X509v2 = 1,
19794:     X509v3 = 2,
19795:     X509vUndefined = $FF // implementation defined value );
19796:   TX509Version',' (X509v1, X509v2,X509v3, X509vUndefined);
19797:
19798:   type
19799:   TX509GeneralNameType = (
19800:     gnOtherName = 0,
19801:     gnRFC822Name = 1,
19802:     gnDNSName = 2,
19803:     gnX400Address = 3,
19804:     gnDirectoryName = 4,
19805:     gnEDIPartyName = 5,
19806:     gnUniformResourceIdentifier = 6,
19807:     gnIPAddress = 7,
19808:     gnRegisteredID = 8);
19809:   TypeS(TX509GeneralNameType',(gnOtherName,gnRFC822Name,gnDNSName,gnX400Address,gnDirectoryName,
gnEDIPartyName,gnUniformResourceIdentifier,gnIPAddress,gnRegisteredID);
19810:

```



```

19811: Proc InitX509Version( var A : TX509Version);
19812: Func EncodeX509Version( const A : TX509Version) : Ansistr;
19813: Func DecodeX509Version(const TypeID:Byte;const Buf:str;const Size:Int;var A: TX509Version):Int;
19814: TypeS(TX509Time', TDateTime;
19815: Func EncodeX509Time( const A : TX509Time) : Ansistr;
19816: Func DecodeX509Time(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Time):Int;
19817: TX509Validity', record NotBefore:TX509Time; NotAfter:TX509Time; _Decoded:Bool; end;
19818: // TypeS('PX509Validity', '^TX509Validity // will not work;
19819: Func EncodeX509Validity( const A : TX509Validity) : Ansistr;
19820: Func DecodeX509Validity(const TypeID:Byte;const Buf:Str; const Size:Int; var A: TX509Validity): Int;
19821: TypeS('TX509DHValidationParms', record Seed: Ansistr; PgenCounter : Int; end;
19822: //TypeS('PX509DHValidationParms', '^TX509DHValidationParms // will not work;
19823: TypeS('TX509DHDomainParameters', record P : Ansistr; G : Ansistr'
19824: +; Q : Ansistr; J : Ansistr; ValidationParms : TX509DHValidationParms; end;
19825: // TypeS('PX509DHDomainParameters', '^TX509DHDomainParameters // will not work;
19826: Func EncodeX509DHValidationParms( const A : TX509DHValidationParms) : Ansistr;
19827: Func DecodeX509DHValidationParms(const TypeID:Byte;const Buf:str;const Size:Int;var
A:TX509DHValidationParms): Int;
19828: Func EncodeX509DHDomainParameters( const A: TX509DHDomainParameters) : Ansistr;
19829: Func DecodeX509DHDomainParameters(const TypeID:Byte;const Buf:str;const Size:Int;var
A:TX509DHDomainParameters):Int;
19830: TypeS('TX509DSSParms', record P : Ansistr; Q : Ansistr; G : Ansistr; end;
19831: //TypeS('PX509DSSParms', '^TX509DSSParms // will not work;
19832: Func EncodeX509DSSParms( const A : TX509DSSParms) : Ansistr;
19833: Func DecodeX509DSSParms(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509DSSParms):Int;
19834: TypeS('TX509DSSSigValue', record R : Ansistr; S : Ansistr; end;
19835: //TypeS('PX509DSSSigValue', '^TX509DSSSigValue // will not work;
19836: Func EncodeX509DSSSigValue( const A : TX509DSSSigValue) : Ansistr;
19837: Func DecodeX509DSSSigValue(const TypeID:Byte; const Buf:Str; const Size:Int; var A:TX509DSSSigValue):Int;
19838: TypeS(TX509AlgorithmIdentifier', record Algorithm : TASN1ObjectIdentifi
19839: +fier; Parameters : Ansistr; _Decoded :Bool; end;
19840: //TypeS('PX509AlgorithmIdentifier', '^TX509AlgorithmIdentifier // will not work;
19841: Proc InitX509AlgorithmIdentifier(var A:TX509AlgorithmIdentifier; const Algorithm : array of Int; const
Parameters : Ansistr);
19842: Proc InitX509AlgorithmIdentifierDSA_SHA1(var A:TX509AlgorithmIdentifier;const Params:Ansistr;
19843: Func EncodeX509AlgorithmIdentifier( const A : TX509AlgorithmIdentifier) : Ansistr;
19844: Func DecodeX509AlgorithmIdentifier(const TypeID:Byte; const Buf:str; const Size:Int; var
A:TX509AlgorithmIdentifier): Int;
19845: TypeS(TX509RSAPublicKey', record Modulus:Ansistr;PublicExponent:Ansistr; end;
19846: //TypeS('PX509RSAPublicKey', '^TX509RSAPublicKey // will not work;
19847: Func EncodeX509RSAPublicKey( const A : TX509RSAPublicKey) : Ansistr;
19848: Func DecodeX509RSAPublicKey(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RSAPublicKey): Int;
19849: Func ParseX509RSAPublicKey(const Buf:str;const Size:Int;var A:TX509RSAPublicKey):Int);
19850: TypeS('TX509DHPublicKey', 'Ansistr;
19851: Func EncodeX509DHPublicKey( const A : TX509DHPublicKey) : Ansistr;
19852: TypeS(TX509DSAPublicKey', 'Ansistr;
19853: Func EncodeX509DSAPublicKey( const A : TX509DSAPublicKey) : Ansistr;
19854: TypeS(TX509SubjectPublicKeyInfo', record Algorithm : TX509AlgorithmId'
19855: +entifier; SubjectPublicKey : Ansistr; _Decoded :Bool; end;
19856: //TypeS('PX509SubjectPublicKeyInfo', '^TX509SubjectPublicKeyInfo // will not work;
19857: Proc InitX509SubjectPublicKeyInfoDSA(var A:TX509SubjectPublicKeyInfo;const B:TX509DSSParms;const
PublicKey: Ansistr);
19858: Func EncodeX509SubjectPublicKeyInfo( const A: TX509SubjectPublicKeyInfo) : Ansistr;
19859: Func DecodeX509SubjectPublicKeyInfo(const TypeID:Byte;const Buf:str;const Size:Int;var A
TX509SubjectPublicKeyInfo): Int;
19860: Func ParseX509SubjectPublicKeyInfo(const Buf:str;const Size:Int;var A:TX509SubjectPublicKeyInfo):Int;
19861: Func EncodeX509GeneralName(const A:TX509GeneralNameType;const EncodedName:Ansistr):Ansistr;
19862: Func DecodeX509GeneralName(const TypeID:Byte; const Buf:str; const Size:Int;var
A:TX509GeneralNameType;var B Ansistr): Int;
19863: TX509GeneralNames', array of record NameType:TX509GeneralNameType; Name : Ansistr; end;
19864: //TypeS('PX509GeneralNames', '^TX509GeneralNames // will not work;
19865: Func EncodeX509GeneralNames( const A : TX509GeneralNames) : Ansistr;
19866: Func DecodeX509GeneralNames( const TypeID:Byte;const Buf:str; const Size:Int; var A:TX509GeneralNames):Int;
19867: TX509BasicConstraints, record CA:Bool;PathLenConstraint:Ansistr; _DecodedCA:Boolean;end
19868: //TypeS('PX509BasicConstraints', '^TX509BasicConstraints // will not work;
19869: Func EncodeX509BasicConstraints( const A : TX509BasicConstraints) : Ansistr;
19870: Func DecodeX509BasicConstraints(const TypeID:Byte;const Buf:str;const Size:Int;var
A:TX509BasicConstraints): Int;
19871: TypeS(TX509AuthorityKeyIdentifier', record KeyIdentifier : Ansistr'
19872: +; AuthorityCertIssuer : TX509GeneralNames; AuthorityCertSerialNumber : Int64; end;
19873: //TypeS('PX509AuthorityKeyIdentifier', '^TX509AuthorityKeyIdentifier //will not work;
19874: Func EncodeX509AuthorityKeyIdentifier(const A:TX509AuthorityKeyIdentifier): Ansistr;
19875: TypeS(TX509SubjectKeyIdentifier', 'Ansistr;
19876: TypeS('TX509KeyUsage', 'Ansistr;
19877: TypeS(TX509Extension', record ExtnID : TASN1ObjectIdentifier; Critica'
19878: +l :Bool; ExtnValue: Ansistr; _Decoded :Bool; end;
19879: //TypeS('PX509Extension', '^TX509Extension // will not work;
19880: Proc InitX509ExtAuthorityKeyIdentifier(var A: TX509Extension; const B TX509AuthorityKeyIdentifier);
19881: Proc InitX509ExtSubjectKeyIdentifier(var A:TX509Extension; const B:TX509SubjectKeyIdentifier);
19882: Proc InitX509ExtBascConstraints(var A:TX509Extension;const B:TX509BasicConstraints);
19883: Func EncodeX509Extension( const A : TX509Extension) : Ansistr;
19884: Func DecodeX509Extension(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509Extension):Int;
19885: TypeS('TX509Extensions', 'array of TX509Extension;
19886: //TypeS('PX509Extensions', '^TX509Extensions // will not work;
19887: Proc AppendX509Extensions( var A : TX509Extensions; const B : TX509Extension);
19888: Func EncodeX509Extensions( const A : TX509Extensions) : Ansistr;
19889: Func DecodeX509Extensions( const TypeID: Byte; const Buf:Str;const Size: Int;var A: TX509Extensions):Int;
19890: Func NormaliseX509IntKeyBuf( var KeyBuf : Ansistr) : Int;
19891: TypeS(TX509TBSCertificate', record Version : TX509Version; SerialNumb'

```

```

19892:   +er : Ansistr; Signature : TX509AlgorithmIdentifier; Issuer : TX509Name;
19893:   + Validity : TX509Validity; Subject : TX509Name; SubjectPublicKeyInfo : TX5
19894:   +09SubjectPublicKeyInfo; IssuerUniqueID : Ansistr; SubjectUniqueID : Ans
19895:   +tString; Extensions: TX509Extensions; DecodedVersion :Bool; _Decoded:Bool; end;
19896: // TypeS('PX509TBSCertificate', '^TX509TBSCertificate // will not work;
19897: Func EncodeX509TBSCertificate( const A : TX509TBSCertificate ) : Ansistr;
19898: Func DecodeX509TBSCertificate(const TypeID:Byte;const Buf:str;const Size:Int;var
A:TX509TBSCertificate):Int;
19899: TypeS('TX509Certificate', record TBSCertificate : TX509TBSCertificate;
19900:   + SignatureAlgorithm: TX509AlgorithmIdentifier;SignatureValue:Ansistr; _Decoded:Bool;end;
19901: // TypeS('PX509Certificate', '^TX509Certificate // will not work;
19902: TypeS('TX509CertificateArray', 'array of TX509Certificate;
19903: Proc InitX509Certificate( var A : TX509Certificate);
19904: Func EncodeX509Certificate( const A : TX509Certificate ) : Ansistr;
19905: Func DecodeX509Certificate(const TypeID:Byte; const Buf:str;const Size: Int; var A:TX509Certificate):Int;
19906: Func ParseX509Certificate(const Buf:str;const Size:Int;var A:TX509Certificate):Int;
19907: Proc ParseX509CertificateStr( const BufStr : Ansistr; var A : TX509Certificate);
19908: Proc ParseX509CertificatePEM( const BufStr : Ansistr; var A : TX509Certificate);
19909: TypeS('TX509RSAPrivateKey', record Version : Int; Modulus : AnsiSt
19910:   +ring; PublicExponent : Ansistr; PrivateExponent : Ansistr; Prime1 : '
19911:   +Ansistr; Prime2 Ansistr;Exponent1:Ansistr;Exponent2:Ansistr;CRTCoefficient:Ansistr; _Decoded:Bool; end;
19912: // TypeS('PX509RSAPrivateKey', '^TX509RSAPrivateKey // will not work;
19913: Func EncodeX509RSAPrivateKey( const A : TX509RSAPrivateKey ) : Ansistr;
19914: Func DecodeX509RSAPrivateKey(const TypeID:Byte;const Buf:str;const Size:Int;var A:TX509RSAPrivateKey):Int;
19915: Func ParseX509RSAPrivateKey(const Buf:str;const Size:Int;var A:TX509RSAPrivateKey):Int;
19916: Proc ParseX509RSAPrivateKeyStr(const BufStr: Ansistr;var A: TX509RSAPrivateKey);
19917: Proc ParseX509RSAPrivateKeyPEM(const BufStr: Ansistr;var A: TX509RSAPrivateKey);
19918: Proc SelfTestX509;
19919: end;
19920:
19921: Proc SIRegister_FileStreamW(CL: TPSPascalCompiler);
19922: begin
19923:   SIRegister_TFileStreamW(CL);
19924:   {TFormFadeSettings = record
19925:     Form: TForm;
19926:     Step: ShortInt;
19927:     DisableBlendOnFinish:Bool;
19928:     Callback: TNotifyEvent;
19929:     MinAlpha, MaxAlpha: Byte;
19930:   end;}
19931:   TFormFadeSettings', 'record Form:TForm;Step:ShortInt;DisableBlendOnFinish:Bool;Callback:TNotifyEvent;
MinAlpha,MaxAlpha:Byte;end;
19932: ConstantN('fmForcePath','LongWord').SetUInt( $80000000);
19933: Func LoadUnicodeFromStream( S : TStream; AsIsAnsi :Bool ) : WideString;
19934: Func GetClipboardText: WideString;;
19935: Proc CopyToClipboard(Str: WideString);;
19936: Func CurrentWinUser: WideString;;
19937: Func GetTempPathW: WideString;;
19938: Func GetTempFileNameW: WideString;;
19939: Func GetDesktopFolderW: WideString;;
19940: Func IsWritable(const FileName: WideString):Bool;;
19941: Func SysErrorMessageW(ErrorCode: Int): WideString;;
19942: Func FormatExceptionInfo: WideString;;
19943: Proc ShowExceptionW(Message: WideString);;
19944: Proc ChangeWindowStyle(const Form: HWND; Style: DWord; AddIt:Bool);;
19945: Func SetNtfsCompression(const FileName: WideString; Level: Word):Bool;;
19946: Func WriteWS(const Stream: TStream; const Str: WideString): Word;;
19947: Proc FormFadeIn(Form: TForm; Step: ShortInt);;
19948: Proc FormFadeOut(Form: TForm; Step: ShortInt);;
19949: Proc FormFadeOutAndWait(Form: TForm; Step: ShortInt);;
19950: Func BrowseForFolderw(const Caption,DefaultPath:WideStr;const OwnerWindow:HWND):WideStr;
19951: Proc FormFade(const Settings: TFormFadeSettings);;
19952: Func HashOfString(const Str: WideString): DWord;
19953: Func ComparePoints(const First, Second: TPoint): ShortInt;
19954: end;
19955:
19956: Proc SIRegister_InetUtils(CL: TPSPascalCompiler);
19957: begin
19958:   TypeS(TNetUtilsSettings', 'record UserAgent :Str; ProxyURL : Strin'
19959:   +g; OpenURLFlags: DWord; TrafficCounter:Dword;UploadedCounter:DWord;ReadBufferSize:DWord;end;
19960:   TRawCharset', 'set of Char;
19961:   TInetHeaders', 'array of String;
19962:   SIRegister_EInet(CL);
19963:   TInetDownloadCallback', 'Func ( Downloaded, TotalSize : DWord) :Bool;
19964:   Func InetDownloadTo(const DestFile:WideString;const URL:str;Callback:TInetDownloadCallback):Bool;
19965:   Func InetDownloadToI(const DestFile:WideString;const URL:Str;const Settings:
TNetUtilsSettings;Callback:TInetDownloadCallback):Bool;;
19966:   Func InetDownload(const URL:str;Dest:TStream;Callback:TInetDownloadCallback):Bool;
19967:   Func InetDownload3(const URL:Str; Dest:TStream; const Settings:TNetUtilsSettings;Callback :
TInetDownloadCallback) :Bool;;
19968:   Func InetBufferedReadFrom(Handle:HInternet;Dest:TStream;const
Settings:TNetUtilsSettings;Callback:TInetDownloadCallback):DWord;;
19969:   Func InetBufferedReadFrom5( Handle : HInternet; const Settings : TNetUtilsSettings; Callback :
TInetDownloadCallback) :Str;;
19970:   Func IsResponseStatusOK( Handle : HInternet) :Bool;
19971:   TMultipartItem', 'record Headers : TInetHeaders; Data : TStream; end;
19972:   { TMaskMatchInfo = record
19973:     Matched:Bool;
19974:     StrPos: Word;

```

```

19975:     MatchLength: Word;
19976: end;
19977: TMaskMatchInfo', 'record Matched:Bool; StrPos: Word; MatchLength: Word; end;
19978: TMultipartItems', 'array of TMultiPartItem;
19979: TUploadFile', 'record Name:str; SourceFileName : WideString; Data : TStream; end;
19980: TUploadFiles', 'array of TUploadFile;
19981: Func FindBoundaryFor( const Items : TMultipartItems) :Str;
19982: Func RandomBoundary :Str;
19983: Func GenerateMultipartFormFrom(const Items:TMultipartItems;out ExtraHeadsr:TInetHeaders):Str
19984: Func InetUploadTo( const ToURL:Str;const Headers:TInetHeaders;const Items: TMultipartItems; const
Settings:TNetUtilsSettings) :Str;;
19985: Func InetUploadTo7(const ToURL:Str;const Items:TMultipartItems;const Settings:TNetUtilsSettings):str;
19986: Func InetUploadTo8(const ToURL:Str; const Items: TMultipartItems) :Str;;
19987: Func InetUploadStreamsTo(const ToURL:Str;const Settings:TNetUtilsSettings;Streams:TUploadFiles):str;
19988: Func InetUploadStreamsTo10(const ToURL:Str;const Streams:TUploadFiles):str;
19989: Func InetUploadFileTo(const ToURL:Str;const Settings:TNetUtilsSettings;const ItemName:str;const
FilePath:Widestring):str;
19990: Func InetUploadFileTo12(const ToURL:Str;const ItemName:Str;const FilePath:Widestr):str;
19991: Func InetUploadFilesTo(const ToURL:Str;const Settings:TNetUtilsSettings;const Files:array of const):str;
19992: Func InetUploadFilesTo14(const ToURL:Str;const Files: array of const):str;
19993: Func AppendQueryTo(const URL:Str; const Arguments: array of const):str;
19994: Func HasQueryPart( const URL :Str) :Bool;
19995: Func BuildQueryFrom( const Arguments : array of const) :Str;
19996: Func BuildURLW(Protocol,Host:Str;Port:Word;Path,Script:Str;const Arguments:array of const):Str
19997: Func CustomEncode(const Str: WideString; const RawChars:TRawCharSet):Str;
19998: Func EncodeURI( const Str : WideString) :Str;
19999: Func EncodeURIComponent( const Str : WideString) :Str;
20000: Proc InetGetLastError( out ErrorCode : DWord; out ErrorMessage :Str);
20001: Func InetGetLastErrorCode : DWord;
20002: Func InetGetLastErrorMsg :Str;
20003: Func AbsoluteURLFrom( URL, BaseURL, BasePath :Str) :Str;
20004: Proc SplitURL( const URL :Str; out Domain, Path :Str);
20005: Func DomainOf( const URL :Str) :Str;
20006: Func PathFromURL( const URL :Str) :Str;
20007: Func InetHeaders( const NameValues : array of const) : TInetHeaders;
20008: Func NoInetHeaders : TInetHeaders;
20009: Func JoinHeaders( const Headers : TInetHeaders) :Str;
20010: ConstantN('InetHeaderEOLN','String').SetString( #13#10);
20011: Proc SetDefaultNetUtilsSettings;
20012: Func TotalDownTrafficThroughNetUtils : DWord;
20013: TDBDraw', 'record DisplayDC : HDC; MemDC : HDC; MemBitmap : HBIT
+MAP; OldBitmap : HBITMAP; OldFont : HFONT; OldPen : HPEN; end;
20014: TPieceFormatData', 'record Position : TMaskMatchInfo; Color : TColor; end;
20015: TFormatData', 'array of TPieceFormatData;
20016: TDrawFormattedTextSettings', 'record Text : WideString; FormatDa
+ta : TFormatData; Canvas : TCanvas; WrapText :Bool; DestPos : TPoint; '
20017: +MaxWidth : Word; CharSpacing : Word; end;
20018: TWrapTextSettings', 'record DC : HDC; Str : WideString; Delimite
+r :WideString; MaxWidth:Word; LeftMargin:Word; CharSpacing:Word;LastChar: TSize; end;
20019: Func TextSize( const DC : HDC; const Str : WideString) : TSize;
20020: Func TextWidthW2( const DC : HDC; const Str : WideString) : Int;
20021: Func TextHeightW2( const DC : HDC; const Str : WideString) : Int;
20022: Func GetLineHeightOf( const Font : HFONT) : Word;
20023: Func TextWidthEx(const DC:HDC;const Str:Widestring; const CharSpacing:Word): Int;
20024: Func TextHeightEx(const DC:HDC;const Str:Widestring; const CharSpacing:Word): Int;
20025: Func TextSizeEx(const DC:HDC;const Str:Widestring;const CharSpacing:Word):TSize;
20026: Func TextWithBreaksSize( Settings : TWrapTextSettings) : TSize;;
20027: Func DoubleBufferedDraw(const DisplaySurface:HDC;const BufferSize:TPoint):TDBDraw;
20028: Func DoubleBufferedDraw17(const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
20029: Func DoubleBufferedDraw2(const Canvas:TCanvas; const BufferSize:TPoint):TDBDraw;
20030: Proc DrawFormattedText( const Settings : TDrawFormattedTextSettings);
20031: Func GetLastCharPos(const DC:HDC;const Str:Widestring;const MaxWidth:Word;const ChrSpacing:Word):TSize;
20032: Func WrapNonMonospacedText(const DC:HDC;const Str:Widestring;const Delimiter:Widestring; const
MaxWidth:Word;const CharSpacing:Word):Widestring;;
20033: Func WrapNonMonospacedText2( var Settings : TWrapTextSettings) : WideString;;
20034: end;
20035: Proc SIRegister_StrConv(CL: TPSPascalCompiler);
20036: begin
20037: TypeS('TCodepage', 'DWord;
20038: ConstantN('CP_INVALID','LongInt').SetInt( TCodepage ( - 1 ));
20039: 'CP_ASIS','LongInt').SetInt( TCodepage ( - 2 ));
20040: 'CP_ANSI','longint').SetInt(0);
20041: 'CP_OEM','longint').SetInt(1);
20042: 'CP_SHIFTJIS','LongInt').SetInt( 932);
20043: 'CP_LATIN1','LongInt').SetInt( 1250);
20044: 'CP_UNICODE','LongInt').SetInt( 1200);
20045: 'CP_UTF8','LongInt').SetInt( 65001);
20046: Func MinStrConvBufSize( SrcCodepage : TCodepage; Str :Str) : Int;;
20047: Func MinStrConvBufSize1( DestCodepage : TCodepage; Wide : WideString) : Int;;
20048: Func ToWideString(SrcCodepage: TCodepage; Str :Str; BufSize: Int) : WideString;
20049: Func FromWideString(DestCodepage:TCodepage;Str:Widestring;BufSize:Int;Fail:Boolean):str;
20050: Func CharSetToID( Str :Str) : TCodepage;
20051: Func IDToCharset( ID : TCodepage; GetDescription :Bool) :Str;
20052: Func CompareStrW(const S1, S2: WideString; Flags: DWord = 0): Int;
20053: Func CompareTextW(const S1, S2: WideString): Int;
20054: Func MaskMatch(const Str, Mask: WideString):Bool;
20055: { Info can have special values in some cases:
20056: * Matched = True but MatchLength = 0(and StrPos having random value) - means that Mask consisted of only
20057: " " and

```

```

20061:      thus no particular substring could specified (since it could match anypart ofthe string). }
20062: Func MaskMatchInfo(const Str,Mask:WideString;StartingPos:Word = 1):TMaskMatchInfo;
20063: //Strutils -----
20064: Func TryStrToIntStrict(const S:Str; out Value:Int; Min:Int):Boolean;
20065: Func TryStrToFloatStrict(const S:Str; out Value:Single;const FormatSettings:TFormatSettings):Bool;
20066: Func TryStrToFloatStrict1(const S:Str;out Value:Double;const FormatSettings: TFormatSettings):Bool;
20067: Func DetectEolnStyleIn( const Str : WideString) : WideString;
20068: Func DetectEolnStyleInANSI( Stream : TStream) : WideString;
20069: Func PascalQuote( const Str : WideString) : WideString;
20070: Func StrRepeatW( const Str : WideString; Times : Int) : WideString;
20071: Func EscapeString(const Str: WideString; CharsToEscape:WideString): WideString;
20072: Func UnescapeString(const Str: WideString; CharsToEscape: WideString):WideString;
20073: Func BinToHexW( const Buf :Str; Delim :Str) :Str;
20074: Func HexToBinW( Text :Str) :Str;
20075: Func SoftHexToBin( Text :Str) :Str;
20076: Func FormatVersion( Version : Word) : WideString;
20077: Func FormatDateW( Date : DWord) : WideString;
20078: Func FormatNumber( Number : DWord) : WideString;
20079: Func GenericFormat( Number:Single;const Language:TGenericFormatLanguage):WideString;
20080: Func FormatInterval( Millisecs : DWord) : WideString;
20081: Func FormatSize( Bytes : DWord) : WideString;
20082: Func PosLast( const Substr, Str :Str; Start : Word) : Int;
20083: Func PosLastW( const Substr, Str : WideString; Start : Word) : Int;
20084: Func IsDelimiterW(const Delimiters,S: WideString; Index: Int):Bool;
20085: Func RemoveNonWordChars(const Str: WideString;DoNotRemove:WideString):WideString;
20086: Func IsQuoteChar( const aChr : Char) :Bool;
20087: Func WrapTextW(const Str:WideStr;const Delimiter:WideStr;const MaxWidth:Word):WideStr;
20088: Func PadText(const Str:WideStr;const NewLine,PadStr:WideStr;const MaxWidth:Word):WideStr;
20089: Func PadTextWithVariableLineLength(const Str:WideString;const NewLine,PadStr:WideString;const
LineLengths:array of Int):WideString;
20090: Func StrPadW(const Str:WideString; ToLength:Int; PadChar: WideChar) : WideString;
20091: Func StrPadLeftW(const Str:WideString;ToLength:Int;PadChar: WideChar): WideString;
20092: //Func StrRepeat( const Str : WideString; Times : Int) : WideString;
20093: Func StrReverseW( const Str : WideString) : WideString;
20094: Func CountSubstr( const Substr, Str : WideString) : Int;
20095: Proc DeleteArrayItem( var A : TWideStringArray; Index : Int);
20096: Func TrimStringArray( WSArray : TWideStringArray) : TWideStringArray;
20097: Func TrimWS( Str : WideString; const Chars : WideString) : WideString;
20098: Func TrimLeftWS( Str : WideString; const Chars : WideString) : WideString;
20099: //Func TrimRightWS( Str : WideString; const Chars : WideString) : WideString;
20100: Func ConsistsOfChars( const Str, Chars : WideString) :Bool;
20101: Func UpperCaseW( const Str : WideString) : WideString;
20102: Func LowerCaseW( const Str : WideString) : WideString;
20103: Func UpperCaseFirst( const Str : WideString) : WideString;
20104: Func LowerCaseFirst( const Str : WideString) : WideString;
20105: Func StripAccelChars( const Str : WideString) : WideString;
20106: end;
20107:
20108: Proc SIRegister_REXX(CL: TPSPascalCompiler);
20109: begin
20110:   TypeS(TState', '( TrimLeader, StartToken, EndToken) );
20111:   TStrIndex', 'LongInt;
20112:   TTokIndex', 'WORD;
20113:   TStrIndexB', 'BYTE;
20114:   TTokIndexB', 'BYTE;
20115:   ClassN(CL.FindClass('TOBJECT'),'EConversionError;
20116: Func Abbrev(const information, info:Str; const nMatch: TStrIndex):BOOLEAN;
20117: Func AllSame( const s :Str; const c : CHAR) :Bool;
20118: Func Capitalize( const s :Str) :Str;
20119: Func Center( const s :Str; const sLength : TStrIndex) :Str;
20120: Func Left( const s :Str; const sLength : TStrIndex) :Str;
20121: Func Right( const s :Str; const sLength : TStrIndex) :Str;
20122: Func Copies( const s :Str; const n : TStrIndex) :Str;
20123: Func CountChar( const s :Str; const c : CHAR) : TStrIndex;
20124: Func DeleteStringrexx( const substring :Str; const s :Str) :Str;
20125: Func Overlay( const ovly, target :Str; const n : TStrIndex) :Str;
20126: Func Plural(const n: LongInt; const singularform, pluralform :Str) :Str;
20127: Func Reverse( const s :Str) :Str;
20128: Func Spacerexx( const s :Str; const n : TStrIndex) :Str;
20129: Func Striprexx( const s :Str; const option :Str) :Str;
20130: Func TestString( const sLength : TStrIndex) :Str;
20131: Func Translate( const s, OutTable, InTable :Str) :Str;
20132: Func XRange( const start, stop : BYTE) :Str;
20133: Func B2X(const b: BYTE) :Str;
20134: Func C2D(const s:Str) : DOUBLE;
20135: Func C2I(const s:Str) : Int;
20136: Func C2L(const s:Str) : LONGINT;
20137: Func C2W(const s:Str) : WORD;
20138: Func C2X(const s:Str) :Str;
20139: Func I2C(const i: Int) :Str;
20140: Func I2X(const i: Int) :Str;
20141: Func L2C(const i: LONGINT) :Str;
20142: Func L2X(const i: LONGINT) :Str;
20143: Func D2C(const x: DOUBLE; const d : BYTE):Str;
20144: Func W2C(const w: WORD) :Str;
20145: Func W2X(const w: WORD) :Str;
20146: Func X2W(const s:Str) : WORD;
20147: Func JulianDate( const DateTime : TDateTime) : LongInt;
20148: Func TimeDifference( const StartTime, StopTime : TDateTime) : DOUBLE;

```



```

20149: Func Pwr( const x, y : DOUBLE ) : DOUBLE;
20150: end;
20151:
20152: Proc SIRegister_StringGridLibrary(CL: TPSPascalCompiler);
20153: begin
20154: Proc ReadGridFile( var StringGrid : TStringGrid; GridFile :Str);
20155: Proc WriteGridFile( var StringGrid : TStringGrid; GridFile :Str);
20156: Proc AddBlankRowToTop( var StringGrid : TStringGrid);
20157: Proc DeleteSelectedRow( var StringGrid : TStringGrid);
20158: Func StringGridSearch(const StringGrid:TStringGrid;const column:Int;const target:STR):Int;
20159: Func XLeft( rect : TRect; canvas : TCanvas; s :Str) : Int;
20160: Func XCenter( rect : TRect; canvas : TCanvas; s :Str) : Int;
20161: Func XRight( rect : TRect; canvas : TCanvas; s :Str) : Int;
20162: Func YCenter( rect : TRect; canvas : TCanvas; s :Str) : Int;
20163: end;
20164:
20165: Proc SIRegister_InetUtils2(CL: TPSPascalCompiler);
20166: begin
20167: Proc AdjustArray(var DWArray:array of DWord;const Delta:Int;MinValueToAdjust:DWord);
20168: Func GetDroppedFileNames( const DropID : Int) : TWideStringArray;
20169: Func AreBytesEqual( const First, Second : array of Byte) :Bool;;
20170: Func AreBytesEqual1( const First, Second, Length : DWord) :Bool;;
20171: Func MaskForBytes( const NumberOfBytes : Byte) : DWord;
20172: Func IntToBinByte( Int : Byte) :Str;;
20173: Func IntToBinWord( Int : Word) :Str;;
20174: Func IntToBinDWord( Int : DWord; Digits : Byte; SpaceEach : Byte) :Str;;
20175: Func WriteWS( const Stream : TStream; const Str : WideString) : Word;
20176: Proc WriteArray( const Stream : TStream; const WSArrary : array of WideString);;
20177: Proc WriteArray6( const Stream : TStream; const DWArray : array of DWord);;
20178: Func ReadWS( const Stream : TStream) : WideString;;
20179: Func ReadWS8( const Stream : TStream; out Len : Word) : WideString;;
20180: Proc ReadArray( const Stream : TStream; var WSArrary : array of WideString);;
20181: Proc ReadArray10( const Stream : TStream; var DWArray : array of DWord);;
20182: Func ParamStrW( Index : Int) : WideString;
20183: Func ParamStrFrom( CmdLine : WideString; Index : Int) : WideString;
20184: Func ParamStrEx( CmdLine : WideString; Index : Int; out Pos : Int) : WideString;
20185: Proc FindMask( Mask : WideString; Result : TStringsW);
20186: Proc FindAll( BasePath, Mask : WideString; Result : TStringsW);
20187: Proc FindAllRelative( BasePath, Mask : WideString; Result : TStringsW);
20188: Func IsInvalidPathChar( const Char : WideChar) :Bool;
20189: Func MakeValidFileNameW(const Str:WideString;const SubstitutionChar:WideChar):WideString;
20190: Func ExtractFilePathW( FileName : WideString) : WideString;
20191: Func ExtractFileNameW( Path : WideString) : WideString;
20192: Func ExpandFileNameW( FileName : WideString) : WideString;;
20193: Func ExpandFileName12( FileName, BasePath : WideString) : WideString;;
20194: Func CurrentDirectory : WideString;
20195: Func ChDirW( const ToPath : WideString) :Bool;
20196: Func ExtractFileExtW( FileName : WideString) : WideString;
20197: Func ChangeFileExtW2( FileName, Extension : WideString) : WideString;
20198: Func IncludeTrailingBackslashW( Path : WideString) : WideString;
20199: Func ExcludeTrailingBackslashW( Path : WideString) : WideString;
20200: Func IncludeTrailingPathDelimiterW( Path : WideString) : WideString;
20201: Func ExcludeTrailingPathDelimiterW( Path : WideString) : WideString;
20202: Func IncludeLeadingPathDelimiter( Path : WideString) : WideString;
20203: Func ExcludeLeadingPathDelimiter( Path : WideString) : WideString;
20204: Func FileInfo( Path : WideString) : TWin32FindData;
20205: Func IsDirectoryW( Path : WideString) :Bool;
20206: Func FileAgeW( const FileName : WideString) : Int;
20207: Func FileExistsW( Path : WideString) :Bool;
20208: Func FileSizeW( Path : WideString) : DWord;
20209: Func FileSize64( Path : WideString) : Int64;
20210: Func DeleteFileW( Path : WideString) :Bool;
20211: Func CopyDirectoryW( Source, Destination : WideString) :Bool;
20212: Func RemoveDirectoryW( Path : WideString) :Bool;
20213: Func ForceDirectoriesW( Path : WideString) :Bool;
20214: Func MkdirW( Path : WideString) :Bool;
20215: Func GetEnvironmentVariableW( Name : WideString) : WideString;
20216: Func ResolveEnvVars(Path:WideString; Callback:TEnvVarResolver;Unescape:Boolean):WideString;
20217: Func ResolveEnvVars14( Path : WideString; Unescape :Bool) : WideString;;
20218: Func ReadRegValue( Root : DWord; const Path, Key : WideString) : WideString;
20219: Func FadeSettings(Form:TForm;Step:ShortInt;Callback:TNotifyEvent):TFormFadeSettings;
20220: Func FadeSettings17(Form:TForm;MinAlpha,MaxAlpha:Byte;Step:ShortInt):TFormFadeSettings
20221: Func GetSpecialFolderID( const Path :Str) : Int;
20222: Func GetSpecialFolderPath2( FolderID : Int) :Str;
20223: end;
20224:
20225: Proc SIRegister_KFunctions(CL: TPSPascalCompiler);
20226: begin
20227: SHFolderDll,'String').SetString( 'SHFolder.dll;
20228: ConstantN('KM_BASE','LongInt').SetInt( LM_USER + 1024);
20229: ConstantN('KM_LATEUPDATE','LongInt').SetInt( KM_BASE + 1);
20230: crHResize','LongInt').SetInt( TCursor ( 101 ));
20231: ConstantN('crVResize','LongInt').SetInt( TCursor ( 102 ));
20232: crDragHandFree','LongInt').SetInt( TCursor ( 103 ));
20233: crDragHandGrip','LongInt').SetInt( TCursor ( 104 ));
20234: cCheckBoxFrameSize','LongInt').SetInt( 13);
20235: cCR','Char).SetString( #13);
20236: cLF','Char).SetString( #10);
20237: cTAB','Char).SetString( #9);

```



```

20238: cSPACE', 'Char').SetString( #32);
20239: cNULL', 'Char').SetString( #0);
20240: {cWordBreaks', 'LongInt').Value.ts32 := ord(cNULL) or ord(cTAB) or ord(cSPACE);
20241: cLineBreaks', 'LongInt').Value.ts32 := ord(cCR) or ord(cLF);
20242: cEllipsis', 'String').SetString( '...;
20243: cEOL', '').SetString( cLF);
20244: cFirstEOL', '').SetString( cLF);
20245: cFirstEOL', '').SetString( cCR); }
20246: TypeS('TLMMessage', 'TMessage;
20247: TypeS('TKkString', 'WideString;
20248: TypeS('TKkChar', 'char;
20249: //TKChar = TUTF8Char;
20250: TypeS(LONG_PTR2', 'Longint;
20251: TypeS(TKSysCharSet', 'set of AnsiChar;
20252: TypeS(TKCurrencyFormat', 'record CurrencyFormat : Byte; CurrencyDecima'
20253: +ls : Byte; CurrencyString : TKkString; DecimalSep : Char; ThousandSep : Cha'
20254: +r; UseThousandSep : Bool; end;
20255: {TypeS('TKAppContext', 'record Application : TApplication; Screen : TScr'
20256: +'een; GlobalNameSpace : IReadWriteSync; MainThreadID : LongWord; IntConstLi'
20257: +st : TThreadList; WidgetSet : TWidgetSet; DragManager : TDragManager; end; }
20258: // TypeS('PKAppContext', '^TKAppContext // will not work;
20259: //TypeS('TKClipboardFormat', 'TClipboardFormat;
20260: TypeS('TKClipboardFormat', 'Word;
20261: TypeS('TKCellSpan', 'record ColSpan : Int; RowSpan : Int; end;
20262: DelphiFunction('Func AdjustDecimalSeparator( const S :Str) :Str;
20263: Func AnsistrToString( const Text : Ansistr; CodePage :Card) : TKkString;
20264: //Func BinarySearch(AData:str; ACount:Int;KeyPtr:str;ACompareProc:TBSCompareProc;ASortedDown: Bool):Int;
20265: Proc CallTrackMouseEvent( Control : TWinControl; var Status :Bool);
20266: Proc CenterWindowInWindow( CenteredWnd, BoundWnd : HWnd);
20267: Proc CenterWindowOnScreen( CenteredWnd : HWnd);
20268: Func CharInSetEx( AChar : AnsiChar; const ASet : TKSysCharSet) :Bool;;
20269: Func CharInSetEx1( AChar : WideChar; const ASet : TKSysCharSet) :Bool;;
20270: Func ClipboardLoadStreamAs(const AFormat:str;AStream:TStream;var AText:TKkString):Bool;
20271: Func ClipboardSaveStreamAs(const AFormat:str;AStream:TStream;const AText:TKkString):Bool;
20272: Func CompareInts(I1,I2: Int):Int;//Func CompareWideChars(W1,W2:PWideChar;Locale:Card):Int;
20273: // Func CompareChars( S1, S2 : PChar; Locale :Card) : Int;
20274: Func CompareWideStrings( W1, W2 : WideString; Locale :Card) : Int;
20275: // Func CompareStrings( S1, S2 :Str; Locale :Card) : Int;
20276: Proc ConvertTabsToSpaces( var AText : TKkString; ASpacesForTab : Int);
20277: Func CreateMultipleDir( const Dir :Str) :Bool;
20278: Func DigitToNibble( Digit : AnsiChar; var Nibble : Byte) :Bool;
20279: Func DivUp( Dividend, Divisor : Int) : Int;
20280: Func DivDown( Dividend, Divisor : Int) : Int;
20281: Func EditIsFocused( AMustAllowWrite :Bool) :Bool;
20282: Func EditFocusedTextCanCopy :Bool;
20283: Func EditFocusedTextCanCut :Bool;
20284: Func EditFocusedTextCanDelete :Bool;
20285: Func EditFocusedTextCanPaste :Bool;
20286: Func EditFocusedTextCanUndo :Bool;
20287: Proc EditUndoFocused;
20288: Proc EditDeleteFocused;
20289: Proc EditCutFocused;
20290: Proc EditCopyFocused;
20291: Proc EditPasteFocused;
20292: Proc EditSelectAllFocused;
20293: Proc EnableControls2( AParent : TWinControl; AEnabled, ARecursive :Bool);
20294: Proc EnsureLastPathSlash( var APath :Str);
20295: Proc Error2( const Msg :Str);
20296: Func FillMessage( Msg :Card; WParam : WPARAM; LParam : LPARAM) : TLMMessage;
20297: Func FormatCurrency( Value : Currency; const AFormat : TKCurrencyFormat) : TKkString;
20298: //Func GetAppContext( var Ctx : TKAppContext) :Bool;
20299: Func GetAppVersion(const ALibName:str;var MajorVersion,MinorVersion,BuildNumber,RevisionNumber:Word):Bool;
20300: Func GetCharCount( const AText : TKkString; AChar : TKkChar) : Int;
20301: Func GetControlText( Value : TWinControl) : TKkString;
20302: Func GetFormatSettings2 : TFormatSettings;
20303: Func GetShiftState : TShiftState;
20304: Func IntToAscii( Value : Int64; Digits : Int) :Str;
20305: Func IntToBinStr( Value : Int64; Digits : Byte; const Suffix :Str) :Str;
20306: Func IntToBCD( Value :Card) :Card;
20307: Func IntToDecStr( Value : Int64) :Str;
20308: Func IntToHexStr(Value:Int64;Digits:Byte;const Prefix,Suffix:str;UseLowerCase:Bool):str;
20309: Func IntToOctStr( Value : Int64) :Str;
20310: Func IntToRoman2( Value : Int; AUpperCase :Bool) :Str;
20311: Func IntToLatin( Value : Int; AUpperCase :Bool) :Str;
20312: Func IntPowerInt( Value : Int64; Exponent : Int) : Int64;
20313: Func AsciiToInt( S :Str; Digits : Int) : Int64;
20314: Func BCDToInt( Value :Card) :Card;
20315: Func BinStrToInt2(S:str; Digits:Byte; Signed :Bool; var Code : Int) : Int64;
20316: Func DecStrToInt( S :Str; var Code : Int) : Int64;
20317: Func HexStrToInt(S:Str; Digits: Byte; Signed:Boolean; var Code : Int) : Int64;
20318: Func OctStrToInt( S :Str; var Code : Int) : Int64;
20319: Func KFormat14(const Format:str;const Args:array of const;const AFormatSettings:TFormatSettings):str;
20320: Func KFormat15(const Format:WideString;const Args:array of const;const AFormatSettings:TFormatSettings):
WideString;
20321: Func MakeCellSpan( AColumns, ARows : Int) : TKCellSpan;
20322: Func MinMax16(Value,Min,Max: ShortInt) : ShortInt;;
20323: Func MinMax17(Value,Min,Max: SmallInt) : SmallInt;;
20324: Func MinMax18(Value,Min,Max: Int) : Int;;
20325: Func MinMax19(Value,Min,Max: Int64) : Int64;;

```

```

20326: Func MinMax20(Value,Min,Max: Single) : Single;;
20327: Func MinMax21(Value,Min,Max: Double) : Double;;
20328: Func MinMax22(Value,Min,Max: Extended) : Extended;;
20329: Func NibbleToDigit( Nibble : Byte; UpperCase :Bool) : AnsiChar;
20330: Proc OpenURLWithShell( const AText : TKkString);
20331: //Proc QuickSortNR(AData: Pointer;ACount:Int; ACompareProc:TQsCompareProc;AExchangeProc:TQsExchangeProc;
ASortedDown:Bool);
20332: //Proc QuickSort(AData:Pointer;ACount:Int;ACompareProc:TQsCompareProc;AExchangeProc:TQsExchangeProc;
ASortedDown:Bool);
20333: Proc OffsetPoint23( var APoint : TPoint; AX, AY : Int);;
20334: Proc OffsetPoint24( var APoint : TPoint; const AOffset : TPoint);;
20335: Func RectInRect( Bounds, Rect : TRect) :Bool;
20336: Proc OffsetRect25( var ARect : TRect; AX, AY : Int);;
20337: Proc OffsetRect26( var ARect : TRect; const AOffset : TPoint);;
20338: //Func SetAppContext( const Ctx : TKAppContext) :Bool;
20339: Proc SetControlClipRect( AControl : TWinControl; const ARect : TRect);
20340: Proc SetControlText( Value : TWinControl; const Text : TKkString);
20341: Proc StripLastPathSlash( var APath :Str);
20342: Func StrNextCharIndex( const AText : TKkString; Index : Int) : Int;
20343: Func StrPreviousCharIndex( const AText : TKkString; Index : Int) : Int;
20344: Func StringCharBegin( const AText : TKkString; Index : Int) : Int;
20345: Func StringLength( const AText : TKkString) : Int;
20346: Func StringCopy( const ASource : TKkString; At, Count : Int) : TKkString;
20347: Proc StringDelete( var ASource : TKkString; At, Count : Int);
20348: Proc TrimWhiteSpaces27( const AText:TKkString;var AStart,ALen:Int;const ASet:TKSysCharSet);
20349: Proc TrimWhiteSpaces28( var AText : TKkString; const ASet : TKSysCharSet);;
20350: Proc TrimWhiteSpaces29( var AText : Ansistr; const ASet : TKSysCharSet);;
20351: Func StringToAnsistr( const AText : TKkString; CodePage :Card) : Ansistr;
20352: Func StringToChar( const AText : TKkString; AIndex : Int) : TKkChar;
20353: Func GetWindowsFolder2( CSIDL :Card; var APath :Str) :Bool;
20354: Func RunExecutable( const AFileName :Str; AWaitForIt :Bool) : DWORD;
20355: Func SystemCodePage : Int;
20356: Func NativeUTFToUnicode( const AText : TKkString) : WideChar;
20357: Func UnicodeUpperCase2( const AText : TKkString) : TKkString;
20358: Func UnicodeLowerCase2( const AText : TKkString) : TKkString;
20359: Func UnicodeToNativeUTF( const AParam : WideChar) : TKkString;
20360: Func UnicodeStringReplace( const AText,AOldPattern,ANewPattern:TKkString;AFlags:TReplaceFlags) :TKkString;
20361: end;
20362:
20363: Proc SIRegister_KMessageBox(CL: TPSPascalCompiler);
20364: begin
20365: TypeS('TKMsgBoxButton', (mbYes,mbNo,mbOk,mbCancel,mbClose,mbAbort,mbRetry,mbIgnorembAllmbNoToAll,mbYesToAll,
mbHelp);
20366: TypeS('TKMsgBoxIcon', ( miNone, miInformation, miQuestion, miWarning, miStop );
20367: Func CreateMsgBox(const Caption,Text:str;const Buttons:array of TKMsgBoxButton;Icon:TKMsgBoxIcon;Def
Int):TCustomForm;
20368: Func CreateMsgBoxEx(const Caption,Text:str;const Btns:array of
string;Icon:TKMsgBoxIcon;Def:Int):TCustomForm;
20369: Proc FreeMsgBox( AMsgBox : TCustomForm);
20370: Func KMsgBox(const Caption,Text:str;const Buttons:array of TKMsgBoxButton;Icon:TKMsgBoxIcon;Def:Int):Int;
20371: Func KMsgBoxEx(const Caption,Text:str;const Buttons:array of string;Icon:TKMsgBoxIcon;Def:Int):Int;
20372: Func KInputDialog( const Caption, Prompt :Str; var Text :Str) : TModalResult;
20373: //Func KNumberInputDialog(const ACaption,APrompt:str;var AValue:double;AMin,AMax:double;
AFormats:TKNumberEditAcceptedFormats) :TModalResult;
20374: TKMsgBoxButtons,'( mbAbortRetryIgnore,mbOkOnly,mbOkCancel,mbRetryCancel,mbYesNo,mbYesNoCancel);
20375: Func MsgBox2(const Caption,Txt:str;const Buttons:TKMsgBoxButtons;Icon:TKMsgBoxIcon):Int;
20376: Func AppMsgBox(const Caption, Text :Str; Flags: Int) : Int;
20377: end;
20378:
20379: Proc SIRegister_Kronos(CL: TPSPascalCompiler);
20380: begin
20381: 'ChurchDayCount','LongInt').SetInt( 21);
20382: ConstantN('CommonDayCount','LongInt').SetInt( 4);
20383: 'chAdvent1','LongInt').SetInt( 1);
20384: 'chAdvent2','LongInt').SetInt( 2);
20385: 'chAdvent3','LongInt').SetInt( 3);
20386: 'chAdvent4','LongInt').SetInt( 4);
20387: 'chChristmasEve','LongInt').SetInt( 5);
20388: 'chChristmasDay','LongInt').SetInt( 6);
20389: 'chBoxingDay','LongInt').SetInt( 7);
20390: 'chNewYearEve','LongInt').SetInt( 8);
20391: 'chNewYearDay','LongInt').SetInt( 9);
20392: 'chShroveTuesday','LongInt').SetInt( 10);
20393: 'chAshWednesday','LongInt').SetInt( 11);
20394: 'chPalmSunday','LongInt').SetInt( 12);
20395: 'chMaundyThursday','LongInt').SetInt( 13);
20396: 'chGoodFriday','LongInt').SetInt( 14);
20397: 'chEasterEve','LongInt').SetInt( 15);
20398: 'chEasterSunday','LongInt').SetInt( 16);
20399: 'chEasterMonday','LongInt').SetInt( 17);
20400: 'chWhitEve','LongInt').SetInt( 18);
20401: 'chWhitSunday','LongInt').SetInt( 19);
20402: 'chWhitMonday','LongInt').SetInt( 20);
20403: 'chAscensionDay','LongInt').SetInt( 21);
20404: 'coUNDay','LongInt').SetInt( 22);
20405: 'coWomensDay','LongInt').SetInt( 23);
20406: 'coMayDay','LongInt').SetInt( 24);
20407: 'coLiteracyDay','LongInt').SetInt( 25);
20408: 'UserDayType','LongInt').SetInt( 26);

```

```

20409: //UserDayType = ChurchDayCount + CommonDayCount + 1;
20410: TFirstLastNumber', 'array[1..2] of word;;
20411: TDaytypeID', 'array[1..255] of word;;
20412: TMonthImage2', 'array[0..7] of smallint;;
20413: TMonthImage', 'array[1..6] of tmonthimage2;;
20414: //TMonthImage = array[1..6, 0..7] of smallint;
20415: // TFirstLastNumber = array[1..2] of word;
20416: // TDaytypeID = array[1..255] of word;
20417: TDay', 'record Daynum : Word; MonthDate : word; DOWNum : word; M'
20418: + 'onth : word; Week : word; DayCode : Word; end;
20419: TypeS('TWeek', 'record WeekNum : word; WhichDays : TFirstLastNumber; end;
20420: TMonth', 'record Month : word; Daycount : Word; WeekCount : Word'
20421: + ' ; WhichWeeks : TFirstLastNumber; WhichDays : TFirstLastNumber; end;
20422: TYear', 'record WeekCount : word; DayCount : Word; end;
20423: TKron', 'record ActiveYear : Word; IsInitialized :Bool; end;
20424: TYearExt', 'record Year : word; NumDays : word; NumWeeks : word;'
20425: + LeapYear :Bool; YeartypeCount : word; end;
20426: TDateExt', 'record Year : word; DayOfWeekNumber : word; DayName '
20427: +Str; MonthDay : Word; DayNumber : word; DaytypeCount : word; DaytypeI'
20428: +D : TDaytypeID; MonthNumber : word; WeekNumber : word; Holiday : bool; '
20429: +ChurchDay : Bool; Flagday :Bool; end;
20430: TMonthExt', 'record Year : word; MonthNumber : word; MonthName : '
20431: + ' string; FirstDay : word; LastDay : word; NumDays : word; NumWeeks : word;'
20432: + ' FirstWeek : word; LastWeek : word; MonthImage : TMonthImage; end;
20433: TWeekExt', 'record Year : word; WeekNumber : word; FirstDay : word; LastDay : word; end;
20434: TForeignKey', 'record KeyName :Str; KeyValue : Variant; end;
20435: TDaytypeDef', 'record AName :Str; ADate : word; ARelDayTyp'
20436: +e : word; AnOffset : Int; AFirstShowUp : word; ALastShowUp : word; ASH'
20437: +owUpFrequency : word; AChurchDay :Bool; AHoliday :Bool; AFlagday : '
20438: + bool; AUserCalc :Bool; ATag : Int; end;
20439: SIRegister_TDaytype(CL);
20440: TWeekDay', '( Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday );
20441: TWeekHolidays', 'set of TWeekDay;
20442: TOcEvent', '( ocYear, ocMonth, ocMonthnumber, ocWeek, ocWeeknumb'
20443: + 'er, ocMonthDay, ocWeekday, ocDate, ocToday, ocCalcDaytype );
20444: TCalcDaytypeEvent', 'Proc ( Sender : TObject; Daytype : TDa'
20445: + 'ytype; ADateExt : TDateExt; IsCurrentDate :Bool; var Accept :Bool);
20446: TLoadDaytypeEvent', 'Proc ( Sender : TObject; const Daytype'
20447: + 'Def : TDaytypeDef; const DescKeys :Str; ClassId : Int; var LoadIt: bool);
20448: TSaveDaytypeEvent', 'Proc ( Sender : TObject; Daytype : TDa'
20449: + 'ytype; var DescKeys :Str; var ClassID : Int; var SaveIt :Bool);
20450: SIRegister_TKronos(CL);
20451: ClassN(CL.FindClass('TOBJECT'), 'EKronosError;
20452: //DelphiFunction('Proc Register;
20453: end;
20454:
20455: Proc SIRegister_MapFiles(CL: TPSPascalCompiler);
20456: begin
20457:   ClassN(CL.FindClass('TOBJECT'), 'EMemoryMappedFile;
20458:   ClassN(CL.FindClass('TOBJECT'), 'EMMEndOfFile;
20459:   TypeS('TFileOptions, (foCreateNew, foCreateAlways, foOpenExisting, foOpenAlways, foTruncateExisting);
20460:   TypeS('TmapFileType', ( ftUnspecified, ftRandomAccess, ftSequentialAccess);
20461:   SIRegister_TMapStream(CL);
20462:   SIRegister_TTextMap(CL);
20463:   SIRegister_TRecordMap(CL);
20464: end;
20465:
20466: Proc SIRegister_KGraphics(CL: TPSPascalCompiler);
20467: begin
20468:   PNGHeader', 'String').SetString(' #137 'PNG' #13#10#26#10;
20469:   MNGHeader', 'String').SetString(' #138 'MNG' #13#10#26#10;
20470:   TKBrightMode', '( bsAbsolute, bsOfBottom, bsOfTop );
20471:   TKColorRec', 'record R : byte; G : byte; B : byte; A : Byte; Value :Card; end;
20472:   //PKColorRec', '^TKColorRec // will not work;
20473:   TKDynColorRecs', 'array of TKColorRec;
20474:   TKImageHeaderString', 'string;
20475:   //TKPngImage', 'TPortableNetworkGraphic;
20476:   with ClassN(CL.FindClass('TLinearGraphic'), 'TPNGGraphic') do
20477:     TKPngImage', 'TPNGGraphic;
20478:     TKKString', 'WideString;
20479:     // TKString = WideString;
20480:     //TKPngImage', 'TPngImage;
20481:     //TKPngImage', 'TPngObject;
20482:     TKTextAttribute', '( taCalcRect, taClip, taEndEllipsis, taFillRe'
20483:     + 'ct, taFillText, taIncludePadding, taLineBreak, taPathEllipsis, taWordBreak'
20484:     + ' , taWrapText, taTrimWhiteSpaces, taStartEllipsis );
20485:     TKTextAttributes', 'set of TKTextAttribute;
20486:     TKHAlign', '( halLeft, halCenter, halRight, halJustify );
20487:     TKStretchMode', '( stmNone, stmZoomOutOnly, stmZoomInOnly, stmZoom );
20488:     TKTextHAlign', 'TKHAlign;
20489:     TKVAlign', '( valTop, valCenter, valBottom );
20490:     TKTextVAlign', 'TKVAlign;
20491:     TKButtonDrawState', '( bsUseThemes, bsDisabled, bsPressed, bsFocused, bsHot );
20492:     TKButtonDrawStates', 'set of TKButtonDrawState;
20493:     SIRegister_TKGraphic(CL);
20494:     SIRegister_TKAlphaBitmap(CL);
20495:     SIRegister_TKMetafile(CL);
20496:     TKTextBoxFunction', '( tbfMeasure, tbfGetIndex, tbfGetRect, tbfDraw );
20497:     SIRegister_TKTextBox(CL);

```

```

20498: SIRegister_TKDragWindow(CL);
20499: SIRegister_TKHintWindow(CL);
20500: SIRegister_TKTextHint(CL);
20501: SIRegister_TKGraphicHint(CL);
20502: //DelphiFunction('Proc BlendLine( Src, Dest : PKColorRecs; Count : Int);
20503: Func TKBrightColor( Color : TColor; Percent : Single; Mode : TKBrightMode): TColor;
20504: Proc CanvasGetScale( ACanvas : TCanvas; out MulX, MulY, DivX, DivY : Int);
20505: Proc CanvasResetScale( ACanvas : TCanvas);
20506: Func CanvasScaled( ACanvas : TCanvas) :Bool;
20507: Proc CanvasSetScale( ACanvas : TCanvas; MulX, MulY, DivX, DivY : Int);
20508: Proc CanvasSetOffset( ACanvas : TCanvas; OfX, OfY : Int);
20509: Func ColorRecToColor( Color : TKColorRec) : TColor;
20510: Func ColorToColorRec( Color : TColor) : TKColorRec;
20511: Func TKColorToGrayScale( Color : TColor) : TColor;
20512: Func CompareBrushes( ABrush1, ABrush2 : TBrush) :Bool;
20513: Func CompareFonts( AFont1, AFont2 : TFont) :Bool;
20514: Proc CopyBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcX, SrcY : Int);
20515: Func CreateEmptyPoint : TPoint;
20516: Func CreateEmptyRect : TRect;
20517: Func CreateEmptyRgn : HRGN;
20518: Proc DrawAlignedText(Canvas: TCanvas; var ARect : TRect; HAlign : TKHAlign; VAlign : TKVAlign; HPadding,
VPadding: Int; const AText:TKKString;BackColor:TColor;Attributes:TKTextAttributes);
20519: Proc TKDrawButtonFrame(ACanvas:TCanvas;const ARect:TRect;AStates:TKButtonDrawStates);
20520: Proc DrawEdges(Canvas:TCanvas;const R:TRect;HighlightColor,ShadowColor:TColor;Flags:Cardi;
20521: Proc DrawFilledRectangle(Canvas: TCanvas; const ARect : TRect; BackColor : TColor);
20522: Proc DrawGradientRect(Canvas:TCanvas; const ARect:TRect;AStartColor,AEndColor TColor; AColorStep:
Int;AHorizontal:Bool);
20523: Proc ExcludeShapeFromBaseRect(var BaseRect:TRect; ShapeWidth, ShapeHeight : Int; HAlign : TKHAlign;
VAlign : TKVAlign; HPadding, VPadding : Int; StretchMode : TKStretchMode; out Bounds, Interior : TRect);
20524: Func ExtSelectClipRect(DC:HDC; ARect:TRect; Mode:Int; var PrevRgn: HRGN):Bool;
20525: Func ExtSelectClipRectEx(DC:HDC;ARect:TRect;Mode:Int;CurRgn,PrevRgn : HRGN):Bool;
20526: Proc FillAroundRect(ACanvas:TCanvas;const Boundary,Interior:TRect;BackColor:TColor);
20527: Func GetFontHeight( DC : HDC) : Int;
20528: Func GetFontAscent( DC : HDC) : Int;
20529: Func GetFontDescent( DC : HDC) : Int;
20530: Func GDICheck( Value : Int) : Int;
20531: Func HorizontalShapePosition(AAlignment:TKHAlign;const ABoundary:TRect;const AShapeSize:TPoint):Int;
20532: Func ImageByType( const Header : TKImageHeaderString) : TGraphic;
20533: Func IntersectClipRectIndirect( DC : HDC; ARect : TRect) :Bool;
20534: Func IsBrightColor( Color : TColor) :Bool;
20535: Proc LoadCustomCursor( Cursor : TCursor; const ResName :Str);
20536: Proc TKLoadGraphicFromResource(Graphic:TGraphic;const ResName:str;ResType:PChar);
20537: Func MakeColorRec( R, G, B, A : Byte) : TKColorRec;
20538: Func MakeColorRec5( Value : LongWord) : TKColorRec;
20539: Func PixelFormatFromBpp( Bpp :Card) : TPixelFormat;
20540: Func TKRectInRegion( Rgn : HRGN; ARect : TRect) :Bool;
20541: Func RgnCreateAndGet( DC : HDC) : HRGN;
20542: Proc RgnSelectAndDelete( DC : HDC; Rgn : HRGN);
20543: Proc RoundRectangle(ACanvas: TCanvas; const ARect: TRect;AXRadius,AYRadius: Int);
20544: Proc SafeStretchDraw(ACanvas:TCanvas; ARect:TRect;AGraphic:TGraphic;ABackColor:TColor);
20545: Proc SelectClipRect( DC : HDC; const ARect : TRect);
20546: Proc TKStretchBitmap( DestDC : HDC; DestRect : TRect; SrcDC : HDC; SrcRect : TRect);
20547: Func SwitchRGBToBGR( Value : TColor) : TColor;
20548: Proc TranslateRectToDevice( DC : HDC; var ARect : TRect);
20549: Func VerticalShapePosition(AAlignment:TKVAlign;const ABoundary:TRect;const AShapeSize: TPoint):Int;
20550: end;
20551:
20552: Proc SIRegister_umaxPipes(CL: TPSPascalCompiler);
20553: begin
20554: ConstantN('cShutdownMsg','String').SetString( 'shutdown pipe ;
20555: ConstantN(cPipeFormat',String).SetString( '\\%s\pipe%s;
20556: SIRegister_TPipeServermax(CL);
20557: SIRegister_TPipeClientmax(CL);
20558: end;
20559:
20560: Proc SIRegister_KControls(CL: TPSPascalCompiler);
20561: begin
20562: TypeS('TKColorArray', 'array of TColor;
20563: TKPreviewColorIndex', 'Int;
20564: TKPrintOption', '( poCollate, poFitToPage, poMirrorMargins, poPa'
20565: + 'geNumbers, poPaintSelection, poTitle, poUseColor );
20566: TKPrintOptions', 'set of TKPrintOption;
20567: TKPrintRange', '( kprAll, kprSelectedOnly, kprRange );
20568: TKPrintUnits', '( kpuMM, kpuCM, kpuInch, kpuHundredthInch );
20569: //ConstantN('cBorderStyleDef','').SetString( bsSingle);
20570: 'cContentPaddingBottomDef','LongInt').SetInt( 0);
20571: 'cContentPaddingLeftDef','LongInt').SetInt( 0);
20572: ConstantN(cOptionsDef', 'LongInt').Value.ts32:= ord(poFitToPage) or ord(poPageNumbers) or ord(poUseColor);
20573: //ConstantN('cRangeDef','').SetString( prAll);
20574: 'cScaleDef','LongInt').SetInt( 100);
20575: 'cScaleMin','LongInt').SetInt( 10);
20576: 'cScaleMax','LongInt').SetInt( 500);
20577: //ConstantN('cUnitsDef','').SetString( puCM);
20578: //ConstantN('cPaperDef','').SetString( clWhite);
20579: //ConstantN('cBkGndDef','').SetString( clAppWorkSpace);
20580: //ConstantN('cBorderDef','').SetString( clBlack);
20581: //ConstantN('cSelectedBorderDef','').SetString( clNavy);
20582: ConstantN('ciPaper','LongInt').SetInt( TKPreviewColorIndex ( 0 ));
20583: 'ciBkGnd','LongInt').SetInt( TKPreviewColorIndex ( 1 ));

```

```

20584: 'ciBorder','LongInt').SetInt( TKPreviewColorIndex ( 2 ));
20585: 'ciSelectedBorder','LongInt').SetInt( TKPreviewColorIndex ( 3 ));
20586: // 'ciPreviewColorsMax','').SetString( ciSelectedBorder);
20587: 'cScrollNoAction','LongInt').SetInt( - 1);
20588: 'cScrollDelta','LongInt').SetInt( - 2);
20589: 'cPF_Dragging','LongWord').SetUInt( $00000001);
20590: 'cPF_UpdateRange','LongWord').SetUInt( $00000002);
20591: TKPreviewScaleMode', '( smScale, smPageWidth, smWholePage );
20592: TKPreviewChangedEvent', 'Proc ( Sender : TObject);
20593: TKPrintMeasureInfo', 'record OutlineWidth : Int; OutlineHeig'
20594: ht : Int; ControlHorzPageCount : Int; ControlVertPageCount : Integ'
20595: er; ExtraLeftHorzPageCount : Int; ExtraLeftVertPageCount : Int; Ex'
20596: traRightHorzPageCount : Int; ExtraRightVertPageCount : Int; end;
20597: TKPrintStatus', '( kpsBegin, kpsNewPage, kpsEnd );
20598: TKPrintNotifyEvent, Procedure (Sender:TObject;Status: TKPrintStatus;var Abort:Bool);
20599: TKPrintPaintEvent', 'Proc ( Sender : TObject);
20600: ClassN (CL.FindClass ('TObject'),TKPrintPageSetup;
20601: ClassN (CL.FindClass ('TObject'),TKPrintPreview;
20602: SIRegister_TKRect (CL);
20603: ClassN (CL.FindClass ('TObject'),'TKObjectList;
20604: SIRegister_TKObject (CL);
20605: //TKObjectClass', 'class of TKObject;
20606: SIRegister_TKObjectList (CL);
20607: SIRegister_TKCustomControl (CL);
20608: TKColorScheme', '( kcsNormal, kcsGrayed, kcsBright, kcsGrayScale );
20609: TKColorIndex', 'Int;
20610: TKColorData', 'record Index:TKColorIndex;Color:TColor; Default:TColor; Name:str; end;
20611: TKColorSpec', 'record Def : TColor; Name :Str; end;
20612: SIRegister_TKCustomColors (CL);
20613: TKPrintMeasureEvent', 'Proc ( Sender : TObject; var Info : TKPrintMeasureInfo);
20614: SIRegister_TKPrintPageSetup (CL);
20615: SIRegister_TKPreviewColors (CL);
20616: SIRegister_TKPrintPreview (CL);
20617: Func InchesToValue (Units : TKPrintUnits; Value : Double) : Double;
20618: Func ValueToInches (Units : TKPrintUnits; Value : Double) : Double;
20619: end;
20620:
20621: Proc SIRegister_IdAntiFreezeBase (CL: TPSPascalCompiler);
20622: begin
20623: ConstantN (ID_Default_TIdAntiFreezeBase_Active,Boolean).SetInt (1);
20624: ConstantN ('ID_Default_TIdAntiFreezeBase_ApplicationHasPriority','Boolean').SetInt (1);
20625: ConstantN ('ID_Default_TIdAntiFreezeBase_IdleTimeOut','LongInt').SetInt ( 250);
20626: ConstantN ('ID_Default_TIdAntiFreezeBase_OnlyWhenIdle','Boolean').SetInt ( 250);
20627: SIRegister_TIdAntiFreezeBase (CL);
20628: end;
20629:
20630: Proc SIRegister_OverbyteIcsConApp (CL: TPSPascalCompiler);
20631: begin
20632: ConstantN ('WM_STARTUP','LongInt').SetInt ( WM_USER + 789);
20633: SIRegister_TKeyboardThread (CL);
20634: //Types ('TConApplicationClass', 'class of TConApplication;
20635: SIRegister_TConApplication (CL);
20636: end;
20637:
20638: Proc SIRegister_KMemo (CL: TPSPascalCompiler);
20639: begin
20640: 'cUndoLimitMin','LongInt').SetInt ( 100);
20641: cUndoLimitMax','LongInt').SetInt ( 10000);
20642: cUndoLimitDef','LongInt').SetInt ( 1000);
20643: cScrollPaddingMin','LongInt').SetInt ( 0);
20644: cScrollPaddingMax','LongInt').SetInt ( 1000);
20645: cScrollPaddingDef','LongInt').SetInt ( 30);
20646: cScrollSpeedMin','LongInt').SetInt ( 50);
20647: cScrollSpeedMax','LongInt').SetInt ( 1000);
20648: cScrollSpeedDef','LongInt').SetInt ( 100);
20649: //cBkGndDef','').SetString( clWindow);
20650: //cInactiveCaretBkGndDef','').SetString( clBlack);
20651: //cInactiveCaretSelBkGndDef','').SetString( clBlack);
20652: //cInactiveCaretSelTextDef','').SetString( clYellow);
20653: //cInactiveCaretTextDef','').SetString( clYellow);
20654: //cSelBkGndDef','').SetString( clGrayText);
20655: //cSelBkGndFocusedDef','').SetString( clHighlight);
20656: //cSelTextDef','').SetString( clHighlightText);
20657: //cSelTextFocusedDef','').SetString( clHighlightText);
20658: ciBkGnd','LongInt').SetInt( TKColorIndex ( 0 ));
20659: ciInactiveCaretBkGnd','LongInt').SetInt( TKColorIndex ( 1 ));
20660: ciInactiveCaretSelBkGnd','LongInt').SetInt( TKColorIndex ( 2 ));
20661: ciInactiveCaretSelText','LongInt').SetInt( TKColorIndex ( 3 ));
20662: ciInactiveCaretText','LongInt').SetInt( TKColorIndex ( 4 ));
20663: ciSelBkGnd','LongInt').SetInt( TKColorIndex ( 5 ));
20664: ciSelBkGndFocused','LongInt').SetInt( TKColorIndex ( 6 ));
20665: ciSelText','LongInt').SetInt( TKColorIndex ( 7 ));
20666: ciSelTextFocused','LongInt').SetInt( TKColorIndex ( 8 ));
20667: //ciMemoColorsMax','').SetString( ciSelTextFocused);
20668: cInvalidListID','LongInt').SetInt( - 1);
20669: cHorzScrollStepDef','LongInt').SetInt( 4);
20670: cVertScrollStepDef','LongInt').SetInt( 10);
20671: //cHeight','LongInt').SetInt( 200);
20672: //cWidth','LongInt').SetInt( 300);

```



```

20673: cNewLineChar', 'Char').SetString( #B6);
20674: cSpaceChar', 'Char').SetString( #B7);
20675: cTabChar', 'Char').SetString( #2192);
20676: cBullet', 'Char').SetString( #2022);
20677: cSquareBullet, Char').SetString( #25AB);
20678: cArrowBullet', Char').SetString( #25BA);
20679: //cDefaultWordBreaks, String').SetString(' ' or '/' or '\' or ';' or ':' or '?' or '!';
20680: cRichText', String').SetString( 'Rich Text Format;
20681: ClassN(CL.FindClass(TOBJECT'), TKCustomMemo;
20682: TypeS('TKMemoLinePosition', '( eolInside, eolEnd );
20683: TKMemoBlockPosition', '( mbpText, mbpRelative, mbpAbsolute );
20684: TKMemoState', '( elCaretCreated, elCaretVisible, elCaretUpdate, '
20685: +elIgnoreNextChar, elModified, elMouseCapture, elOverwrite, elReadOnly );
20686: TKMemoStates', 'set of TKMemoState;
20687: TKMemoUpdateReason', '( muContent, muExtent, muSelection, muSelectionScroll );
20688: TKMemoUpdateReasons', 'set of TKMemoUpdateReason;
20689: SIRegister_TKMemoSparseItem(CL);
20690: SIRegister_TKMemoSparseList(CL);
20691: SIRegister_TKMemoSparseStack(CL);
20692: SIRegister_TKMemoDictionaryItem(CL);
20693: SIRegister_TKMemoDictionary(CL);
20694: TKMemoParaNumbering', '( pnuNone, pnuBullets, pnuArabic, pnuLett'
20695: +erLo, pnuLetterHi, pnuRomanLo, pnuRomanHi );
20696: SIRegister_TKMemoNumberingFormatItem(CL);
20697: SIRegister_TKMemoNumberingFormat(CL);
20698: ClassN(CL.FindClass(TOBJECT'), TKMemoListLevels;
20699: SIRegister_TKMemoListLevel(CL);
20700: ClassN(CL.FindClass(TOBJECT'), TKMemoList;
20701: SIRegister_TKMemoListLevels(CL);
20702: ClassN(CL.FindClass(TOBJECT'), TKMemoListTable;
20703: SIRegister_TKMemoList(CL);
20704: TKMemoListChangeEvent', 'Proc (AList: TKMemoList; ALevel: TKMemoListLevel);
20705: SIRegister_TKMemoListTable(CL);
20706: TKMemoScriptCapitals', '( tcaNone, tcaNormal, tcaSmall );
20707: TKMemoScriptPosition', '( tpoNormal, tpoSuperscript, tpoSubscript );
20708: SIRegister_TKMemoTextStyle(CL);
20709: TKMemoBlockWrapMode', '( wrAround, wrAroundLeft, wrAroundRight, '
20710: +wrTight, wrTightLeft, wrTightRight, wrTopBottom, wrNone, wrUnknown );
20711: TKMemoBlockStyleChangeEvent', 'Proc ( Sender : TObject; AR'
20712: +easons : TKMemoUpdateReasons);
20713: SIRegister_TKMemoBlockStyle(CL);
20714: TKMemoLineSpacingMode', '( lsmFactor, lsmValue );
20715: SIRegister_TKMemoParaStyle(CL);
20716: SIRegister_TKMemoLine(CL);
20717: SIRegister_TKMemoLines(CL);
20718: SIRegister_TKMemoWord(CL);
20719: SIRegister_TKMemoWordList(CL);
20720: ClassN(CL.FindClass(TOBJECT'), TKMemoBlocks;
20721: TKMemoMouseAction', '( maMove, maLeftDown, maLeftUp, maRightDown'
20722: +', maRightUp, maMidDown, maMidUp );
20723: //TKMemoBlockClass', 'class of TKMemoBlock;
20724: SIRegister_TKMemoBlock(CL);
20725: SIRegister_TKMemoSingleton(CL);
20726: SIRegister_TKMemoTextBlock(CL);
20727: SIRegister_TKMemoHyperlink(CL);
20728: SIRegister_TKMemoParagraph(CL);
20729: SIRegister_TKMemoImageBlock(CL);
20730: SIRegister_TKMemoContainer(CL);
20731: ClassN(CL.FindClass(TOBJECT'), TKMemoTable;
20732: ClassN(CL.FindClass(TOBJECT'), TKMemoTableRow;
20733: SIRegister_TKMemoTableCell(CL);
20734: SIRegister_TKMemoTableRow(CL);
20735: SIRegister_TKMemoTable(CL);
20736: TKMemoUpdateEvent', 'Proc ( Reasons : TKMemoUpdateReasons);
20737: SIRegister_TKMemoBlocks(CL);
20738: SIRegister_TKMemoColors(CL);
20739: TKMemoChangeKind', '( ckCaretPos, ckDelete, ckInsert );
20740: TKMemoUndoChangeEvent', 'Procedure (Sender : TObject; ItemReason: TKMemoChangeKind);
20741: TKMemoChangeItem', 'record Blocks : TKMemoBlocks; Group : Cardin'
20742: al; GroupKind : TKMemoChangeKind; Inserted : Bool; ItemKind : TKMemoCha'
20743: ngeKind; Position : Int; end;
20744: //PKMemoChangeItem', '^TKMemoChangeItem // will not work;
20745: SIRegister_TKMemoChangeList(CL);
20746: TKMemoRTFString', 'string;
20747: SIRegister_TKCustomMemo (CL);
20748: SIRegister_TKMemo (CL);
20749: SIRegister_TKMemoEditAction (CL);
20750: SIRegister_TKMemoEditCopyAction (CL);
20751: SIRegister_TKMemoEditCutAction (CL);
20752: SIRegister_TKMemoEditPasteAction (CL);
20753: SIRegister_TKMemoEditSelectAllAction (CL);
20754: Func NewLineChar : TKkString;
20755: Func SpaceChar : TKkString;
20756: Func TabChar : TKkString;
20757: end;
20758:
20759: Proc SIRegister_OverbyteIcsTicks64 (CL: TPSPascalCompiler);
20760: begin
20761: ConstantN('ISODateMask', String).SetString( 'yyyy-mm-dd;

```

```

20762: 'ISODateTimeMask','String').SetString( 'yyyy-mm-dd"T"hh:nn:ss;
20763: 'ISODateLongTimeMask','String').SetString( 'yyyy-mm-dd"T"hh:nn:ss.zzz;
20764: 'ISOTimeMask','String').SetString( 'hh:nn:ss;
20765: 'LongTimeMask','String').SetString( 'hh:nn:ss.zzz;
20766: 'Ticks64PerDay','int64').SetInt64( 24 * 60 * 60 * 1000);
20767: 'Ticks64PerHour','int64').SetInt64( 60 * 60 * 1000);
20768: 'Ticks64PerMinute','int64').SetInt64( 60 * 1000);
20769: 'Ticks64PerSecond','int64').SetInt64( 1000);
20770: TypeS('TTicks64Mode', ( TicksNone, TicksAPI64, TicksPerf, TicksAPI32 );
20771: DelphiFunction(Func IcsGetTickCount64 : int64;
20772: Proc IcsInitTick64( NewMode : TTicks64Mode);
20773: Func IcsNowPC : TDateTime;
20774: Proc IcsAlignNowPC;
20775: Func IcsLastBootDT : TDateTime;
20776: Func IcsGetPerfCountsPerSec : int64;
20777: Func IcsPerfCountCurrent : int64;
20778: Func IcsPerfCountCurrMilli : int64;
20779: Func IcsPerfCountToMilli( LI : int64) : int64;
20780: Func IcsPerfCountGetMilli( startLI : int64) : int64;
20781: Func IcsPerfCountGetMillStr( startLI : int64) :Str;
20782: Func IcsPerfCountToSecs( LI : int64) : Int;
20783: Func IcsPerfCountGetSecs( startLI : int64) : Int;
20784: Func IcsDiffTicks64( const StartTick, EndTick : int64) : int64;
20785: Func IcsElapsedTicks64( const StartTick : int64) : int64;
20786: Func IcsElapsedMsecs64( const StartTick : int64) : int64;
20787: Func IcsElapsedSecs64( const StartTick : int64) : Int;
20788: Func IcsElapsedMins64( const StartTick : int64) : Int;
20789: Func IcsWaitingSecs64( const EndTick : int64) : Int;
20790: Func IcsGetTrgMsecs64( const MilliSecs : int64) : int64;
20791: Func IcsGetTrgSecs64( const DurSecs : Int) : int64;
20792: Func IcsGetTrgMins64( const DurMins : Int) : int64;
20793: Func IcsTestTrgTick64( const TrgTick : int64) :Bool;
20794: Func IcsAddTrgMsecs64( const TickCount, MilliSecs : int64) : int64;
20795: Func IcsAddTrgSecs64( const TickCount : int64; DurSecs : Int) : int64;
20796: end;
20797:
20798: Proc SIRegister_OverbyteIcsShal(CL: TPSPascalCompiler);
20799: begin
20800: ConstantN(IcsSHALVersion','LongInt.SetInt( 800);
20801: Copyright','String.SetString( IcsSHAL (c) 2004-2012 F. Piette V8.00 ;
20802: shaSuccess','LongInt.SetInt( 0);
20803: shaNull','LongInt.SetInt( 1);
20804: shaInputTooLong','LongInt.SetInt( 2);
20805: shaStateError','LongInt.SetInt( 3);
20806: SHA1HashSize','LongInt.SetInt( 20);
20807: TypeS('uint32_t', 'LongWord;
20808: TypeS('uint8_t', 'Byte;
20809: TypeS('int_least16_t', 'LongInt;
20810: TypeS('SHA1DigestString', 'Ansistr;
20811: Func SHA1Reset( var context : SHA1Context) : Int;
20812: Func SHA1Input( var context:SHA1Context;const message_array:str;message_array:PAnsiChar;length
Cardinal):Int;
20813: Func SHA1Result( var context : SHA1Context; var Message_Digest : SHA1Digest) : Int;
20814: Func SHA1ofStr( const s : Ansistr) : SHA1DigestString;
20815: Func SHA1ofBuf( const buf, buflen : Int) : SHA1DigestString;
20816: Func SHA1ofStream( const strm : TStream) : SHA1DigestString;
20817: Func SHA1toHex( const digest : SHA1DigestString) :Str;
20818: Func SHA1DigestToLowerHex( const Digest : SHA1Digest) :Str;
20819: Func SHA1DigestToLowerHexA( const Digest : SHA1Digest) : RawByteString;
20820: Func SHA1DigestToLowerHexW( const Digest : SHA1Digest) : UnicodeString;
20821: Proc HMAC_SHA1(const Data,DataLen:Int;const Key,KeyLen:Int;out Digest:SHA1Digest);
20822: Func HMAC_SHA1_EX( const Data : Ansistr; const Key : Ansistr) : Ansistr;
20823: end;
20824:
20825: Proc SIRegister_KEditCommon(CL: TPSPascalCompiler);
20826: begin
20827: cCharMappingSize','LongInt.SetInt( 256);
20828: TKEditCommandk', '( ecNonek, ecLeftk, ecRightk, ecUpk, ecDownk, ecLinek'
20829: Start, ecLineEndk, ecPageUpk, ecPageDownk, ecPageLeftk, ecPageRightk, ecPageTopk,
20830: htk, ecSelUpk, ecSelDownk, ecSelLineStartk, ecSelLineEndk, ecSelPageUpk, ecSelPa'
20831: geDownk, ecSelPageLeftk, ecSelPageRightk, ecSelPageTopk, ecSelPageBottomk, ecSe'
20832: lEditorTopk, ecSelEditorBottomk, ecSelGotoXYk, ecScrollUpk, ecScrollDownk, ecSc'
20833: rollLeftk, ecScrollRightk, ecScrollCenterk, ecUndok, ecRedok, ecCopyk, ecCutk, ec'
20834: +Pastek, ecInsertChark, ecInsertDigitk, ecInsertStringk, ecInsertNewLinek, ecDe'
20835: +leteLastChark, ecDeleteChark, ecDeleteBOLk, ecDeleteEOLk, ecDeleteLinek, ecSele'
20836: +ctAllk, ecClearAllk, ecClearIndexSelectionk, ecClearSelectionk, ecSearchk, ecRe'
20837: +placek, ecInsertModek, ecOverwriteModek, ecToggleModek, ecGotFocusk, ecLostFocusk ); *)
20838: TKEditDisabledDrawStyle', '( eddBright, eddGrayed, eddNormal );
20839: TKEditKey', 'record Key : Word; Shift : TShiftState; end;
20840: TKEditCommandAssignment', 'record Key : TKEditKey; Command : TKEditCommand; end;
20841: TKEditCommandMap', 'array of TKEditCommandAssignment;
20842: TKEditDropFilesEvent','Procedure(Sender:TObject;X,Y:Int;Files:TStrings);
20843: SIRegister_TKEditKeyMapping(CL);
20844: TKEditCharMapping', 'array of AnsiChar;
20845: //TypeS('PKEditCharMapping', '^TKEditCharMapping // will not work;
20846: //TypeS('TKEditOption', '( eoDropFiles, eoGroupUndo, eoUndoAfterSave, eoS'
20847: //+howFormatting, eoWantTab );
20848: //TypeS('TKEditOptions', 'set of TKEditOption;
20849: TKEditReplaceAction', '( eraCancel, eraYes, eraNo, eraAll );

```

```

20850:   TKEditReplaceTextEvent', 'Proc ( Sender : TObject; const Te'
20851:     xtToFind, TextToReplace :Str; var Action : TKEditReplaceAction);
20852:   TKEditSearchError', '( eseOk, eseNoDigitsFind, eseNoDigitsReplace, eseNoMatch );
20853:   TKEditSearchOption', '( esoAll, esoBackwards, esoEntireScope, es'
20854:     oFirstSearch, esoMatchCase, esoPrompt, esoSelectedOnly, esoTreatAsDigits, esoWereDigits );
20855:   TKEditSearchOptions', 'set of TKEditSearchOption;
20856:   TKEditSearchData', 'record ErrorReason : TKEditSearchError; Opti'
20857:     ons:TKEditSearchOptions;SelStart:Int;SelEnd:Int;TextToFind:str;TextToReplace:str; end;
20858:   //Types('PKEditSearchData', '^TKEditSearchData // will not work;
20859:   //cEditDisabledDrawStyleDef', '.SetString( eddBright);
20860:   Func CreateDefaultKeyMapping : TKEditKeyMapping;
20861:   Func DefaultCharMapping : TKEditCharMapping;
20862:   Func DefaultSearchData : TKEditSearchData;
20863: end;
20864:
20865: unit UtilsMax4; unit uPSI_UtilsMax4;
20866: Func AllDigitsDifferent(N: Int64):Bool;
20867: Proc DecimalToFraction(Decimal: Extended; out FractionNumerator: Extended;
20868:   out FractionDenominator: Extended; const AccuracyFactor: Extended);
20869: Func ColorToHTML(const Color: TColor):Str;
20870: Func DOSCommand(const CommandLine:Str; const CmdShow:Int;
20871:   const WaitUntilComplete:Bool; const WorkingDir:Str = ':Bool;
20872: Func GetDosOutput(CommandLine:Str; Work:Str = 'C:\':Str;
20873: Ex.: writeln(GetDosOutput('java -version', 'C:\'));
20874:   >>>java version "1.8.0_211"
20875:   Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
20876:   Java HotSpot(TM) Client VM (build 25.211-b12, mixed mode)
20877: Proc CaptureConsoleOutput(DosApp :Str;AMemo : TMemo);
20878: Ex.: CaptureConsoleOutput('cmd /C dir *.* ',memo2);
20879:   CaptureConsoleOutput('powershell /C dir *.* ',memo2);
20880: Func ExecuteCommandDOS(CommandLine:str):str;
20881: Func DOSCommandRedirect(const CommandLine:Str;
20882:   const OutStream: Classes.TStream):Bool; overload; //8
20883: Proc SendKeysToWindow(const HWnd: Windows.HWND; const Text:Str);
20884: Func IsRunningOnBattery:Bool;
20885: Func IsHexStr(const S:Str):Bool;
20886: Func IsCharInSet(const Ch: Char; const Chars: TCharSet):Bool;
20887: Func StreamHasWatermark(const Stm:Classes.TStream;const Watermark:array of Byte):Bool;
20888: Func ReadBigEndianWord(Stm: Classes.TStream): Word;
20889: Func DownloadURLToFile(const URL, FileName:Str):Bool;
20890: Func ExtractURIQueryString(const URI:Str):Str;
20891: Func GetBiosVendor:Str;
20892: Func GetIEVersionStr:Str; //18
20893: Func FloatToFixed(const Value: Extended; const DecimalPlaces: Byte;
20894:   const SeparateThousands:Bool):Str;
20895: Func IntToFixed(const Value: Int;const SeparateThousands:Bool):Str;
20896: Func Int64ToFixed(const Value: Int64;const SeparateThousands:Bool):Str;
20897: Func IntToNumberText2(const Value: Int):Str; //22
20898:
20899: Func IsLibraryInstalled2(const LibFileName:Str):Bool;
20900: Func RemainingBatteryPercent: Int;
20901: Proc SetLockKeyState(KeyCode: Int; IsOn:Bool);
20902: Func IsLockKeyOn(const KeyCode: Int):Bool;
20903: Proc PostKeyEx322(const Key: Word; const Shift: Classes.TShiftState;
20904:   const SpecialKey:Bool = False);
20905: Func TerminateProcessByID(ProcessID:Card):Bool;
20906: Func GetWindowProcessName(const Wnd: Windows.HWND):Str;
20907: Func GetProcessName(const PID: Windows.DWORD):Str;
20908: Func GetWindowProcessID(const Wnd: Windows.HWND): Windows.DWORD;
20909: Func IsAppResponding(const Wnd: Windows.HWND):Bool;
20910: Func IsTabletOS:Bool;
20911: Func ProgIDInstalled(const PID:Str):Bool;
20912: Func GetProcessorName:Str;
20913: Func GetProcessorIdentifier:Str; //36
20914: Proc EmptyKeyQueue;
20915: Proc TrimAppMemorySize;
20916: Func GetEnvironmentBlockSize:Card;
20917: Func GetDefaultPrinterName:Str; //40
20918: Proc ListDrives(const List: Classes.TStrings);
20919: Proc MultiSzToStrings(const MultiSz: PChar; const Strings: Classes.TStrings);
20920: Func BrowseURL(const URL:Str):Bool;
20921: Func IsValidURLProtocol(const URL:Str):Bool;
20922: Func ExecAssociatedApp(const FileName:Str):Bool; //45
20923: Func CheckInternetConnection(AHost: PAnsiChar):Bool;
20924: Func MakeSafeHTMLText(TheText:Str):Str;
20925: Func RemoveURIQueryString(const URI:Str):Str;
20926: Func GetRegistryString(const RootKey: Windows.HKEY;const SubKey, Name:str):str;
20927: Proc RefreshEnvironment2(const Timeout:Card = 5000); //50
20928: Func IsKeyPressed2(const VirtKeyCode: Int):Bool;
20929: Func SizeOfFile64(const FileName:Str): Int64;
20930: Func IsHugeFile(const FileName:Str):Bool;
20931: Func SetTransparencyLevel(const HWnd: Windows.HWND;const Level: Byte):Bool;
20932: Func IsEqualResID(const R1, R2: PChar):Bool; //55
20933: Func GetGenericFileType(const FileNameOrExt:Str):Str;
20934: Func GetFileType2(const FilePath:Str):Str;
20935: Proc ShowShellPropertiesDlg(const APath:Str);
20936: Func EllipsifyText(const AsPath:Bool; const Text:Str;
20937:   const Canvas: Graphics.TCanvas; const MaxWidth: Int ):Str;
20938: Func CloneByteArray(const B: array of Byte): TBytes; //60

```

```

20939: Proc AppendByteArray(var B1: TBytes; const B2: array of Byte);
20940: Func IsUnicodeStream(const Stm: Classes.TStream):Bool;
20941: Func FileHasWatermark(const FileName:Str;
20942:     const Watermark: array of Byte; const Offset: Int = 0):Bool;
20943: Func FileHasWatermarkAnsi(const FileName:Str;
20944:     const Watermark: Ansistr; const Offset: Int = 0):Bool;
20945: Func IsASCIISTream(const Stm: Classes.TStream; Count: Int64 = 0;
20946:     BufSize: Int = 8*1024):Bool;
20947: Func IsASCIIFile(const FileName:Str; BytesToCheck: Int64 = 0;
20948:     BufSize: Int = 8*1024):Bool; //66
20949: Func BytesToAnsistr(const Bytes: SysUtils.TBytes; const CodePage: Word):Str;
20950: Func UnicodeStreamToWideString(const Stm: Classes.TStream): WideString;
20951: Proc WideStringToUnicodeStream(const Str: WideString; const Stm: Classes.TStream);
20952: Proc GraphicToBitmap(const Src: Graphics.TGraphic;
20953:     const Dest: Graphics.TBitmap; const TransparentColor: Graphics.TColor); //70
20954: Func URIDecode(S:Str; const IsQueryString:Bool):Str;
20955: Func URIEncode(const S:Str; const InQueryString:Bool):Str;
20956: Func URLDecode(const S:Str):Str;
20957: Func URLEncode(const S:Str; const InQueryString:Bool):Str;
20958: Func AllDigitsSame(N: Int64):Bool;
20959: Func GetDosOutput(CommandLine:Str; Work:Str = 'C:\:Str;
20960: Proc CaptureConsoleOutput(DosApp :Str; AMemo : TMemo);
20961: Ex.: CaptureConsoleOutput('cmd /C dir *.* ',memo2);
20962: CaptureConsoleOutput('powershell /C dir *.* ',memo2);
20963: Func ExecuteCommandDOS(CommandLine:Str):Str;
20964: Func DOSCommandRedirect(const CommandLine:Str;
20965:     const OutStream: Classes.TStream):Bool; overload; //8
20966: Proc SendKeysToWindow(const HWnd: Windows.HWND; const Text:Str); //75
20967: Func FoldWrapText(const Line,BreakStr:Str; BreakChars:TSysCharSet; MaxCol: Int):Str;;
20968: Func TextWrap(const Text:Str; const Width, Margin: Int):Str;
20969: Func ShellExecuteX(Operation, FileName, Parameters, Directory:Str; ShowCmd: Int): Card;;
20970: Func KeyboardStateToShiftState1(const KeyboardState: TKeyboardState): TShiftState;;
20971: Proc SetCustomFormGlassFrame(const CustomForm: TCustomForm; const GlassFrame: TGlassFrame);
20972: Func GetCustomFormGlassFrame(const CustomForm: TCustomForm): TGlassFrame;
20973: Proc SetApplicationMainFormOnTaskBar(const Application: TApplication; Value: Bool);
20974: Func GetApplicationMainFormOnTaskBar(const Application: TApplication): Bool;
20975: Func CompressWhiteSpace(const S:Str):Str;
20976: Func IsASCIIDigit(const Ch: Char):Bool;
20977: Func CompareNumberStr(const S1, S2:Str): Int;
20978: Proc HexToBuf(HexStr:Str; var Buf:Str);
20979: Func BufToHex(const Buf:Str; const Size: Card):Str;
20980: Func DOSExec( CommandLine :Str; Work :Str) :Str;
20981: Proc HexToStrBuf(HexStr:Str; var Buf:Str);
20982: Func StrBufToHex(const Buf:Str; const Size: Card):Str;;
20983: Func GetCharFromVirtualKey(AKey: Word):Str;
20984: Func GetParentProcessID(const PID: Windows.DWORD): Windows.DWORD;
20985: Func FormInstanceCount2(AFormClass: Forms.TFormClass): Int; //overload;
20986: Func FormInstanceCount(const AFormClassName:Str): Int; //overload;
20987: Func FindAssociatedApp(const Doc:Str):Str;
20988: Func CreateShellLink2(const LinkFileName, AssocFileName, Desc, WorkDir, Args, IconFileName:Str; const
    IconIdx: Int):Bool;
20989: Func FileFromShellLink(const LinkFileName:Str):Str;
20990: Func IsShellLink(const LinkFileName:Str):Bool;
20991: Func ResourceIDToStr(const ResID: PChar):Str;
20992: Func IsEqualResID(const R1, R2: PChar):Bool;
20993: Func RecycleBinInfo(const Drive: Char; out BinSize, FileCount: Int64):Bool;
20994: Func FormInstanceCount2(AFormClass: TFormClass): Int;;
20995: Func FormInstanceCount(const AFormClassName:Str): Int;;
20996: Func FindAssociatedApp(const Doc:Str):Str;;
20997: Func CreateShellLink2(const LinkFileName, AssocFileName, Desc, WorkDir, Args
20998:     ,IconFileName:Str; const IconIdx: Int):Bool;
20999: Func FileFromShellLink(const LinkFileName:Str):Str;
21000: Func IsShellLink(const LinkFileName:Str):Bool;
21001: Func ResourceIDToStr(const ResID: PChar):Str;
21002: Func IsEqualResID(const R1, R2: PChar):Bool;
21003: Func RecycleBinInfo(const Drive: Char; out BinSize, FileCount: Int64):Bool;
21004: Func SysImageListHandle(const Path:Str; const WantLargeIcons: Boolean): THandle;
21005: Func EmptyRecycleBin:Bool;
21006: Func ExploreFile(const Filename:Str ):Bool;
21007: Func ExploreFolder(const Folder:Str):Bool;
21008: Proc ClearRecentDocs2;
21009: Proc AddToRecentDocs2(const FileName:Str); //100
21010: Func StringsToMultiSz(const Strings: TStrings; const MultiSz: PChar; const BufSize: Int): Int;
21011: Proc DrawTextOutline(const Canvas: TCanvas; const X, Y: Int; const Text: Str);
21012: Func CloneGraphicAsBitmap(const Src: Graphics.TGraphic; const PixelFmt: Graphics.TPixelFormat;
21013:     const TransparentColor: Graphics.TColor): Graphics.TBitmap;
21014: Proc InvertBitmap(const ABitmap: Graphics.TBitmap); //overload;
21015: Proc InvertBitmap2(const SrcBmp, DestBmp: Graphics.TBitmap); //overload;
21016: Func MakeFilenameInExePath( aFilename: TFilename): TFilename;
21017: Func YearOfDate2( DateTime : TDateTime) : Int;
21018: Func MonthOfDate2( DateTime : TDateTime) : Int;
21019: Func DayOfDate2( DateTime : TDateTime) : Int;
21020: Func HourOfTime2( DateTime : TDateTime) : Int;
21021: Func MinuteOfTime2( DateTime : TDateTime) : Int;
21022: Func SecondOfTime2( DateTime : TDateTime) : Int;
21023: Func IsLeapYear2( DateTime : TDateTime) : Bool;
21024: Func DaysInMonth2( DateTime : TDateTime) : Int;
21025: Func MakeUTCtime( DateTime : TDateTime) : TDateTime;
21026: Func MakeLocalTimeFromUTC( DateTime : TDateTime) : TDateTime;

```

```

21027: Func IsStandardTime :Bool;
21028: Func UnixNow : Int64;
21029: Func NowString :Str;
21030: Func MakeClosedTag( aTagName, aTagValue :Str) :Str;
21031: Func MakeOpenTag( aTagName, aTagAttributes :Str) :Str;
21032: Func MakeBold( Str :Str) :Str;
21033: Func MakeItalic( Str :Str) :Str;
21034: Func MakeUnderline( Str :Str) :Str;
21035: Func MakeStrikeout( Str :Str) :Str;
21036: Func MakeCenter( Str :Str) :Str;
21037: Func MakeParagraph( Str :Str) :Str;
21038: Func MakeCode( Str :Str) :Str;
21039: Func MakeOption( aValue, aText :Str) :Str;
21040: Func MakeHTMLFontSize( Str :Str; SizeParam :Str) :Str;
21041: Func AddQuotes2( Str :Str) :Str;
21042: Func AddSingleQuotes( Str :Str) :Str;
21043: Func MakeHTMLParam( Str :Str) :Str;
21044: Func MakeLink( URL, name :Str) :Str;
21045: Func MakeLinkTarget( URL, name, Target :Str) :Str;
21046: Func MakeMailTo( Address, name :Str) :Str;
21047: Func HTMLToDelphiColor( S :Str) : TColor;
21048: Func ColorToHTMLHex( Color : TColor) :Str;
21049: Func GetStringFromRes( ResName :Str) :Str;
21050: Func EscapeText( sText :Str) :Str;
21051: Func EncodeForXML( const aString :Str) :Str;
21052: Func IsStringAlpha( Str :Str) :Bool;
21053: Func IsStringNumber( Str :Str) :Bool;
21054: Func EnsurePrefix( aPrefix, aText :Str) :Str;
21055: Func StringToAcceptableChars( S :Str; AcceptableChars : TCharSet) :Str;
21056: Func StringIsAcceptable( S :Str; AcceptableChars : TCharSet) :Bool;
21057: Func ValidateEMailAddress( aEmail :Str) :Bool;
21058: Func FirstChar( Str :Str) : Char;
21059: Func LastChar( Str :Str) : Char;
21060: Func StringIsEmpty( Str :Str) :Bool;
21061: Func StringIsNotEmpty( Str :Str) :Bool;
21062: Func StringHasSpacesInMiddle( Str :Str) :Bool;
21063: Func StringContains( aString :Str; aSubStr :Str) :Bool;
21064: Func SpacesToUnderscore( S :Str) :Str;
21065: Func SpacesToPluses( Str :Str) :Str;
21066: Func SwapString( Str :Str) :Str;
21067: Func MakeCopyrightNotice( aCopyrightHolder :Str) :Str;
21068: Proc WriteStringToFile( aStr :Str; aFilename : TFilename);
21069: //Func WinExecAndWait32( Path : PChar; Visibility : Word) : Int;
21070: Func WinExecAndWait32V2( Filename :Str; Visibility : Int) : DWORD;
21071: Func WindowsExit( RebootParam : Longword) :Bool;
21072: Func GetVersionInfo2 :Str;
21073: Func VersionString( aPrefix :Str; aUseColon :Bool) :Str;
21074: Func CreateTempFileName( aPrefix :Str) :Str;
21075: Func GetWindowsTempDir :Str;
21076: Func GetWindowsDir2 :Str;
21077: Func GetSystemDir2 :Str;
21078: Func GetSpecialFolderLocation( aFolderType : Int) :Str;
21079: Func GetTextFileContents( aFilename : TFilename) :Str;
21080: Func GetBDSDir( aVersion : Int) :Str;
21081: Func CaptionMessageDlg(const aCaption:str;const Msg:str;DlgType:TMsgDlgType;Buttons
TMsgDlgButtons;HelpCtx:Longint):Int;
21082: Func StreamFileCopy(const SourceFilename,TargetFilename:str;KeepDate:Boolean):Int;
21083: Func MakePercentString( f : Double) :Str;
21084: Proc DumpKey( var aKey : Char);
21085: Func ValidateKey2(var aKey:Char; AcceptableKeys:TCharSet;KillTheKey:Bool) :Bool;
21086: Func MakeInterbaseString( aHostName, aFilename :Str) :Str;
21087: Func ParseToken( const S :Str; var FromPos : Int; Delimiter : Char):Str;
21088: Proc MatchBounds( MovedControl, TemplateControl : TControl);
21089: Proc LockWidth( aControl : TControl);
21090: Proc LockHeight( aControl : TControl);
21091: Proc LockBounds( aControl : TControl);
21092: Func TruncateFilename(aCanvas:TCanvas;aRect:TRect;aFilename:str;aMargin Int):Str;
21093: Func PointInRect2( const R : TRect; const P : TPoint) :Bool;
21094: Func PointInRect3( const R : TRect; const X, Y : Int) :Bool;
21095: Proc VariantToStream2( const V : OLEVariant; Stream : TStream);
21096: Proc StreamToVariant2( Stream : TStream; var V : OLEVariant);
21097: Proc AssignDocument( Browser : TWebBrowser; Text :Str);
21098: Proc LoadStreamToWebBrowser( WebBrowser : TWebBrowser; Stream : TStream);
21099: Proc SaveWebBrowserSourceToStream( Document : IDispatch; Stream : TStream);
21100: Proc GetStylesFromBrowser( aBrowser : TWebBrowser; aStyles : TStrings);
21101: TypeS('TnxBits', 'Int; TypeS('TDayHours', 'Int;
21102: ClassN(CL.FindClass('TOBJECT','NixUtilsException;
21103: Func REG_CURRENT_VERSION :Str;
21104: Func RegisteredOwner :Str;
21105: Func RegisteredCompany :Str;
21106: Func GetLocalComputerName2 :Str;
21107: Func GetLocalUserName2 :Str;
21108: Func DeleteToRecycleBin(WindowHandle: HWND; Filename:str; Confirm:Bool): Bool;
21109: Func RemoveBackSlash2( const Dir :Str) :Str;
21110: Func EnsureBackSlash( aPath :Str) :Str;
21111: Func EnsureForwardSlash( aPath :Str) :Str;
21112: Func RemoveLeadingSlash( aPath :Str) :Str;
21113: Func SameDirectories( aDir1, aDir2 : TFilename) :Bool;
21114: Func EnsureExtensionHasLeadingPeriod( aExtension :Str) :Str;

```



```

21115: Func RemoveExtension( aFilename :Str) :Str;
21116: Func GetModuleFileNameStr :Str;
21117: Func ModulePath :Str;
21118: Func IniFileName :Str;
21119: Func MakeFileNameInExePath( aFilename : TFilename) : TFilename;
21120: Func IsBitSet( Bits : Int; BitToSet : TnxBits) :Bool;
21121: Func SetBit6( Bits : Int; BitToSet : TnxBits) : Int;
21122: Func UnSetBit( Bits : Int; BitToSet : TnxBits) : Int;
21123: Func FlipBit( Bits : Int; BitToSet : TnxBits) : Int;
21124: Func / FontToOleFont
21125: Func / OleFontToFont
21126: Func GetCachedFileFromURL(strUL:Str; var strLocalFile:Str):Bool;
21127: Func IAddrToHostName(const IP:Str):Str;
21128: Func GetIEHandle(WebBrowser: TWebbrowser; ClassName:Str): HWND;
21129: Func GetTextFromHandle(WinHandle: THandle):Str;
21130: Proc Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
21131: Func FillWebForm(WebBrowser:TWebBrowser;FieldName:Str; Value:Str):Bool;
21132: Proc WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode:Str);
21133: Func NetSend(dest, Source, Msg:Str): Longint; overload;
21134: end;
21135: Proc SIRegister_UWANTUtils(CL: TPSPascalCompiler);
21136: begin
21137: TErrorSeverity, (errwantNone, errwantInfo, errwantHint, errwantWarning, errwantError, errwantFatal );
21138: Func ErrTag( Sev : TErrorSeverity) :Str;
21139: Func MakeInt64( LowInt32, HiInt32 :Card) : Int64;
21140: Func FolderContent( const APath :Str) : INode;
21141: Func FindFiles2(const AFolder, AMask:Str; Attrs: Int; AList: TStrings; ASearchSubdirs: Bool): Bool;
21142: Func ErrorSeverityStr( ASeverity : TErrorSeverity) :Str;
21143: Func IsParameterOptional( AParamDef : INode) :Bool;
21144: Func ExtractMethodName( const S :Str) :Str;
21145: Func LineNoOf( ANode : INode) : Int;
21146: Func ColNoOf( ANode : INode) : Int;
21147: Func WANTBoolToStr( AValue :Bool) :Str;
21148: Func WANTStrToBool( const AValue :Str; default :Bool) :Bool;
21149: Func StrDefault( const S, Default :Str) :Str;
21150: Func PluralNoun( Count : Int; const S :Str) :Str;
21151: Func IsUnqualifiedName( const AName :Str) :Bool;
21152: Func IsModuleTag( const ATag :Str) :Bool;
21153: Func IsCallableTag( const ATag :Str) :Bool;
21154: Func IsConditionalTag( const ATag :Str) :Bool;
21155: Func IsMethodTag( const ATag :Str) :Bool;
21156: Func StripDriveFromPath( const AFileName :Str) :Str;
21157: Func TotalTime( NumTicks : Int64) :Str;
21158: end;
21159:
21160: Proc SIRegister_IdNNTPServer(CL: TPSPascalCompiler);
21161: begin
21162: TIdNNTPAuthType, '( atUserPass, atSimple, atGeneric );
21163: TIdNNTPAuthTypes, 'set of TIdNNTPAuthType;
21164: SIRegister_TIdNNTPThread(CL);
21165: TypeS('TIdNNTPOnAuth', Proc ( AThread : TIdNNTPThread; var VAccept :Bool);
21166: TIdNNTPOnNewGroupsList, 'Procedure (AThread:TIdNNTPThread;const ADateStamp:TDateTime;const
ADistributions:Str);
21167: TIdNNTPOnNewNews, 'Proc ( AThread : TIdNNTPThread; const Nributions :Str);
21168: TIdNNTPOnHaveCheck, 'Procedure(AThread:TIdNNTPThread;const AMsgID String; VAccept : Bool);
21169: TIdNNTPOnArticleByNo, 'Proc ( AThread : TIdNNTPThread; const AMsgNo : Int);
21170: TIdNNTPOnArticleByID, 'Proc ( AThread : TIdNNTPThread; const AMsgID :Str);
21171: TIdNNTPOnCheckMsgNo, 'Procedure(AThread:TIdNNTPThread;const AMsgNo: Int; var VMsgID :Str);
21172: TIdNNTPOnCheckMsgID, 'Proc (AThread:TIdNNTPThread; const AMsgID:Str;var VMsgNo : Int);
21173: TIdNNTPOnMovePointer, 'Proc (AThread TIdNNTPThread;var AMsgNo: Int;var VMsgID:Str);
21174: TIdNNTPOnPost, 'Procedure(AThread:TIdNNTPThread;var VPostOk:Boolean;var VErrorText :Str);
21175: TIdNNTPOnSelectGroup, 'Procedure(AThread:TIdNNTPThread;const AGroup:Str;var VMsgCount: Int;var
VMsgFirst: Int;var
+VMsgLast : Int; var VGroupExists :Bool);
21177: TIdNNTPOnCheckListGroup, 'Proc ( AThread : TIdNNTPThread; '
+const AGroup :Str; var VCanJoin :Bool; var VFirstArticle : Int);
21179: TIdNNTPOnXHdr, 'Proc ( AThread : TIdNNTPThread; const AHea
+derName:Str; const AMsgFirst: Int; const AMsgLast: Int;const AMsgID:Str);
21181: TIdNNTPOnXOver, 'Proc (AThread:TIdNNTPThread;const AMsgFirst: Int;const AMsgLast : Int);
21182: TIdNNTPOnXPat, 'Procedure(AThread:TIdNNTPThread;const AHeaderName:Str;const AMsgFirst: Int; const
AMsgLast: Int;const AMsgID:Str;const AHeaderPattern:Str);
21183: TIdNNTPOnAuthRequired, 'Procedure(AThread:TIdNNTPThread;const ACommand, AParams:Str;var VRequired:Bool);
21184: TIdNNTPOnListPattern, 'Proc (AThread:TIdNNTPThread;const AGroupPattern:Str);
21185: SIRegister_TIdNNTPServer(CL);
21186: end;
21187:
21188: Proc SIRegister_OverbyteIcsAsn1Utils(CL: TPSPascalCompiler);
21189: begin
21190: 'ASN1UtilsVersion', 'LongInt.SetInt( 101);
21191: ConstantN('ASN1_BOOL', 'LongWord.SetUInt( $01);
21192: 'ASN1_INT', 'LongWord.SetUInt( $02);
21193: 'ASN1_OCTSTR', 'LongWord'.SetUInt( $04);
21194: 'ASN1_NULL', 'LongWord'.SetUInt( $05);
21195: 'ASN1_OBJID', 'LongWord'.SetUInt( $06);
21196: 'ASN1_ENUM', 'LongWord'.SetUInt( $0a);
21197: 'ASN1_SEQ', 'LongWord'.SetUInt( $30);
21198: 'ASN1_SETOF', 'LongWord'.SetUInt( $31);
21199: 'ASN1_IPADDR', 'LongWord'.SetUInt( $40);
21200: 'ASN1_COUNTER', 'LongWord'.SetUInt( $41);

```

```

21201: 'ASN1_GAUGE','LongWord').SetUInt( $42);
21202: 'ASN1_TIMETICKS','LongWord').SetUInt( $43);
21203: 'ASN1_OPAQUE','LongWord').SetUInt( $44);
21204: Func ASNEncOIDItem2( Value : Int) : Ansistr;
21205: Func ASNDecOIDItem2( var Start : Int; const Buffer : Ansistr) : Int;
21206: Func ASNEncLen2( Len : Int) : Ansistr;
21207: Func ASNDecLen2( var Start : Int; const Buffer : Ansistr) : Int;
21208: Func ASNEncInt2( Value : Int) : Ansistr;
21209: Func ASNEncUInt2(Value : Int) : Ansistr;
21210: Func ASNObject2( const Data : Ansistr; ASNTType : Int) : Ansistr;
21211: Func ASNItem2( var Start:Int; const Buffer:Ansistr; var ValueType:Int):Ansistr;
21212: Func MibToId2( Mib :Str) : Ansistr;;
21213: //Func MibToId1( Mib : Ansistr) : Ansistr;;
21214: Func IdToMib2( const Id : Ansistr) :Str;
21215: Func IntMibToStr2( const Value : Ansistr) : Ansistr;
21216: Func ASNdump2( const Value : Ansistr) :Str;
21217: end;
21218: const SM_MEDIACENTER = 87; //metrics flag not defined in Windows unit
21219: SM_TABLETPC = 86;
21220: http://snippets.delphidabbler.com/#
21221:
21222: Proc SIRegister_wmiserv(CL: TPSPascalCompiler);
21223: begin
21224: ConstantN('EOAC_NONE','LongInt').SetInt( 0);
21225: 'RPC_C_AUTHN_WINNT','LongInt').SetInt( 10);
21226: 'RPC_C_AUTHZ_NONE','LongInt').SetInt( 0);
21227: 'RPC_E_CHANGED_MODE','LongInt').SetInt( - 2147417850);
21228: Func WMIStart : ISWBemLocator;
21229: Func WMIConnect(WBemLocator:ISWBemLocator;Server,account,password:str):ISWBemServices;
21230: Func WMIExecQuery( WBemServices : ISWBemServices; query :Str) : ISWBemObjectSet;
21231: Func WMIRowFindFirst(ObjectSet:ISWBemObjectSet;var Enum:IEnumVariant;var tempobj:OleVariant):bool;
21232: Func WMIRowFindNext( Enum : IEnumVariant; var tempobj : OleVariant) :Bool;
21233: Func WMIColFindFirst(var propEnum:IEnumVariant; var tempObj : OleVariant):Bool;
21234: Func WMIColFindNext( propEnum : IEnumVariant; var tempobj : OleVariant) :Bool;
21235: Func WMIGetValue(wbemservices:ISWBemServices; tablename,fieldname:str):Str;
21236: Func WMIConvValue( tempobj : OleVariant; var keyname :Str):Str;
21237: --- wmiserv2:
21238: Func WMIRegConnect(WBemLocator:ISWBemLocator;Server,account,password:str): ISWBemServices;
21239: Proc WMIGetMethodInfo(srv:ISWBemServices;objname,method:str; var regobject,inparams: ISWBemObject);
21240: Proc WMISetValue(InParam: ISWBemObject; keyvalue:Str; inval: OleVariant);
21241: end;
21242:
21243: Proc SIRegister_RegSvrUtils(CL: TPSPascalCompiler);
21244: begin
21245: SIRegister_ERegistryException(CL);
21246: Func RegOpenKey2( Key : HKey; const SubKey :Str) : HKey;
21247: Func RegGetKey( Key : HKey; const SubKey :Str) :Str;
21248: Func RegCanOpenKey( Key : HKey; const SubKey :Str; var OutKey : HKey) :Bool;
21249: Func RegKeyExists2( Key : HKey; const SubKey :Str) :Bool;
21250: Func RegCloseAndNilKey( var Key : HKey) :Bool;
21251: Func RegQuerySubKeyCount( Key : HKey) : Int;
21252: Func RegEnumKey2( Key : HKey; Index : Int; var Value :Str) :Bool;
21253: Func RegQueryKey( Key : HKey; const SubKey :Str; var Value :Str) :Bool;
21254: Func RegGetDefaultValue( Key : HKey) :Str;
21255: Func RegGetValueEx( Key : HKey; const ValName :Str) :Str;
21256: Proc ErrorFmt( const Ident :Str; const Args : array of const);
21257: Proc FmtError( const Ident :Str; const Args : array of const);
21258: end;
21259:
21260: Proc SIRegister_osFileUtil(CL: TPSPascalCompiler); //Linux
21261: begin
21262: Func OsDOS2UnixFileAttributes( Attr : LongWord) : LongWord;
21263: Func OsUnix2DosFileAttributes( Attr : LongWord) : LongWord;
21264: Func OsUnixFileTimeToDateTime( UnixTime : LongInt) : TDateTime;
21265: Func OsDateTimeToUnixFileTime( DateTime : TDateTime) : LongInt;
21266: Func OsDosFileTimeToDateTime( DosTime : LongInt) : TDateTime;
21267: Func OsDateTimeToDosFileTime( Value : TDateTime) : LongInt;
21268: Func OsFileTimeToLocalFileTime( FileTime : LongInt) : LongInt;
21269: Func OsLocalFileTimeToFileTime( FileTime : LongInt) : LongInt;
21270: end;
21271:
21272: Proc SIRegister_TWebBrowser(CL: TPSPascalCompiler);
21273: begin
21274: //with RegClassS(CL,'TOleControl', 'TWebBrowser') do
21275: with ClassN(CL.FindClass('TOleControl'),'TWebBrowser') do begin
21276: RegisterMethod(Proc GoBack;
21277: Proc GoForward;
21278: Proc GoHome;
21279: Proc GoSearch;
21280: Proc Navigate7( const URL : WideString);;
21281: Proc Navigate8( const URL : WideString; const Flags : OleVariant);;
21282: Proc Navigate9(const URL:WideString;const Flags:OleVariant;const TargetFrameName: OleVariant);;
21283: Proc Navigate10(const URL:WideString;const Flags:OleVariant;const TargetFrameName:OleVariant;var
PostData:OleVariant);
21284: Proc Navigate11(const URL:WideString;const Flags:OleVariant;const TargetFrameName:OleVariant;var
PostData:OleVariant;const Headers:OleVariant);
21285: Proc Refresh;
21286: Proc Refresh212;;
21287: Proc Refresh213( var Level : OleVariant);;

```

```

21288: Proc Stop;
21289: Proc Quit;
21290: Proc ClientToWindow( var pcx : SYSINT; var pcy : SYSINT);
21291: Proc PutProperty( const Property_ : WideString; vtValue : OleVariant);
21292: Func GetProperty( const Property_ : WideString) : OleVariant;
21293: Proc Navigate214( var URL : OleVariant);
21294: Proc Navigate215( var URL : OleVariant; var Flags : OleVariant);
21295: Proc Navigate216( var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVar);
21296: Proc Navigate217( var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVariant;var PostData:
OleVariant);
21297: Proc Navigate218( var URL:OleVariant;var Flags:OleVariant;var TargetFrameName:OleVar;var
PostData:OleVariant;var Headers:OleVariant);
21298: Func QueryStatusWB( cmdID : OLECMDID) : OLECMDF;
21299: Proc ExecWB19( cmdID : OLECMDID; cmdexeopt : OLECMDEXECHOPT);
21300: Proc ExecWB20( cmdID:OLECMDID;cmdexeopt: OLECMDEXECHOPT; var pvaIn : OleVariant);
21301: Proc ExecWB21( cmdID:OLECMDID;cmdexeopt:OLECMDEXECHOPT;var pvaIn:OleVariant;var pvaOut:OleVariant);
21302: Proc ShowBrowserBar22( var pvaClsid : OleVariant);
21303: Proc ShowBrowserBar23( var pvaClsid : OleVariant; var pvarShow : OleVariant);
21304: Proc ShowBrowserBar24( var pvaClsid:OleVariant;var pvarShow:OleVariant;var pvarSize: OleVariant);
21305: ControlInterface', 'IWebBrowser2', iptr);
21306: RegisterProperty(DefaultInterface, IWebBrowser2', iptr);
21307: Application', 'IDispatch', iptr);
21308: Parent', 'IDispatch', iptr);
21309: Container', 'IDispatch', iptr);
21310: Document', 'IDispatch', iptr);
21311: TopLevelContainer', 'WordBool', iptr);
21312: type_', 'WideString', iptr);
21313: LocationName', 'WideString', iptr);
21314: LocationURL', 'WideString', iptr);
21315: Busy', 'WordBool', iptr);
21316: Name', 'WideString', iptr);
21317: FullName', 'WideString', iptr);
21318: Path', 'WideString', iptr);
21319: Visible', 'WordBool', iptrw);
21320: StatusBar', 'WordBool', iptrw);
21321: StatusText', 'WideString', iptrw);
21322: ToolBar', 'Int', iptrw);
21323: MenuBar', 'WordBool', iptrw);
21324: FullScreen', 'WordBool', iptrw);
21325: Offline', 'WordBool', iptrw);
21326: Silent', 'WordBool', iptrw);
21327: RegisterAsBrowser', 'WordBool', iptrw);
21328: RegisterAsDropTarget', 'WordBool', iptrw);
21329: TheaterMode', 'WordBool', iptrw);
21330: AddressBar', 'WordBool', iptrw);
21331: Resizable', 'WordBool', iptrw);
21332: OnStatusTextChange', 'TWebBrowserStatusTextChange', iptrw);
21333: OnProgressChange', 'TWebBrowserProgressChange', iptrw);
21334: OnCommandStateChange', 'TWebBrowserCommandStateChange', iptrw);
21335: OnDownloadBegin', 'TNotifyEvent', iptrw);
21336: OnDownloadComplete', 'TNotifyEvent', iptrw);
21337: OnTitleChange', 'TWebBrowserTitleChange', iptrw);
21338: OnPropertyChange', 'TWebBrowserPropertyChange', iptrw);
21339: OnBeforeNavigate2', 'TWebBrowserBeforeNavigate2', iptrw);
21340: OnNewWindow2', 'TWebBrowserNewWindow2', iptrw);
21341: OnNavigateComplete2', 'TWebBrowserNavigateComplete2', iptrw);
21342: OnDocumentComplete', 'TWebBrowserDocumentComplete', iptrw);
21343: OnQuit', 'TNotifyEvent', iptrw);
21344: OnVisible', 'TWebBrowserOnVisible', iptrw);
21345: OnToolBar', 'TWebBrowserOnToolBar', iptrw);
21346: OnMenuBar', 'TWebBrowserOnMenuBar', iptrw);
21347: OnStatusBar', 'TWebBrowserOnStatusBar', iptrw);
21348: OnFullScreen', 'TWebBrowserOnFullScreen', iptrw);
21349: OnTheaterMode', 'TWebBrowserOnTheaterMode', iptrw);
21350: OnWindowSetResizable', 'TWebBrowserWindowSetResizable', iptrw);
21351: OnWindowSetLeft', 'TWebBrowserWindowSetLeft', iptrw);
21352: OnWindowSetTop', 'TWebBrowserWindowSetTop', iptrw);
21353: OnWindowSetWidth', 'TWebBrowserWindowSetWidth', iptrw);
21354: OnWindowSetHeight', 'TWebBrowserWindowSetHeight', iptrw);
21355: OnWindowClosing', 'TWebBrowserWindowClosing', iptrw);
21356: OnClientToHostWindow', 'TWebBrowserClientToHostWindow', iptrw);
21357: OnSetSecureLockIcon', 'TWebBrowserSetSecureLockIcon', iptrw);
21358: OnFileDownload', 'TWebBrowserFileDownload', iptrw);
21359: OnNavigateError', 'TWebBrowserNavigateError', iptrw);
21360: OnPrintTemplateInstantiation', 'TWebBrowserPrintTemplateInstantiation', iptrw);
21361: OnPrintTemplateTeardown', 'TWebBrowserPrintTemplateTeardown', iptrw);
21362: OnUpdatePageStatus', 'TWebBrowserUpdatePageStatus', iptrw);
21363: OnPrivacyImpactedStateChange', 'TWebBrowserPrivacyImpactedStateChange', iptrw);
21364: end;
21365: end;
21366:
21367: Proc SIRegister_xutils(CL: TPSPascalCompiler);
21368: begin
21369: //TypeS('PshortString', '^ShortString // will not work;
21370: 'EXIT_DOSERROR','LongInt').SetInt( 2);
21371: 'EXIT_ERROR','LongInt').SetInt( 1);
21372: 'adCmdTxt','LongInt').SetInt($00000001);
21373: 'adExecNoRecords','LongInt').SetInt($00000080);
21374: {** Byte order mark: UTF-8 encoding signature }

```

```

21375: ('BOM_UTF8','String').SetString( #SEF#SBB#SBF);
21376: ** Byte order mark: UTF-8 encoding signature }
21377: BOM_UTF8 = #SEF#SBB#SBF;
21378: ** Byte order mark: UCS-4 big endian encoding signature }
21379: BOM_UTF32_BE = #00#00#SFE#SFF;
21380: ** Byte order mark: UCS-4 little endian encoding signature }
21381: BOM_UTF32_LE = #SFF#SFE#00#00;
21382: BOM_UTF16_BE = #SFE#SFF;
21383: BOM_UTF16_LE = #SFF#SFE;
21384: //const adCmdTxt = $00000001; //adExecNoRecords = $00000080;
21385: Func AnsiFileExists( const FName :Str) :Bool;
21386: Func AnsiDirectoryExists( DName :Str) :Bool;
21387: Proc SwapLong2( var x : longword);
21388: Proc SwapWord2( var x : word);
21389: Func UpString( s :Str) :Str;
21390: Func LowString( s :Str) :Str;
21391: Func AddDoubleQuotes( s :Str) :Str;
21392: Func RemoveDoubleQuotes( s :Str) :Str;
21393: Proc StreamErrorProcedure( var S : TStream);
21394: Func StrToken( var Text :Str; Delimiter : Char; UseQuotes:boolean):Str;
21395: Func StrGetNextLine( var Text :Str) :Str;
21396: Func TrimLeftx( const S :Str) :Str;
21397: Func TrimRightx( const S :Str) :Str;
21398: Func hexstr( val : longint; cnt : byte) :Str;
21399: Func decstr( val : longint; cnt : byte) :Str;
21400: Func decstrunsigned( l : longword; cnt : byte) :Str;
21401: Func boolstr( val :Bool; cnt : byte) :Str;
21402: Func CompareByte( buf1, buf2 : pchar; len : longint) : Int;
21403: Func Trimx( const S :Str) :Str;
21404: Func Printf2( const s :Str; var Buf:Str; size : word) :Str;
21405: Func Printfx( const s :Str; var Buf:Str; size : word) :Str;
21406: Func FillTo( s :Str; tolength : Int) :Str;
21407: Func stringdup( const s :Str) :Str;
21408: //Proc stringdispose( var p : pShortstring);
21409: Func EscapeToPascal( const s :Str; var code : Int) :Str;
21410: Func ValDecimal( const S :Str; var code : Int) : longint;
21411: Func ValUnsignedDecimal( const S :Str; var code : Int) : longword;
21412: Func ValOctal( const S :Str; var code : Int) : longint;
21413: Func ValBinary( const S :Str; var code : Int) : longint;
21414: Func ValHexadecimal( const S :Str; var code : Int) : longint;
21415: Func CleanString( const s :Str) :Str;
21416: Func ChangeFileExt2( const FileName, Extension :Str) :Str;
21417: Func fillwithzero( s :Str; newlength : Int) :Str;
21418: Func removenuLLs( const s :Str) :Str;
21419: //ConstantN('WhiteSpace','Char').SetString( ' ' or #10 or #13 or #9);
21420: end;
21421:
21422: Proc SIRegister_ietf(CL: TPSPascalCompiler);
21423: begin
21424: Func mime_isvalidcontenttype( const s : shortstring) :Bool;
21425: Func langtag_isvalid( const s :Str) :Bool;
21426: Func langtag_split( const s :Str; var primary, sub :Str) :Bool;
21427: URI_START_DELIMITER_CHAR,'String').SetString( '<';
21428: URI_END_DELIMITER_CHAR,'String').SetString( '>';
21429: URI_SCHEME_NAME_EMAIL,'String').SetString( 'mailto;
21430: URI_SCHEME_SEPARATOR','String').SetString( ':';
21431: Func uri_split( url :Str; var scheme, authority, path, query :Str) :Bool;
21432: Func urn_isvalid( s : shortstring) :Bool;
21433: Func urn_isvalidnid( nid :Str) :Bool;
21434: Func urn_split( urn :Str; var urnidstr, nidstr, nssstr :Str) :Bool;
21435: Func urn_pathsplit( path :Str; var namespace, nss :Str) :Bool;
21436: Func http_pathsplit( path :Str; var directory, name :Str) :Bool;
21437: Func file_pathsplit( path :Str; var directory, name :Str) :Bool;
21438: end;
21439:
21440: Proc SIRegister_dateutilreal(CL: TPSPascalCompiler);
21441: begin
21442: TypeS('TDatetimeReal', 'real;
21443: TypeS(TDateInfo', 'record DateTime : TDateTime; UTC :Bool; end;
21444: //TypeS('float', 'real;
21445: TypeS('big_int_t', 'int64;
21446: 'DayMondayR','LongInt').SetInt( 1);
21447: ConstantN('DayTuesdayR','LongInt').SetInt( 2);
21448: 'DayWednesdayR','LongInt').SetInt( 3);
21449: 'DayThursdayR','LongInt').SetInt( 4);
21450: 'DayFridayR','LongInt').SetInt( 5);
21451: 'DaySaturdayR','LongInt').SetInt( 6);
21452: 'DaySundayR','LongInt').SetInt( 7);
21453: Func CurrentYearreal : word;
21454: Func Datereal : TDatetimeReal;
21455: Func DateOfreal( const AValue : TDatetimeReal) : TDatetimeReal;
21456: Func DateTimeToStrreal( DateTime : TDatetimeReal) :Str;
21457: Func DateToStrreal( date : TDatetimeReal) :Str;
21458: Func DayOfreal( const AValue : TDatetimeReal) : Word;
21459: Func DaysBetweenreal( const ANow, AThen : TDatetimeReal) : Int;
21460: Proc DecodeDatereal( Date : TDatetimeReal; var Year, Month, Day : Word);
21461: Proc DecodeDateTimereal(const AValue:TDatetimeReal;var Year,Month,Day,Hour,Minute,Second,
MilliSecond:Word);
21462: Proc DecodeTimereal( Time : TDatetimeReal; var Hour, Min, Sec, MSec : Word);

```

```

21463: Func HourOfreal( const AValue : TDateTimeReal) : Word;
21464: Func IncDayreal(const AValue:TDateTimeReal;const ANumberOfDays:Int): TDateTimeReal;
21465: Func IncHourreal(const AValue:TDateTimeReal;const ANumberOfHours:longint):TDateTimeReal;
21466: Func IncMilliSecondreal(const AValue:TDateTimeReal; const ANumberOfMilliseconds: big_Int_t):TDateTimeReal;
21467: Func IncMinutereal(const AValue: TDateTimeReal; const ANumberOfMinutes: big_Int_t) : TDateTimeReal;
21468: Func IncSecondreal(const AValue: TDateTimeReal; const ANumberOfSeconds: big_Int_t) : TDateTimeReal;
21469: Func IncWeekreal(const AValue:TDateTimeReal;const ANumberOfWeeks:Int): TDateTimeReal;
21470: Func IsPMreal( const AValue : TDateTimeReal) :Bool;
21471: Func IsValidDatereal( const AYear, AMonth, ADay : Word) :Bool;
21472: Func IsValidDateTimereal(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSec:Word):Bool;
21473: Func IsValidTimereal( const AHour, AMinute, ASecond, AMilliSecond : Word) :Bool;
21474: Func MinuteOfreal( const AValue : TDateTimeReal) : Word;
21475: Func MonthOfreal( const AValue : TDateTimeReal) : Word;
21476: Func Nowreal : TDateTimeReal;
21477: Func SameDatereal( const A, B : TDateTimeReal) :Bool;
21478: Func SameDateTimereal( const A, B : TDateTimeReal) :Bool;
21479: Func SameTimereal( const A, B : TDateTimeReal) :Bool;
21480: Func SecondOfreal( const AValue : TDateTimeReal) : Word;
21481: Func Timereal : TDateTimeReal;
21482: Func GetTimereal : TDateTimeReal;
21483: Func TimeOfreal( const AValue : TDateTimeReal) : TDateTimeReal;
21484: Func TimeToStrreal( Time : TDateTimeReal) :Str;
21485: Func Todayreal : TDateTimeReal;
21486: Func TryEncodeDatereal( Year, Month, Day : Word; var Date : TDateTimeReal) :Bool;
21487: Func TryEncodeTimereal(Hour,Min,Sec MSec: Word; var Time : TDateTimeReal) :Bool;
21488: Func TryEncodeDateTimereal(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond:Word;var
AValue:TDateTimeReal):Boolean;
21489: Func TryStrToDatereal( const S :Str; var Value : TDateTimeReal) :Bool;
21490: Func TryStrToDateTimereal( const S :Str; var Value : TDateTimeReal) :Bool;
21491: Func TryStrToTimereal( const S :Str; var Value : TDateTimeReal) :Bool;
21492: Func YearOfreal( const AValue : TDateTimeReal) : Word;
21493: Func parsetimeISO(timestr:str;var hourval,minval,secval:word;var offsethourval,offsetminval:Int;var
UTC:bool):bool;
21494: Func parsedateISO( datestr :Str; var yearval, monthval, dayval : word) :Bool;
21495: Func datetojd( year, month, day, hour, minute, second, millisecond : word) : float;
21496: Proc jdtodate(jday:float; var year, month, day, hour, minute, second, msec : word);
21497: Func converttoisotime( timestr :Str) :Str;
21498: Func AdobeDateToISODate( s :Str) :Str;
21499: Func RFC822ToISODateTime( s :Str) :Str;
21500: Proc getdatedos( var year, month, mday, wday : word);
21501: Proc gettimedos( var hour, minute, second, sec100 : word);
21502: end;
21503:
21504: Proc SIRegister_dateext4(CL: TPSPascalCompiler);
21505: begin
21506: TypeS('tfiletime2', 'record LowDateTime: longword; HighDateTime: longword; end;
21507: // TypeS('big_Int_t', 'int64;
21508: Func DosToWinTime( DTime : longint; var Wtime : TFileTime) : longbool;
21509: Func WinToDosTime( const Wtime : TFileTime; var DTime : longint) : longbool;
21510: Func TryStrToDateTimeExt(const S:str;var Value:TDateTime;var UTC:boolean):Bool;
21511: Func TryEncodeDateAndTimeToStr(const Year,Month,Day,Hour,Minute,Second,MilliSecond:word;UTC:boolean;var
AValue:str):boolean;
21512: Func DateTimeToStrExt( DateTime : TDateTime; utc :Bool) :Str;
21513: Proc GetCurrentDate( var Year, Month, Day, DayOfWeek : Int);
21514: Proc GetCurrentTime2( var Hour, Minute, Second, Sec100 : Int);
21515: Func TryUNIXToDateTimeExt(unixtime:big_Int_t;var DateTime:TDateTime;var UTC:bool): bool;
21516: Func TryFileTimeToDateTimeExt( ftime : tfiletime; var DateTime : TDateTime; var UTC :Bool): bool;
21517: Proc JulianToGregorian(JulianDN:big_Int_t;var Year,Month,Day:Word);
21518: end;
21519:
21520:
21521: Proc SIRegister_locale(CL: TPSPascalCompiler);
21522: begin
21523: Func GetISODateString( Year, Month, Day : Word) :Str;
21524: Func GetISODateStringBasic( Year, Month, Day : Word) :Str;
21525: Func IsValidISODateString( datestr : shortstring; strict :Bool) :Bool;
21526: Func IsValidISODateStringExt(datestr:shortstring;strict:bool;var Year,Month,Day:word):bool;
21527: Func IsValidISOTimeString( timestr : shortstring; strict :Bool) :Bool;
21528: Func IsValidISOTimeStringExt(timestr:shortstring;strict:boolean;var hour,min,sec:word;var offhour,offmin:
smallint):bool;
21529: Func IsValidISODateTimeString( str : shortstring; strict :Bool) :Bool;
21530: Func GetISOTimeString( Hour, Minute, Second : Word; UTC :Bool) : shortstring;
21531: Func GetISOTimeStringBasic(Hour,Minute,Second: Word; UTC :Bool) : shortstring;
21532: Func GetISODateTimeString(Year,Month,Day,Hour,Minute,Second:Word;UTC:Bool):shortstring;
21533: Proc UNIXToDateTime2(epoch:big_Int_t; var year,month,day,hour,minute,second: Word);
21534: Func GetCharEncoding( alias :Str; var name :Str) : Int;
21535: Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
21536: Func MicrosoftLanguageCodeToISOCode( langcode : Int) :Str;
21537: CHAR_ENCODING_UTF8,'LongInt').SetInt( 0);
21538: ConstantN(CHAR_ENCODING_UNKNOWN,'LongInt').SetInt( - 1);
21539: 'CHAR_ENCODING_UTF32BE','LongInt').SetInt( 1);
21540: 'CHAR_ENCODING_UTF32LE','LongInt').SetInt( 2);
21541: 'CHAR_ENCODING_UTF16LE','LongInt').SetInt( 3);
21542: 'CHAR_ENCODING_UTF16BE','LongInt').SetInt( 4);
21543: 'CHAR_ENCODING_BYTE','LongInt').SetInt( 5);
21544: 'CHAR_ENCODING_UTF16','LongInt').SetInt( 6);
21545: 'CHAR_ENCODING_UTF32','LongInt').SetInt( 7);
21546: end;
21547:

```



```

21548: Proc SIRegister_Strings (CL: TPSPascalCompiler);
21549: begin
21550:   DelphiFunction('Func StrLenPchar( Str : PChar) : longint;
21551:   Func StrEndPchar( Str : PChar) : PChar;
21552:   Func StrMovePchar( Dest, Source : PChar; l : Longint) : pchar;
21553:   Func StrCopyPchar( Dest, Source : PChar) : PChar;
21554:   Func StrECopyPchar( Dest, Source : PChar) : PChar;
21555:   Func StrLCopyPchar( Dest, Source : PChar; MaxLen : Longint) : PChar;
21556:   Func StrPCopyPchar( Dest : PChar; Source : Str) : PChar;
21557:   Func StrCatPchar( Dest, Source : PChar) : PChar;
21558:   Func strlcatPchar( dest, source : pchar; l : Longint) : pchar;
21559:   Func StrCompPchar( Str1, Str2 : PChar) : Int;
21560:   Func StrICompPchar( Str1, Str2 : PChar) : Int;
21561:   Func StrLCompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Int;
21562:   Func StrLICompPchar( Str1, Str2 : PChar; MaxLen : Longint) : Int;
21563:   Func StrScanPchar( Str : PChar; Ch : Char) : PChar;
21564:   Func StrRScanPchar( Str : PChar; Ch : Char) : PChar;
21565:   Func StrPosPchar( Str1, Str2 : PChar) : PChar;
21566:   Func StrUpperPchar( Str : PChar) : PChar;
21567:   Func StrLowerPchar( Str : PChar) : PChar;
21568:   Func StrPasPchar( Str : PChar) : Str;
21569:   Func StrNewPchar( P : PChar) : PChar;
21570:   Proc StrDisposePchar( P : PChar);
21571: end;
21572:
21573: Proc SIRegister_crc_checks (CL: TPSPascalCompiler);
21574: begin
21575:   Func UpdateCrc32_2( InitCrc : longword; b : byte) : longword;
21576:   Func UpdateCrc16_2( InitCrc : word; b : byte) : word;
21577:   Func UpdateAdler32( InitAdler : longword; b : byte) : longword;
21578:   Func UpdateFletcher8( InitFletcher : word; b : byte) : word;
21579:   Func UpdateCRC( InitCrc : word; b : byte) : word;
21580: end;
21581:
21582: Proc SIRegister_extdos (CL: TPSPascalCompiler);
21583: begin
21584:   TypeS('utf8char', 'char;
21585:   TypeS('putf8char', 'pchar;
21586:   TypeS('utf16char', 'word;
21587:   TypeS('ucs4char', 'longword;
21588:   //TypeS('pucs4char', '^ucs4char // will not work;
21589:   TypeS('ucs2char', 'word;
21590:   TypeS('pucs2char', 'ucs2char; // will not work;
21591:   //TypeS('pucs2char', '^ucs2char // will not work;
21592:   TypeS('utf8string', 'string;
21593:   // TypeS('putf8shortstring', '^shortstring // will not work;
21594:   TypeS(' _USER_INFO_2', record usri2_name : pucs2char; usri2_password : '
21595:   +pucs2char; usri2_password_age : DWORD; usri2_priv : DWORD; usri2_home_dir '
21596:   +: pucs2char; usri2_comment : pucs2char; usri2_flags : DWORD; usri2_script '
21597:   +path : pucs2char; usri2_auth_flags : DWORD; usri2_full_name : pucs2char; u'
21598:   +sri2_usr_comment : pucs2char; usri2_parms : pucs2char; usri2_workstations '
21599:   +: pucs2char; usri2_last_logon : DWORD; usri2_last_logoff : DWORD; usri2_ac'
21600:   +ct_expires : DWORD; usri2_max_storage : DWORD; usri2_units_per_week : DWOR'
21601:   +d; usri2_logon_hours : pchar; usri2_bad_pw_count : DWORD; usri2_num_logons'
21602:   +:DWORD; usri2_logon_server:pucs2char;usri2_country_code:DWORD;usri2_code_page:DWORD; end;
21603:   //TypeS('P_USER_INFO_2', '^ _USER_INFO_2 // will not work;
21604:   TypeS('SE_OBJECT_TYPE', '( SE_UNKNOWN_OBJECT_TYPE, SE_FILE_OBJECT, SE_S'
21605:   +ERVICE, SE_PRINTER, SE_REGISTRY_KEY, SE_LMSHARE, SE_KERNEL_OBJECT, SE_WIND'
21606:   +OW_OBJECT, SE_DS_OBJECT, SE_DS_OBJECT_ALL, SE_PROVIDER_DEFINED_OBJECT, SE_'
21607:   +WMIGUID_OBJECT );
21608:   //TypeS('PPSID', '^PSID // will not work;
21609:   TypeS('ASSOCF', 'DWORD;
21610:   TypeS('ASSOCSTR', '( ASSOCSTR_NONE, ASSOCSTR_COMMAND, ASSOCSTR_EXECUTAB'
21611:   +'LE, ASSOCSTR_FRIENDLYDOCNAME, ASSOCSTR_FRIENDLYAPPNAME, ASSOCSTR_NOOPEN, A'
21612:   +'SSOCSTR_SHELLNEWVALUE,ASSOCSTR_DDECOMMAND,ASSOCSTR_DDEIFEXEC,ASSOCSTR_DDEAPPLICATION,
ASSOCSTR_DDETOPIC);
21613:   TypeS('TCHAR', 'ucs2char;
21614:   {TypeS('TFileTime', 'FILETIME;
21615:   TypeS('TWin32FindDataW', ' _WIN32_FIND_DATAW;
21616:   TypeS('TCHAR', 'ucs2char;
21617:   TypeS('TFileTime', 'FILETIME;
21618:   TypeS('TWin32FindDataW', ' _WIN32_FIND_DATAW; }
21619:   //dev: array[0..127] of char;
21620:   /** Unique file serial number, this may change from one boot to the next.}
21621:   //ino: array[0..127] of char;
21622:
21623:   TypeS('tresourceattribute', '( attr_any, attr_readonly, attr_hidden, at'
21624:   +tr_system, attr_archive, attr_link, attr_directory, attr_temporary, attr_e'
21625:   +ncrypted,attr_no_indexing,attr_device,attr_extended,attr_compressed,attr_offline, attr_sparse);
21626:   TFileAssociation', record appname : utf8string; exename : utf8string; end;
21627:   tresourceattributes', set of tresourceattribute;
21628:   TypeS('TFileStats', record name : utf8string; size : big_Int_t; ow'
21629:   +ner : utf8string; ctime : TDateTime; mtime : TDateTime; atime : TDateTime;'
21630:   +nlink : Int; attributes : tresourceattributes; association : tfileass'
21631:   +ociation; streamcount: Int; accesses: Int; utc : Bool; dev: array[0..127] of char; '
21632:   +ino: array[0..127] of char; comment : utf8string; dirstr : utf8string; end;
21633:   TSearchRecExt', record Stats : TFileStats; FindHandle : THandle'
21634:   +: W32FindData TWin32FindData; IncludeAttr: longint; SearchAttr: TResourceAttributes; end;
21635:   Func GetFileOwner( fname : putf8char) : utf8string;

```

```

21636: GetFileATime( fname : putf8char; var atime : TDateTime) : Int;
21637: GetFileMTime( fname : putf8char; var mtime : TDateTime) : Int;
21638: GetFileCTime( fname : putf8char; var ctime : TDateTime) : Int;
21639: GetFileSizeExt( fname : putf8char) : big_Int_t;
21640: GetFileAttributesExt( fname : putf8char) : tresourceattributes;
21641: GetFileStats( fname : putf8char; var stats : TFileStats) : Int;
21642: DirectoryExistsExt( DName : utf8string) : Bool;
21643: FileExistsExt( const FName : utf8string) : Bool;
21644: GetCurrentDirectoryExt( var DirStr : utf8string) : Bool;
21645: SetCurrentDirectoryExt( const DirStr : utf8string) : Bool;
21646: SetFileATime( fname : putf8char; newatime : tdatetime) : Int;
21647: SetFileMTime( fname : putf8char; newmtime : tdatetime) : Int;
21648: SetFileCTime( fname : putf8char; newctime : tdatetime) : Int;
21649: FindFirstEx(path:putf8char; attr:tresourceattributes;var SearchRec:TSearchRecExt):int;
21650: FindNextEx( var SearchRec : TSearchRecExt) : Int;
21651: Proc FindCloseEx( var SearchRec : TSearchRecExt);
21652: GetUserFullName( account : utf8string) : utf8string;
21653: GetLoginConfigDirectory : utf8string;
21654: GetGlobalConfigDirectory : utf8string;
21655: GetLoginHomeDirectory : utf8string;
21656: // ucs4strnewstr(str:Str; srctype:Str): pucs4char;;
21657: // ucs4strnewucs4(src: pucs4char): pucs4char;;
21658: end;
21659:
21660: Proc SIRegister_xBase(CL: TPSPascalCompiler);
21661: begin
21662:   INADDR_ANY, 'LongWord').SetUInt($00000000);
21663:   //CL.AddConstantN('DCX_EXCLUDEGRN', 'LongWord').SetUInt( $40);
21664:   INVALID_HANDLE_VALUE, 'LongInt').SetInt( DWORD ( - 1 ));
21665:   CL.AddConstantN('INVALID_FILE_SIZE', 'LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21666:   INVALID_FILE_ATTRIBUTES, 'LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21667:   INVALID_FILE_TIME, 'LongWord').SetUInt( DWORD ( $FFFFFFFF ));
21668:   INVALID_REGISTRY_KEY, 'LongWord').SetInt( DWORD ( - 1 ));
21669:   INVALID_VALUE, 'LongWord').SetInt( DWORD ( - 1 ));
21670:   SleepQuant, 'LongInt').SetInt( 1 * 60 * 1000);
21671:   MMaxChars, 'LongInt').SetInt( 250);
21672:   CL.AddTypeS('TBase64Table', '( bsBase64, bsUUE, bsXXE );
21673:   CL.AddTypeS('TWOHandleArray', 'array[0..64 - 1] of THandle;;
21674:   CL.AddTypeS('TWOHandleArray2', 'array[0..MAXIMUM_WAIT_OBJECTS - 1] of THandle;;
21675:   //type TWOHandleArray = array[0..MAXIMUM_WAIT_OBJECTS - 1] of THandle;
21676:   // MAXIMUM_WAIT_OBJECTS = 64;
21677:   SIRegister_TMimeCoder(CL);
21678:   CL.AddTypeS('TSocketOption', '(soBroadcast, soDebug, soDontLinger, soDontRou'
21679:   + 'te, soKeepAlive, soOoBInLine, soReuseAddr, soNoDelay, soBlocking,soAcceptConn );
21680:   TSocketOptions, 'set of TSocketOption;
21681:   //TSocketClass, 'class of TSocket;
21682:   SIRegister_TSocketX(CL);
21683:   TObjProcX, 'Procedure;
21684:   TForEachProc, 'Proc ( P : _Pointer);
21685:   //PFileInfo, '^TFileInfo // will not work;
21686:   TFileInfo, 'record Attr : DWORD; Size : DWORD; Time : DWORD; end;
21687:   TuFindData, 'record Info : TFileInfo; FName :Str; end;
21688:   TCreateFileMode, '( cRead, cWrite, cFlag, cEnsureNew, cTruncate'
21689:   +, cExisting, cShareAllowWrite, cShareDenyRead, cOverlapped, cRandomAccess, cSequentialScan, cDeleteOnClose
);
21690:   TCreateFileModeSet, 'set of TCreateFileMode;
21691:   //PCharSet, '^TCharSet // will not work;
21692:   TCharSetX, 'set of Char;
21693:   //PCharArray, '^TCharArray // will not work;
21694:   //PByteArray, '^TByteArray // will not work;
21695:   //PIntArray, '^TIntArray // will not work;
21696:   //PDwordArray, '^TDwordArray // will not work;
21697:   //PvIntArr, '^TvIntArr // will not work;
21698:   //TvIntArr, 'record Arr : PIntArray; Cnt : Integer; end;
21699:   //PBoolean, '^Boolean // will not work;
21700:   //PItemList, '^TItemList // will not work;
21701:   TThreadMethodX, 'Procedure;
21702:   TThreadPriorityXX, (tpIdle tpLowest, tpLower, tpNormal, tpHigher, tpHighest, tpTimeCritical);
21703:   SIRegister_TThreadX(CL);
21704:   CL.AddClassN(CL.FindClass('TObject'), 'TAdvObject;
21705:   SIRegister_TAdvObject(CL);
21706:   SIRegister_TAdvCpObject(CL);
21707:   //TAdvClass, 'class of TAdvObject;
21708:   //TCollClass, 'class of TColl;
21709:   SIRegister_TColl(CL);
21710:   SIRegister_TSortedColl(CL);
21711:   SIRegister_TStringColl(CL);
21712:   Func AddRightSpaces( const S :Str; NumSpaces : Integer) :Str;
21713:   Proc AddStr( var S :Str; C : char);
21714:   Proc Add_Str( var S : ShortString; C : char);
21715:   Func CompareStrX( const S1, S2 :Str) : Integer;
21716:   Func CopyLeft( const S :Str; I : Integer) :Str;
21717:   Proc DelDoubles( const St :Str; var Source :Str);
21718:   Proc DelFC( var s :Str);
21719:   Proc DelLC( var s :Str);
21720:   Func DelLeft( const S :Str) :Str;
21721:   Func DelRight( const S :Str) :Str;
21722:   Func DelSpaces( const s :Str) :Str;
21723:   Proc DeleteLeft( var S :Str; I : Integer);

```

```

21724: Func DigitsOnly( const AStr :Str) :Bool;
21725: Proc DisposeStr( P :Str);
21726: Func ExpandFileNameX( const FileName :Str) :Str;
21727: Func ExtractFilePathX( const FileName :Str) :Str;
21728: Func ExtractDir( const S :Str) :Str;
21729: Func ExtractFileRootX( const FileName :Str) :Str;
21730: Func ExtractFileExtX( const FileName :Str) :Str;
21731: Func ExtractFileNameX( const FileName :Str) :Str;
21732: Func ExtractFileDriveX( const FileName :Str) :Str;
21733: Func ExtractFileDirX( const FileName :Str) :Str;
21734: Proc FSplit( const FName :Str; var Path, Name, Ext :Str);
21735: Proc FillCharSetX( const AStr :Str; var CharSet : TCharSet);
21736: Func GetWrdStrictUC( var s, w :Str) :Bool;
21737: Func GetWrdStrict( var s, w :Str) :Bool;
21738: Func GetWrdD( var s, w :Str) :Bool;
21739: Func GetWrdA( var s, w :Str) :Bool;
21740: Func GetWrd( var s, w :Str; c : char) :Bool;
21741: Func Hex2X( a : Byte) :Str;
21742: Func Hex4X( a : Word) :Str;
21743: Func Hex8X( a : DWORD) :Str;
21744: Func Int2Hex( a : Integer) :Str;
21745: Func Int2StrX( L : Integer) :Str;
21746: Func ItoS( I : Integer) :Str;
21747: Func ItoSz( I, Width : Integer) :Str;
21748: Func LastDelimiterX( const Delimiters, S :Str) : Integer;
21749: Func LowerCaseX( const S :Str) :Str;
21750: Func MakeFullDir( const D, S :Str) :Str;
21751: Func MakeNormName( const Path, Name :Str) :Str;
21752: Func MonthE( m : Integer) :Str;
21753: Func NewStr( const S :Str) :Str;
21754: Func ReplaceX( const Pattern, ReplaceString:Str; var S:Str):Bool;
21755: Func StoI( const S :Str) : Integer;
21756: Func StrEnds( const S1, S2 :Str) :Bool;
21757: Func StrRightX( const S :Str; Num : Integer) :Str;
21758: Func UpperCaseX( const S :Str) :Str;
21759: Func WipeChars( const AStr, AWipeChars :Str) :Str;
21760: Func _Val( const S :Str; var V : Integer) :Bool;
21761: Func ProcessQuotes( var s :Str) :Bool;
21762: Func UnpackPchars( var s :Str) :Bool;
21763: Func UnpackUchars( var s :Str) :Bool;
21764: Func __alpha( c : char) :Bool;
21765: Func __ctl( c : char) :Bool;
21766: Func __digit( c : char) :Bool;
21767: Func __extra( c : char) :Bool;
21768: Func __national( c : char) :Bool;
21769: Func __pchar( c : char) :Bool;
21770: Func __reserved( c : char) :Bool;
21771: Func __safe( c : char) :Bool;
21772: Func __uchar( c : char) :Bool;
21773: Func __unsafe( c : char) :Bool;
21774: Func Buf2Str( const Buffer:Str) :Str;
21775: //Proc Clear( var Buf, Count : Integer);
21776: Func CompareMem( P1, P2 : _Pointer; Length : Integer) :Bool;
21777: Proc FreeObjectX( var O: TObject);
21778: Proc LowerPrec( var A, B : Integer; Bits : Byte);
21779: Func MemEqu( const A, B, Sz : Integer) :Bool;
21780: Func MaxIX( A, B : Integer) : Integer;
21781: Func MinIX( A, B : Integer) : Integer;
21782: Func MaxDX( A, B : DWORD) : DWORD;
21783: Func MinDX( A, B : DWORD) : DWORD;
21784: Func NulSearch( const Buffer:Str) : Integer;
21785: Func NumBits( I : Integer) : Integer;
21786: //Proc XAdd( var Critical, Normal);
21787: //Proc XChg( var Critical, Normal);
21788: Func CreateEvtA : DWORD;
21789: Func CreateEvt( Initial :Bool) : DWORD;
21790: Func SignaledEvt( id : DWORD) :Bool;
21791: Func WaitEvt( const id : TWOHandleArray; Timeout : DWORD) : DWORD;
21792: //Func WaitEvtA(nCount: Integer;lpHandles: PWOHandleArray; Timeout:DWORD): DWORD;
21793: Func ClearHandle( var Handle : THandle) :Bool;
21794: Proc CloseHandles( const Handles : array of DWORD);
21795: Func FileExistsX( const FName :Str) :Bool;
21796: Func FindExecutableX( FileName, Directory : PChar; Result : PChar) : HINST;
21797: Func GetEnvVariable( const Name :Str) :Str;
21798: Func GetFileNfo( const FName:Str; var Info: TFileInfo; NeedAttr:Boolean):Bool;
21799: Func GetFileNfoByHandle( Handle : DWORD; var Info : TFileInfo) :Bool;
21800: Func ZeroHandle( var Handle : THandle) :Bool;
21801: Func _CreateFile( const FName :Str; Mode : TCreateFileModeSet) : DWORD;
21802: Func _CreateFileSecurity(const
FName:Str;Mode:TCreateFileModeSet;lpSecurityAttributes:PSecurityAttributes):DWORD;
21803: Func _GetFileSize( const FName :Str) : DWORD;
21804: Func _MatchMaskBody( AName, AMask :Str; SupportPercent :Bool) :Bool;
21805: Func _MatchMask(const AName:Str; AMask :Str;SupportPercent:Boolean):Boolean;
21806: Func MatchMask( const AName, AMask :Str) :Bool;
21807: Func SysErrorMsg( ErrorCode : DWORD) :Str;
21808: Func CreateRegKeyX( const AFName :Str) : HKey;
21809: Func OpenRegKeyEx( const AName :Str; AMode : DWORD) : HKey;
21810: Func OpenRegKey( const AName :Str) : DWORD;
21811: Func ReadRegBin(Key: DWORD; const rvn :Str; Bin: _Pointer; Sz : DWORD) :Bool;

```

```

21812: Func ReadRegInt( Key : DWORD; const AStrName :Str) : DWORD;
21813: Func ReadRegString( Key : DWORD; const AStrName :Str) :Str;
21814: Func WriteRegBin(Key:DWORD;const rvn :Str; Bin: _Pointer; Sz : DWORD):Bool;
21815: Func WriteRegInt( Key : DWORD; const AStrName :Str; AValue : DWORD) :Bool;
21816: Func WriteRegString( Key : DWORD; const AStrName, AStr :Str) :Bool;
21817: Func AddrInet( i : DWORD) :Str;
21818: Func GetHostNameByAddr( Addr : DWORD) :Str;
21819: Func Inet2addr( const s :Str) : DWORD;
21820: Func InetAddr( const s :Str) : DWORD;
21821: Proc GlobalFail;
21822: Func _LogOK( const Name :Str; var Handle : DWORD) :Bool;
21823: Proc xBaseDone;
21824: Proc xBaseInit;
21825: Proc uCvtSetFileTime( T : DWORD; var L, H : DWORD);
21826: Func uCvtGetFileTime( L, H : DWORD) : DWORD;
21827: Func uGetSystemTime : DWORD;
21828: Func VtX( const s :Str) : DWORD;
21829: Func VtHX( const s :Str) : DWORD;
21830: Func StrAsg( const Src :Str) :Str;
21831: SRegister_TResetterThread(CL);
21832: CServerVersion','String').SetString( '1.93;
21833: CServerProductName','String').SetString( 'TinyWeb;
21834: end;
21835:
21836: Proc SRegister_ImageHistogram(CL: TPSPascalCompiler);
21837: begin
21838:   THistogramChannel', '( hclGray, hclRed, hclGreen, hclBlue );
21839:   THistogramArray', 'array[0..255] of Cardinal;;
21840:   //THistogramArray = array[0..255] of Cardinal;
21841:   SRegister_THistogram(CL);
21842:   Proc AdjustBitmap(Bmp: TBitmap; Channel: THistogramChannel; Tolerance :Card);
21843:   Proc AdjustBitmap1(Bmp: TBitmap; Hist: THistogram; Channel: THistogramChannel; Tolerance: Card);
21844:   Proc AdjustBitmap2(Bmp: TBitmap; Low, High : Byte; Channel : THistogramChannel);
21845: end;
21846:
21847: Proc SRegister_cCipherRSA(CL: TPSPascalCompiler);
21848: begin
21849:   CL.AddClassN(CL.FindClass('TOBJECT'),'RSA;
21850:   TRSAPublicKey', 'record KeySize : Integer; Modulus : HugeWord; Exponent : HugeWord; end;
21851:   TRSAPrivateKey', 'record KeySize : Integer; Modulus : HugeWord; '
21852:   Exponent : HugeWord; PublicExponent : HugeWord; Primel : HugeWord; Prime2 '
21853:   : HugeWord; Phi: HugeWord; Exponent1: HugeWord; Exponent2: HugeWord; Coefficient: HugeWord; end;
21854:   CL.AddTypeS('TRSAEncryptionType', 'HugeWord;
21855:   TRSAEncryptionType', '( rsaetPKCS1, rsaetOAEP );
21856:   Proc RSAPublicKeyInit( var Key : TRSAPublicKey);
21857:   Proc RSAPublicKeyFinalise( var Key : TRSAPublicKey);
21858:   Proc RSAPublicKeyAssign( var KeyD : TRSAPublicKey; const KeyS : TRSAPublicKey);
21859:   Proc RSAPublicKeyAssignHex( var Key: TRSAPublicKey; const KeySize: Integer; const HexMod, HexExp: Str
21860:   ModBufSize: Integer; const ExpBuf: Str; const ExpBufSize: Integer; const ReverseByteOrder: Bool);
21861:   Proc RSAPublicKeyAssignBufStr( var Key: TRSAPublicKey; const KeySize: Integer; const ModBuf, ExpBuf: Ansistr);
21862:   Proc RSAPrivateKeyInit( var Key : TRSAPrivateKey);
21863:   Proc RSAPrivateKeyFinalise( var Key : TRSAPrivateKey);
21864:   Proc RSAPrivateKeyAssign( var KeyD : TRSAPrivateKey; const KeyS : TRSAPrivateKey);
21865:   Proc RSAPrivateKeyAssignHex( var Key: TRSAPrivateKey; const KeySize: Integer; const HexMod, HexExp: Ansistr);
21866:   Proc RSAPrivateKeyAssignBuf( var Key: TRSAPrivateKey; const KeySize: Integer; const ModBuf, ExpBuf: Str; const
   ModBufSize: Integer; const ExpBuf: Str; const ExpBufSize: Integer; const ReverseByteOrder: Boolean);
21867:   Proc RSAPrivateKeyAssignBufStr( var Key: TRSAPrivateKey; const KeySize: Integer; const ModBuf, ExpBuf: Ansistr);
21868:   Proc RSAGenerateKeys( const KeySize: Integer; var PrivateKey: TRSAPrivateKey; var PublicKey: TRSAPublicKey);
21869:   Func RSACipherMessageBufSize( const KeySize: Integer) : Integer;
21870:   Proc RSAEncodeMessagePKCS1( const KeySize: Integer; const Buf: Str; const BufSize: Integer; var EncodedMessage:
   TRSAMessage);
21871:   Proc RSAEncodeMessageOAEP( const KeySize: Integer; const Buf: Str; const BufSize: Integer; var EncodedMessage:
   TRSAMessage);
21872:   Proc RSAEncryptMessage( const PublicKey: TRSAPublicKey; const PlainMessage: TRSAMessage; var
   CipherMessage: TRSAMessage);
21873:   Func RSACipherMessageToBuf( const KeySize: Integer; const CipherMessage: TRSAMessage; var CipherBuf: Str; const
   CipherBufSize: Integer);
21874:   Func RSAEncryptC( const EncryptionType: TRSAEncryptionType; const PublicKey: TRSAPublicKey; const
   PlainBuf: Str; const PlainBufSize: Integer; var CipherBuf: Str; const CipherBufSize: Integer);
21875:   Func RSAEncryptStr( const EncryptionType: TRSAEncryptionType; const PublicKey: TRSAPublicKey; const
   Plain: Ansistr): Ansistr;
21876:   Proc RSACipherBufToMessage( const KeySize: Integer; const CipherBuf: Str; const CipherBufSize: Integer; var
   CipherMessage: TRSAMessage);
21877:   Proc RSADecryptMessage( const PrivateKey: TRSAPrivateKey; const CipherMessage: TRSAMessage; var
   EncodedMessage: TRSAMessage);
21878:   Func RSADecodeMessagePKCS1( const KeySize: Integer; const EncodedMessage: HugeWord; var Buf: Str; const BufSize:
   Integer): Integer;
21879:   Func RSADecodeMessageOAEP( const KeySize: Integer; const EncodedMessage: HugeWord; var Buf: Str; const BufSize:
   Integer): Integer;
21880:   Func RSADecryptC( const EncryptionType : TRSAEncryptionType; const PrivateKey : TRSAPrivateKey; const
   CipherBuf :Str; const CipherBufSize : Integer; var PlainBuf :Str; const PlainBufSize : Integer) : Integer;
21881:   Func RSADecryptStr( const EncryptionType: TRSAEncryptionType; const PrivateKey: TRSAPrivateKey; const
   Cipher: Ansistr): Ansistr;
21882:   Proc SelfTestRSA;
21883:   //CL.AddDelphiFunction('Proc Profile;
21884: end;
21885:
21886: // add to V 4.2.6.10

```

```

21887: Proc SIRegister_Streams(CL: TPSPascalCompiler);
21888: begin
21889:   SIRegister_TBaseStream(CL);
21890:   SIRegister_TFilterStream(CL);
21891:   SIRegister_TSeekableStream(CL);
21892:   //CL.AddTypeS('TFileAttributes2', 'set of ( sfaArchive, sfaHidden, sfaSystem, sf
21893:   //+'aReadOnly, sfaTemporary, sfaCompressed, sfaDirectory );
21894:   TFileAttribute2, (sfaArchive, sfaHidden, sfaSystem, sfaReadOnly, sfaTemporary, sfaCompressed, sfaDirectory);
21895:   TFileAttributes2, set of TFileAttribute2);
21896:   TfsMode, (sfsRead, sfsWrite, sfsCreate, sfsShareRead, sfsShareWrite);
21897:   TfsModes, 'set of TfsMode';
21898:   // CL.AddTypeS('TfsModes', 'set of( fsRead, fsWrite, fsCreate, fsShareRead, fsShareWrite );
21899:   fsReset, 'LongInt'.Value.ts32 := ord(fsRead) or ord(fsWrite);
21900:   fsShared, 'LongInt'.Value.ts32 := ord(fsShareRead) or ord(fsShareWrite);
21901:   SIRegister_TFileStream(CL);
21902:   SIRegister_TDumyStream(CL);
21903:   Proc CopyStream( Source, Dest: TBaseStream; const Size : Integer);
21904:   Func GetLastErrorText :Str';
21905: end;
21906:
21907: Proc SIRegister_CromisStreams(CL: TPSPascalCompiler);
21908: begin
21909:   Proc WriteToStreamAsString( const Stream : TStream; const Content :Str);
21910:   Proc WriteToStreamAsUnicode( const Stream : TStream; const Content :Str);
21911:   Proc WriteToStreamAsUTF80( const Stream : TStream; const Content :Str);
21912:   Proc WriteToStreamAsUTF81( const Stream : TStream; const Content: Ansistr);
21913:   Func ReadFromStreamAsUnicode( const Stream : TStream) :Str';
21914:   Func ReadFromStreamAsString( const Stream : TStream) :Str';
21915:   Func ReadFromStreamAsUTF8( const Stream : TStream) : Ansistr';
21916:   CL.AddConstantN('cDefaultSearchSize', 'LongInt').SetInt( 50);
21917:   SIRegister_TNamesEnumerator(CL);
21918:   SIRegister_IStorageStream(CL);
21919:   SIRegister_TStreamStorage(CL);
21920: end;
21921:
21922: // new functions on 4.2.6.10
21923: Func ArcTan3(const X : Extended) : Extended'; //bugfix of ArcTan()
21924: Func WMIConnect2(WBemLocator: ISWBemLocator; namespace, Server, account, password: str): ISWBemServices;
21925: Func WMIConnect3(WBemLocator: ISWBemLocator; Server, namespace, account, password: str): ISWBemServices;
21926: Func CLSIDFromProgID(pszProgID: widestring; out clsid: TGUID): Longint'; //stdcall;
21927: Func ProgIDFromCLSID(const clsid: TGUID; out pszProgID: widestring): Longint'; //stdcall;
21928: Func StringFromCLSID(const clsid: TGUID; out psz: widestring): longint'; //stdcall;
21929: //{$EXTERNALSYM CLSIDFromString}
21930: Func CLSIDFromString(psz: widestring; out clsid: TGUID): longint'; //stdcall;
21931: Func StringFromGUID2(const guid: TGUID; psz: widestring; cbMax: Integer): Integer'; //stdcall;
21932: Func ProgIDExists(const ProgID: WideString): Boolean';
21933: Func ByteArrayToString(const bin: array of byte): Str;
21934: Func ByteArrayToHex(const bin: array of byte): Str';
21935: Proc SaveFile(Contents : Ansistr; const FileName: Ansistr; Append :Bool);
21936: Proc SaveToFile(Contents: Ansistr; const FileName: Ansistr; Append :Bool);
21937: Func LoadFromFile( const FileName : Ansistr) : Ansistr';
21938: Func VariantIsObject(const V: Variant): Bool;
21939: Func DiskSpaceInfo(const Drive: str; out AvailBytes, TotalBytes, FreeBytes: Int64): Boolean;
21940: Func DriveDisplayName(const Drive: Str): Str;
21941: Proc DriveDisplayNames(const List: TStrings);
21942: Proc Log( const Line: Str; amemo: Tmemo);
21943: begin amemo.Lines.Add( Line) end;
21944: Proc LogFmtX( const Line: Str; amemo: Tmemo; const Args: array of const);
21945: begin
21946:   Log( Format( Line, Args), amemo)
21947: end;
21948:
21949: Proc SIRegister_USha256(CL: TPSPascalCompiler);
21950: begin
21951:   type TSHA256HASH, 'array[0..7] of Cardinal';
21952:   type TSHACHunk, 'array[0..7] of Cardinal';
21953:   //TSHA256HASH = array[0..7] of Cardinal;
21954:   //TChunk = array[0..15] of Cardinal;
21955:   Func CalcDoubleSHA256( Msg : Ansistr) : TSHA256HASH;
21956:   Func CalcSHA256( Msg : Ansistr) : TSHA256HASH;
21957:   Func CalcSHA2561( Stream : TStream) : TSHA256HASH;
21958:   Func SHA256ToStr( Hash : TSHA256HASH) : Str;
21959:   Func CanBeModifiedOnLastChunk(MessageTotalLength: Int64; var startBytePos: int): Bool;
21960:   Proc PascalCoinPrepareLastChunk(const messageToHash: Ansistr; var stateForLastChunk: TSHA256HASH; var
bufferForLastChunk : TSHACHunk);
21961:   Func ExecuteLastChunk(const stateForLastChunk: TSHA256HASH; const bufferForLastChunk:
TSHACHunk; nPos: Integer; nOnce, Timestamp: Card): TSHA256HASH';
21962:   Func ExecuteLastChunkAndDoSha256(const stateForLastChunk: TSHA256HASH; const
bufferForLastChunk: TSHACHunk; nPos: Integer; nOnce, Timestamp: Card): TSHA256HASH';
21963:   Proc PascalCoinExecuteLastChunkAndDoSha256(const stateForLastChunk: TSHA256HASH; const
bufferForLastChunk: TSHACHunk; nPos: Integer; nOnce, Timestamp: Card; var ResultSha256: AnsiStr);
21964:   Func Sha256HashToRaw(const hash : TSHA256HASH): Ansistr';
21965:   Func GetSHA256( Msg : Ansistr) : Str;
21966:   Func GetDriveNumber(const Drive: Str): Integer;
21967:   Func HardDiskSerial(const Drive: Str): DWORD;
21968:   Func IsDriveReady2(const Drive: Str): Bool;
21969:   Func Touchfile(const FileName: Str): Bool;
21970:   Func URLFromShortcut(const Shortcut: Str): Str;
21971:

```



```

21972: Func GetSHA256(Msg: Ansistr):Str; //overload;
21973: var Stream: TMemoryStream;
21974: begin
21975:   Stream:= TMemoryStream.Create;
21976:   try
21977:     Stream.WriteBuffer(PAnsiChar(Msg)^,Length(Msg));
21978:     Stream.Position:= 0;
21979:     Result:= SHA256ToStr(CalcSHA256(Stream));
21980:   finally
21981:     Stream.Free;
21982:   end;
21983: end;
21984:
21985: Code Example Binary file:
21986: sr:= filetostring(Exepath+'maxbox4.exe')
21987: writeln(SHA256ToStr(CalcSHA256(sr)))
21988: stringtofile(Exepath+'maxbox4.exe',sr,false); //append = false
21989:
21990: Proc SIRegister_uTPLb_BinaryUtils(CL: TPSPascalCompiler);
21991: begin
21992:   Func SwapEndien_u32(Value: uint32) : uint32;
21993:   Func SwapEndien_s64(Value: int64) : int64;
21994:   Func SwapEndien_u64(Value: uint64) : uint64;
21995:   Func RotateLeft1Bit_u32(Value : uint32) : uint32;
21996:   Proc Read_BigEndien_u32_Hex(const Value :Str; BinaryOut: TStream);
21997:   Func Get_TP_LockBox3_HINSTANCE : HMODULE;
21998: end;
21999:
22000: Proc SIRegister_Series(CL: TPSPascalCompiler);
22001: begin
22002:   CL.AddConstantN('PiDegree','Double').setExtended( Pi / 180.0);
22003:   Tee_CircledShadowColor,'LongWord'.SetUInt( TColor ( $A0A0A0 ));
22004:   CL.AddTypeS('TTreatNullsStyle','( teetnDontPaint, teetnSkip, teetnIgnore );
22005:   SIRegister_TCustomLineSeries(CL);
22006:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomSeries');
22007:   CL.AddTypeS(TSeriesClickPointerEvent,'Procedure(Sender:TCustomSeries;ValueIndex:Int;X,Y:Int;
22008:   TCustomSeriesStack','( cssNone, cssOverlap, cssStack, cssStack100 );
22009:   TOnGetPointerStyle,'Func ( Sender : TChartSeries; ValueIndex: Integer) : TSeriesPointerStyle');
22010:   SIRegister_TCustomSeries(CL);
22011:   SIRegister_TLineSeries(CL);
22012:   SIRegister_THorizLineSeries(CL);
22013:   SIRegister_TPointSeries(CL);
22014:   CL.AddTypeS(TMultiArea,'( mamaNone,maStacked, maStacked100 );
22015:   SIRegister_TAreaSeries(CL);
22016:   SIRegister_THorizAreaSeries(CL);
22017:   TMultiBar,(teembNone,teembSide,teembStacked,teembStacked100,teembSideAll,teembSelfStack);
22018:   CL.AddClassN(CL.FindClass('TOBJECT'),'TCustomBarSeries');
22019:   TBarStyle,(teebsRectangle, teebsPyramid, teebsInvPyramid, teebsCylinder,teebsEllipse, teebsArrow,
teebsRectGradient,teebsCone,teebsBevel,teebsSlantCube,teebsDiamond,teebsInvArrow, teebsInvCone);
22020:   TGetBarStyleEvent,'Procedure(Sender:TCustomBarSeries;ValueIndex:Integer;var TheBarStyle: TBarStyle);
22021:   SIRegister_TBarSeriesGradient(CL);
22022:   SIRegister_TCustomBarSeries(CL);
22023:   SIRegister_TBarSeries(CL);
22024:   SIRegister_THorizBarSeries(CL);
22025:   SIRegister_TCircledShadow(CL);
22026:   SIRegister_TCircledSeries(CL);
22027:   TPieAngle,'record StartAngle : Double; MidAngle : Double; EndAngle : Double; end';
22028:   TPieAngles,'array of TPieAngle';
22029:   SIRegister_TSliceValueList(CL);
22030:   TPieOtherStyle,'( piepoNone, piepoBelowPercent, piepoBelowValue );
22031:   SIRegister_TPieOtherSlice(CL);
22032:   SIRegister_TPieMarks(CL);
22033:   TMultiPie,'( teempAutomatic, teempDisabled );
22034:   SIRegister_TPieSeries(CL);
22035:   SIRegister_TFastLineSeries(CL);
22036:   //CL.AddConstantN('bsCylinder','').SetString( bsCylinder);
22037:   Proc RegisterTeeStandardSeries();
22038:   Proc TeePointerDrawLegend(Pointer:TSeriesPointer;AColor:TColor;const Rect:TRect;DrawPen:Bool;
AStyle:TSeriesPointerStyle);
22039: end;
22040:
22041: Proc SIRegister_UJSONFunctions(CL: TPSPascalCompiler);
22042: begin
22043:   CL.AddTypeS('TJSONValue','TJSONData');
22044:   SIRegister_TPCJSONData(CL);
22045:   //CL.AddTypeS('TPCJSONDataClass','class of TPCJSONData');
22046:   SIRegister_TPCJSONVariantValue(CL);
22047:   SIRegister_TPCJSONNameValue(CL);
22048:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPCJSONArray');
22049:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPCJSONObject');
22050:   SIRegister_TPCJSONList(CL);
22051:   SIRegister_TPCJSONArray(CL);
22052:   SIRegister_TPCJSONObject(CL);
22053:   CL.AddClassN(CL.FindClass(TOBJECT),EPCParametresError');
22054: end;
22055:
22056: Proc SIRegister_uTPLb_Hash(CL: TPSPascalCompiler);
22057: begin
22058:   CL.AddTypeS(TOnHashProgress,'Function(sender:TObject;CountBytesProcessed:int64):bool;

```

```

22059:   SIRegister_IHash(CL);
22060:   SIRegister_IHash_TestAccess(CL);
22061:   SIRegister_TSimpleHash(CL);
22062:   SIRegister_THash(CL);
22063: end;
22064:
22065: Proc SIRegister_UTime(CL: TPSPascalCompiler);
22066: begin
22067:   //Func TzSpecificLocalTimeToSystemTime( lpTimeZoneInformation:PTimeZoneInformation; var lpLocalTime,
22068:   lpUniversalTime : TSystemTime) : BOOL');
22069:   //Func SystemTimeToTzSpecificLocalTime( lpTimeZoneInformation:PTimeZoneInformation;var lpUniversalTime,
22070:   lpLocalTime : TSystemTime) : BOOL');
22071:   Func DateTime2UnivDateTime(d : TDateTime) : TDateTime';
22072:   Func UnivDateTime2LocalDateTime(d: TDateTime): TDateTime';
22073:   Func UnivDateTimeToUnix(dtDate: TDateTime : Longint');
22074:   Func UnixToUnivDateTime(USec: Longint): TDateTime';
22075:   Func UnixTimeToLocalElapsedTime( USec : Longint): Ansistr');
22076:   Func DateTimeElapsedTime(dtDate: TDateTime): Ansistr');
22077: end;
22078: Proc SIRegister_uTPLb_BlockCipher(CL: TPSPascalCompiler);
22079: begin
22080:   SIRegister_IBlockCodec(CL);
22081:   SIRegister_IBlockCipher(CL);
22082:   SIRegister_TBlockChainLink(CL);
22083:   CL.AddTypeS(TChainingFeature', '( cfNoNonce, cfKeyStream, cfAutoXOR, cf8bit);
22084:   CL.AddTypeS(TChainingFeatureSet', 'set of TChainingFeature');
22085:   SIRegister_IBlockChainingModel(CL);
22086:   SIRegister_IBlockCipherSelector(CL);
22087: end;
22088: Proc SIRegister_XMLDoc(CL: TPSPascalCompiler);
22089: begin
22090:   FindInterface('IUNKNOWN', IXMLNode, 'IXMLNode');
22091:   TOBJECT', 'TXMLNode');
22092:   TOBJECT', 'TXMLNodeList');
22093:   TOBJECT', 'TXMLNodeCollection');
22094:   TOBJECT', 'TXMLDocument');
22095:   CL.AddTypeS(TNodeListOperation', '( nlInsert, nlRemove, nlCreateNode );
22096:   CL.AddTypeS(TNodeListNotification', 'Proc ( Operation : TNodeListOperat
22097:   +ion; var Node : IXMLNode; const IndexOrName: OleVariant; BeforeOperation : Bool);
22098:   SIRegister_TXMLNodeList(CL);
22099:   CL.AddTypeS(TAsyncEventHandler, procedure(Sender: TObject; AsyncLoadState: Integer) of Object);
22100:   //CL.AddTypeS('TXMLNodeClass', 'class of TXMLNode');
22101:   TXMLNodeArray', 'array of TXMLNode');
22102:   TNodeClassInfo', 'record NodeName: DOMString; NamespaceURI: DOMString; NodeClass: TXMLNode; end');
22103:   CL.AddTypeS(TNodeClassArray', 'array of TNodeClassInfo');
22104:   //CL.AddTypeS('TXMLNodeCollectionClass', 'class of TXMLNodeCollection');
22105:   CL.AddTypeS(TNodeChange', '( ncUpdateValue, ncInsertChild, ncRemoveChild, nc'
22106:   +AddAttribute, ncRemoveAttribute );
22107:   SIRegister_IXMLNodeAccess(CL);
22108:   SIRegister_TXMLNode(CL);
22109:   SIRegister_TXMLNodeCollection(CL);
22110:   TNodeChangeEvent', 'Proc ( const Node : IXMLNode; ChangeType : TNodeChange);
22111:   XMLPrologItem', '( xmlpVersion, xmlpEncoding, xmlpStandalone );
22112:   TXMLDocumentSource', '( xdsNone, xdsXMLProperty, xdsXMLData, xdsFile, xdsStream);
22113:   SIRegister_IXMLDocumentAccess(CL);
22114:   SIRegister_TXMLDocument(CL);
22115:   Func CreateDOMNode(Doc: IDOMDocument; const NameOrData: DOMString; NodeType: TNodeType; const
22116:   AddlData: DOMString): IDOMNode;
22117:   Func DetectCharEncoding( S : TStream) : TXmlEncodingType');
22118:   Proc CheckEncoding(var XMLData: DOMString; const ValidEncodings: array of string);
22119:   Func XMLStringToWideString( const XMLString :Str) : WideString;
22120:   Func FormatXMLData( const XMLData : DOMString) : DOMString;
22121:   Func LoadXMLDocument3( const FileName : DOMString) : IXMLDocument';
22122:   Func LoadXMLData( const XMLData : DOMString) : IXMLDocument;
22123:   Func LoadXMLData2( const XMLData :Str) : IXMLDocument;
22124:   Func NewXMLDocument( Version : DOMString) : IXMLDocument';
22125:   Proc XMLDocError( const Msg :Str);
22126:   Proc XMLDocError2( const Msg :Str; const Args : array of const);
22127:   Func CreateXMLDocument();
22128: end;
22129: Proc SIRegister_uTPLb_Codec(CL: TPSPascalCompiler);
22130: begin
22131:   SIRegister_TSimpleCodec(CL);
22132:   SIRegister_ICodec_TestAccess(CL);
22133:   SIRegister_TCodec(CL);
22134: end;
22135: Proc SIRegister_uTPLb_StreamCipher(CL: TPSPascalCompiler);
22136: begin
22137:   CL.AddTypeS(TAlgorithmicFeature', '( afStar, afCryptographicallyWeak, afNotI'
22138:   +mplementedYet, afForTestOnly, afForRunTimeOnly, afEncumberedByPatent, afEn'
22139:   +cumberedByCopyRight, afOpenSourceSoftware, afCommercial, afCompressor, afC'
22140:   +onverter, afBlockAdapter, afDisplayNameOnKeySize, afDoesNotNeedSalt, afAsymmetric );
22141:   CL.AddTypeS(TAlgorithmicFeatureSet', 'set of TAlgorithmicFeature');
22142:   SIRegister_ICryptoGraphicAlgorithm(CL);
22143:   SIRegister_TSymetricKey(CL);
22144: end;

```

```

22145:   SIRegister_IStreamEncryptor(CL);
22146:   SIRegister_IStreamDecryptor(CL);
22147:   SIRegister_IStreamCipher(CL);
22148:   SIRegister_IIsBase64Converter(CL);
22149: end;
22150:
22151: Proc SIRegister_MidasCon(CL: TPSPascalCompiler);
22152: begin
22153:   SIRegister_TRemoteServer(CL);
22154:   CL.AddTypeS(TConnectType, '( ctDCOM, ctSockets, ctOLEEnterprise );
22155:   SIRegister_TMidasConnection(CL);
22156: end;
22157:
22158: Proc SIRegister_IXmlDocument2(CL: TPSPascalCompiler);
22159: begin
22160:   //with RegInterfaceS(CL, 'IXmlNode', 'IXmlDocument') do
22161:   with CL.AddInterface(CL.FindInterface('IXmlBase'), IXmlNode2, 'IXmlDocument2') do begin
22162:     Func Get_DocumentElement : IXmlElement, cdRegister);
22163:     Func Get_BinaryXML : RawByteString, r);
22164:     Func Get_PreserveWhiteSpace : Bool, );
22165:     Proc Set_PreserveWhiteSpace( aValue : Bool), );
22166:     Func Get_OnTagEnd : THookTag, cdRegister);
22167:     Proc Set_OnTagEnd( aValue : THookTag), cdRegister);
22168:     Func Get_OnTagBegin : THookTag, cdRegister);
22169:     Proc Set_OnTagBegin( aValue : THookTag), cdRegister);
22170:     Func NewDocument(const aValue, anEncoding: TXmlString; aRootElementNameID: NativeInt): IXmlElement;
22171:     Func NewDocument1(const aValue, anEncoding, aRootElementName: TXmlString): IXmlElement;
22172:     Func CreateElement( aNameID : NativeInt) : IXmlElement, ');
22173:     Func CreateElement1(const aName : TXmlString) : IXmlElement, ');
22174:     Func CreateText(const aData : TXmlString) : IXmlText, ');
22175:     Func CreateCDATASection( const aData : TXmlString) : IXmlCDATASection, ');
22176:     Func CreateComment( const aData : TXmlString) : IXmlComment, ');
22177:     Func CreateProcessingInstruction(const aTarget, aData: TXmlString): IXmlProcessingInstruction, ');
22178:     Func CreateProcessingInstruction1(aTargetID: NativeInt, const aData: TXmlString): IXmlProcessingInstruction);
22179:     Proc LoadXML( const aXML : RawByteString), ');
22180:     Proc LoadBinaryXML( const aXML : RawByteString), ');
22181:     Proc Load( aStream : TStream), ');
22182:     Proc Load1( const aFileName : Str), ');
22183:     Proc LoadResource( aType, aName : PChar), ');
22184:     Proc Save( aStream : TStream), ');
22185:     Proc Save1( const aFileName : Str), ');
22186:     Proc SaveBinary( aStream : TStream; anOptions : LongWord), ');
22187:     Proc SaveBinary1( const aFileName : Str; anOptions : LongWord), ');
22188:   end;
22189: end;
22190:
22191: Proc SIRegister_OmnixXMLUtils(CL: TPSPascalCompiler);
22192: begin
22193:   // XmlString = WideString;
22194:   CL.AddTypeS('XmlString', 'WideString');
22195:   Func XMLStrToReal38( nodeValue : XmlString; var value : real) : Bool;
22196:   Func XMLStrToReal39( nodeValue : XmlString) : real;
22197:   Func XMLStrToRealDef( nodeValue : XmlString; defaultValue : real) : real);
22198:   Func XMLStrToExtended40( nodeValue : XmlString; var value : extended): Bool;
22199:   Func XMLStrToExtended41( nodeValue : XmlString) : extended;
22200:   Func XMLStrToExtendedDef( nodeValue : XmlString; defaultValue : extended): extended;
22201:   Func XMLStrToCurrency42( nodeValue : XmlString; var value : Currency): Bool;
22202:   Func XMLStrToCurrency43( nodeValue : XmlString) : Currency;
22203:   Func XMLStrToCurrencyDef( nodeValue : XmlString; defaultValue : Currency): Currency;
22204:   Func XMLStrToInt44( nodeValue : XmlString; var value : integer) : Bool;
22205:   Func XMLStrToInt45( nodeValue : XmlString) : integer;
22206:   Func XMLStrToIntDef( nodeValue : XmlString; defaultValue : integer) : integer);
22207:   Func XMLStrToInt6446( nodeValue : XmlString; var value : int64) : Bool;
22208:   Func XMLStrToInt6447( nodeValue : XmlString) : int64;
22209:   Func XMLStrToInt64Def( nodeValue : XmlString; defaultValue : int64) : int64);
22210:   Func XMLStrToBool48( nodeValue : XmlString; var value : Bool) : Bool;
22211:   Func XMLStrToBool49( nodeValue : XmlString) : Bool;
22212:   Func XMLStrToBoolDef( nodeValue : XmlString; defaultValue : Bool): Bool;
22213:   Func XMLStrToDateTime50( nodeValue : XmlString; var value : TDateTime): Bool;
22214:   Func XMLStrToDateTime51( nodeValue : XmlString) : TDateTime;
22215:   Func XMLStrToDateTimeDef( nodeValue : XmlString; defaultValue : TDateTime): TDateTime;
22216:   Func XMLStrToDate52( nodeValue : XmlString; var value : TDateTime): Bool;
22217:   Func XMLStrToDate53( nodeValue : XmlString) : TDateTime;
22218:   Func XMLStrToDateDef( nodeValue : XmlString; defaultValue : TDateTime) : TDateTime;
22219:   Func XMLStrToTime54( nodeValue : XmlString; var value : TDateTime): Bool;
22220:   Func XMLStrToTime55( nodeValue : XmlString) : TDateTime;
22221:   Func XMLStrToTimeDef( nodeValue : XmlString; defaultValue : TDateTime): TDateTime;
22222:   Func XMLStrToBinary( nodeValue : XmlString; const value : TStream): Bool);
22223:   Func XMLRealToStr( value : real; precision : byte) : XmlString);
22224:   Func XMLExtendedToStr( value : extended) : XmlString);
22225:   Func XMLCurrencyToStr( value : Currency) : XmlString);
22226:   Func XMLIntToStr( value : integer) : XmlString);
22227:   Func XMLInt64ToStr( value : int64) : XmlString);
22228:   Func XMLBoolToStr( value : Bool; useBoolStrs : Bool) : XmlString);
22229:   Func XMLDateTimeToStr2( value : TDateTime) : XmlString);
22230:   Func XMLDateTimeToStrEx( value : TDateTime) : XmlString);
22231:   Func XMLDateToStr2( value : TDateTime) : XmlString);
22232:   Func XMLTimeToStr( value : TDateTime) : XmlString);
22233:   Func XMLBinaryToStr( value : TStream) : XmlString);

```

```

22234: Func XMLVariantToStr( value : Variant) : XmlString');
22235: end;
22236:
22237: Proc SIRegister_IdCoderHeader(CL: TPSPascalCompiler);
22238: begin
22239:   CL.AddTypeS('TTransfer', ( bit7, bit8, iso2022jp );
22240:   Func EncodeAddressItem(EmailAddr:TIdEmailAddressItem; const HeaderEncoding: Char;
22241:   TransferHeader: TTransfer; MimeCharSet:Str):Str;
22242:   Func EncodeHeader(const Header:Str; specials : CSET2; const HeaderEncoding: Char;
22243:   TransferHeader: TTransfer; MimeCharSet:Str):Str;
22244:   Func Encode2022JP(const S:Str):Str;
22245:   Func EncodeAddress(EmailAddr:TIdEmailAddressList;const HeaderEncoding:Char;
22246:   TransferHeader: TTransfer; MimeCharSet:Str):Str;
22247:   Func DecodeHeader(Header:Str):str;
22248:   Func Decode2022JP(const S:Str):Str;
22249:   Proc DecodeAddress(EMailAddr : TIdEmailAddressItem);
22250:   Proc DecodeAddresses(AEMails :Str; EMailAddr : TIdEmailAddressList);
22251:   Proc InitializeISO(var TransferHeader:TTransfer;var HeaderEncoding:char;var CharSet:str);
22252: end;
22253:
22254: Proc SIRegister_FannNetwork(CL: TPSPascalCompiler);
22255: begin
22256:   // CL.AddTypeS('PCardinalArray', '^CardinalArray // will not work');
22257:   CL.AddConstantN('DLL_FILE','String').SetString( 'fannfloat.dll');
22258:   CL.AddTypeS('fann_type', 'single');
22259:   Fann_Type_Array', 'array [0..65535] of fann_type');
22260:   TFann_Type_Array', 'array [0..65535] of fann_type');
22261:   TFann_Type_Array2', 'array of fann_type');
22262:   TFann_Type_Array3', 'array of single');
22263:   TTrainingAlgorithm', '( taFANN_TRAIN_INCREMENTAL, taFANN_TRAIN_B'
22264:   +ATC, taFANN_TRAIN_RPROP, taFANN_TRAIN_QUICKPROP );
22265:   CL.AddTypeS(TActivationFunction', '( affANN_LINEAR, affANN_THRESHOLD, affANN'
22266:   +'_THRESHOLD_SYMMETRIC, affANN_SIGMOID, affANN_SIGMOID_STEPWISE, affANN_SIGM'
22267:   +'_OID_SYMMETRIC, affANN_SIGMOID_SYMMETRIC_STEPWISE, affANN_GAUSSIAN, affANN_'
22268:   +'_GAUSSIAN_SYMMETRIC, affANN_GAUSSIAN_STEPWISE, affANN_ELLIOT, affANN_ELLIOT'
22269:   +'_SYMMETRIC, affANN_LINEAR_PIECE, affANN_LINEAR_PIECE_SYMMETRIC, affANN_SIN'
22270:   +'_SYMMETRIC, affANN_COS_SYMMETRIC, affANN_SIN, affANN_COS );
22271:   SIRegister_TFannNetwork(CL); //CL.AddDelphiFunction('Proc Register');
22272: end;
22273:
22274: Proc SIRegister_TFannNetwork(CL: TPSPascalCompiler);
22275: begin
22276:   //with RegClassS(CL,'TComponent', 'TFannNetwork') do
22277:   with CL.AddClassN(CL.FindClass('TComponent'),'TFannNetwork') do begin
22278:     Constructor Create( Aowner : TComponent);
22279:     Proc Build( );
22280:     Proc Free';
22281:     Proc UnBuild( );
22282:     Func Train( Input : array of fann_type; Output : array of fann_type) : single');
22283:     Proc TrainOnFile(FileName:str; MaxEpochs:Card;DesiredError: Single);
22284:     Proc Run(Inputs: array of fann_type; var Outputs : TFann_Type_Array2);
22285:     Proc Run2(Inputs: array of fann_type; var Outputs : array of fann_type);
22286:     Proc Run3( Inputs : array of fann_type; var Outputs : TFann_Type_Array3);
22287:     Proc Run4( Inputs : array of fann_type; var Outputs : TFann_Type_Array3);
22288:     Proc SaveToFile( FileName :Str);
22289:     Proc LoadFromFile( Filename :Str);
22290:     FannObject', 'PFann', iptr);
22291:     Layers', 'TStrings', iptrw);
22292:     LearningRate', 'Single', iptrw);
22293:     ConnectionRate', 'Single', iptrw);
22294:     LearningMometum', 'single', iptrw);
22295:     MSE', 'Single', iptr);
22296:     TrainingAlgorithm', 'TTrainingAlgorithm', iptrw);
22297:     ActivationFunctionHidden', 'TActivationFunction', iptrw);
22298:     ActivationFunctionOutput', 'TActivationFunction', iptrw);
22299:   end;
22300: end;
22301: TFannNetwork encapsulates the Fast Artificial Neural Network. fann.sourceforge.net
22302:
22303: Proc SIRegister_RTLDatetimeplus(CL: TPSPascalCompiler);
22304: begin
22305:   SIRegister_TNullableDateTime(CL);
22306:   Func IsValidDate2( const Value :Str; out ADate : TDateTime) :Bool;
22307:   Func IsValidTime2( const Value :Str; out ATime : TDateTime) :Bool;
22308:   Func IsValidDateTime2( const Value :Str; out ADateTime : TDateTime) :Bool;
22309:   Proc ChangeSystemTime( const Value : TDateTime);
22310:   Func TryChangeSystemTime( const Value : TDateTime) :Bool');
22311: end;
22312:
22313: // new functions on V 4.2.8.10 III /IV - V 4.5.8.10 IV
22314: Func IsUTF8File(const FileName:Str):Bool;
22315: Func IsUTF16File(const FileName:Str):Bool;
22316: Proc RichEditToCanvas(RichEdit: TRichEdit; Canvas: TCanvas; PixelsPerInch:Integer);
22317: --> RichEditToCanvas(RichEdit1, Image1.Canvas, Self.PixelsPerInch);
22318: Func {$I ActiveDEP:Card; }
22319: Func {$I ActivateDEP:Card;}
22320: Proc WriteXMLFile(doc: TXMLDocument; const AFileName:Str);
22321: Proc WriteXML(Element: TDOMNode; const AFileName:Str);
22322: Func ReadXMLFile(const AFileName:Str) : TXMLDocument;

```

```

22323: Proc WriteXMLFile(doc: TXMLDocumentDOM; const AFileName:Str);
22324: Proc WriteXML(Element: TDOMNode; const AFileName:Str);
22325: Proc WriteXMLFileStream(ADoc : TXMLDocument; AStream : TStream);
22326:
22327: Func CountColors(const Bitmap: Graphics.TBitmap): Integer;
22328: var Flags: array[Byte, Byte] of Classes.TBits;
22329:   I, J, K: Integer;
22330:   RowIn: PRGBTripleArray;
22331: begin
22332:   // Be sure bitmap is 24-bits/pixel
22333:   Assert(Bitmap.PixelFormat = Graphics.pf24Bit);
22334:
22335: Func SndPlaySound(const AFilename: pchar; flags:Card):Bool;
22336: Proc PlaySound1(const AFilename:Str);
22337: Func mciSendCommand(mciID, umessage, dwparam1, dwparam2:Card):Card;
22338: Func ProxyActive:Bool;
22339: Func GetProxyServer:Str;
22340: Func GetProxyOverride:Str;
22341: Func GetProxyServerIP:Str;
22342: Func GetProxyServerPort: Integer;
22343: Func GetHtml(const Url :Str) :Str;
22344: Func GetHtml2(const Url, UserAgent :Str) :Str;
22345: Proc FormatXMLFile(const XmlFile:Str);
22346: Func FormatXMLFile2String(const XmlFile:Str):Str;
22347: Func selectXMLNode(xnRoot: IXmlNode; const nodePath: WideString): IXmlNode;
22348: Func HTMLEncode3(const Data:Str):Str; //vfast
22349: Func IsSpecialFolderSupported(CSIDL: Integer):Bool;
22350: Func SpecialFolderPath(CSIDL: Integer):Str;
22351: Func DataSetToXML(const ADataSet: TDataSet):Str;
22352: Func PackageParams(Params : TParams; Types : TParamTypes) : OleVariant';
22353: Proc UnpackParams(const Source : OleVariant; Dest : TParams);
22354: Proc ActivateDEP2;
22355: Func JavaToDelphiDateTime(const dt: int64): TDateTime;
22356: Func DelphiToJavaDateTime(const dt: TDateTime): int64;
22357: Func GetTimeBias: integer;
22358: Proc WrapLines(var a, b:Str; len: Integer);
22359: Func DelphiDateTimeToISO8601Date(dt: TDateTime):Str;
22360: Proc BackgroundWorkerWaitFor(isworking:Bool; fWindow: HWND);
22361:
22362: uPSI_CromisStreams,
22363: uPSI_Streams,
22364: uPSI_BitStream,
22365: uPSI_UJSONFunctions.pas
22366: uPSI_uTPLb_BinaryUtils.pas
22367: uPSI_USha256.pas
22368: uPSI_uTPLb_HashDsc.pas
22369: uPSI_uTPLb_Hash.pas
22370: SIRegister_Series(X); //4.2.6.10
22371: uPSI_Series,
22372: unit uPSI_mimeinln;
22373: unit uPSI_UTime; //unit UTime;
22374: RIRegister_TStreamStorage(X);
22375: RIRegister_TNamesEnumerator(X);
22376: 1283 uPSI_uTPLb_StreamCipher.pas
22377: 1284 uPSI_uTPLb_BlockCipher.pas
22378: 1285 uPSI_uTPLb_Asymetric.pas
22379: 1286 uPSI_uTPLb_CodecIntf.pas
22380: 1287 uPSI_uTPLb_Codec.pas
22381: 1288 uPSI_ADOInt.pas
22382: 1289 uPSI_MidasCon.pas
22383: 1290 uPSI_XMLDoc.pas
22384: 1291 uPSI_XMLIntf.pas
22385: 1292 uPSI_ProxyUtils.pas
22386: 1293 unit uPSI_maxXMLUtils2;
22387: 1294 unit StDict_Routines(S: TPSExec);
22388: 1295 unit uPSI_Hashes2
22389: 1296 unit uPSI_IdCoderHeader;
22390:
22391: Func TRestRequest_createStringStreamFromStringList(strings:TStringList):TStringStream;
22392: {A simple Oscilloscope using TWaveIn class.
22393: More info at http://www.delphi4fun.org/programs/oscilloscope.htm }
22394: http://www.retroarchive.org/garbo/pc/turbopas/index.html
22395: uses Forms,
22396:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
22397:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
22398:   uColorFunctions in 'uColorFunctions.pas',
22399:   AMixer in 'AMixer.pas',
22400:   uSettings in 'uSettings.pas',
22401:   UWavein4 in 'UWavein4.pas',
22402:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
22403:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
22404: BSpectrum in unit uPSI_BSpectrum; {SpectralLibrary}
22405: 1250 unit uPSI_U_Oscilloscope4_2
22406: 1231 FormIpac: TFormIpac;
22407:
22408: // new functions on V 4.6.3.10
22409: Proc SIRegister_cHash(CL: TObject);
22410: var sha: TSha256hash2;
22411: begin

```



```

22412: DigestToHexBuf( const Digest :Str; const Size : Integer; const Buf);
22413: DigestToHex( const Digest :Str; const Size : Integer) : Ansistr);
22414: Digest128Equal( const Digest1, Digest2 : T128BitDigest) :Bool);
22415: Digest160Equal( const Digest1, Digest2 : T160BitDigest) :Bool);
22416: Digest224Equal( const Digest1, Digest2 : T224BitDigest) :Bool);
22417: Digest256Equal( const Digest1, Digest2 : T256BitDigest) :Bool);
22418: Digest384Equal( const Digest1, Digest2 : T384BitDigest) :Bool);
22419: Digest512Equal( const Digest1, Digest2 : T512BitDigest) :Bool);
22420: hashNoError('LongInt').SetInt( 0);
22421: hashInternalError('LongInt').SetInt( 1);
22422: hashInvalidHashType('LongInt').SetInt( 2);
22423: hashInvalidBuffer('LongInt').SetInt( 3);
22424: hashInvalidBufferSize('LongInt').SetInt( 4);
22425: hashInvalidDigest('LongInt').SetInt( 5);
22426: hashInvalidKey('LongInt').SetInt( 6);
22427: hashInvalidFileName('LongInt').SetInt( 7);
22428: hashFileOpenError('LongInt').SetInt( 8);
22429: hashFileSeekError('LongInt').SetInt( 9);
22430: hashFileReadError('LongInt').SetInt( 10);
22431: hashNotKeyedHashType('LongInt').SetInt( 11);
22432: hashTooManyOpenHandles('LongInt').SetInt( 12);
22433: hashInvalidHandle('LongInt').SetInt( 13);
22434: hashMAX_ERROR('LongInt').SetInt( 13);
22435: GetHashErrorMessage( const ErrorCode : LongWord) : PChar);
22436: /SIRegister_EHashError(CL);
22437: SecureClear( var Buf :Str; const BufSize : Integer);
22438: SecureClear512( var Buf : T512BitBuf);
22439: SecureClear1024( var Buf : T1024BitBuf);
22440: SecureClearStr2( var S : Ansistr); -----
22441: CalcChecksum32( const Buf : Ansistr) : LongWord);
22442: CalcXOR8( const Buf : Ansistr) : Byte);
22443: CalcXOR16( const Buf : Ansistr) : Word);
22444: CalcXOR32( const Buf : Ansistr) : LongWord);
22445: CRC16Init( var CRC16 : Word);
22446: CRC16Byte( const CRC16 : Word; const Octet : Byte) : Word);
22447: CRC16Buf( const CRC16 : Word; const Buf :Str; const BufSize : Integer) : Word);
22448: CalcCRC16( const Buf : Ansistr) : Word);
22449: SetCRC32Poly( const Poly : LongWord);
22450: CRC32Init( var CRC32 : LongWord);
22451: CRC32Byte( const CRC32 : LongWord; const Octet : Byte) : LongWord);
22452: CRC32Buf( const CRC32 : LongWord; const Buf :Str; const BufSize : Integer) : LongWord);
22453: CRC32BufNoCase(const CRC32: LongWord; const Buf:Str; const BufSize: Integer) : LongWord);
22454: CalcCRC32( const Buf : Ansistr) : LongWord);
22455: Adler32Init( var Adler32 : LongWord);
22456: Adler32Byte( const Adler32 : LongWord; const Octet : Byte) : LongWord);
22457: Adler32Buf( const Adler32:LongWord;const Buf:str; const BufSize:Integer) : LongWord);
22458: CalcAdler32( const Buf : Ansistr) : LongWord);
22459: ELFInit( var Digest : LongWord);
22460: ELFBuf( const Digest : LongWord; const Buf :Str; const BufSize : Integer):LongWord);
22461: CalcELF( const Buf : Ansistr) : LongWord);
22462: IsValidISBN2( const S : Ansistr) :Bool); -----
22463: IsValidLUHN( const S : Ansistr) :Bool);
22464: KnuthHash( const S : Ansistr) : LongWord);
22465: MD5InitDigest( var Digest : T128BitDigest);
22466: MD5Buf( var Digest : T128BitDigest; const Buf :Str; const BufSize : Integer);
22467: MD5FinalBuf(var Digest:T128BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22468: CalcMD5( const Buf : Ansistr) : T128BitDigest);
22469: MD5DigestAsString( const Digest : T128BitDigest) : Ansistr);
22470: MD5DigestToHex( const Digest : T128BitDigest) : Ansistr);
22471: SHA1InitDigest( var Digest : T160BitDigest);
22472: SHA1Buf( var Digest : T160BitDigest; const Buf :Str; const BufSize : Integer);
22473: SHA1FinalBuf(var Digest:T160BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64);
22474: CalcSHA1( const Buf : Ansistr) : T160BitDigest);
22475: SHA1DigestAsString( const Digest : T160BitDigest) : Ansistr);
22476: SHA1DigestToHex( const Digest : T160BitDigest) : Ansistr);
22477: SHA224InitDigest( var Digest : T256BitDigest);
22478: SHA224Buf( var Digest : T256BitDigest; const Buf:Str;const BufSize : Integer);
22479: SHA224FinalBuf(var Digest:T256BitDigest;const Buf:str;const BufSize:Integer;const TotalSize:Int64;var
OutDigest:T224BitDigest);
22480: CalcSHA224( const Buf : Ansistr) : T224BitDigest);
22481: SHA224DigestAsString( const Digest : T224BitDigest) : Ansistr);
22482: SHA224DigestToHex( const Digest : T224BitDigest) : Ansistr);
22483: SHA256InitDigest( var Digest : T256BitDigest);
22484: SHA256Buf( var Digest : T256BitDigest; const Buf :Str; const BufSize : Integer);
22485: SHA256FinalBuf(var Digest:T256BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64;
22486: CalcSHA2562( const Buf : Ansistr) : T256BitDigest); -----
22487: SHA256DigestAsString( const Digest : T256BitDigest) : Ansistr);
22488: SHA256DigestToHex( const Digest : T256BitDigest) : Ansistr);
22489: SHA384InitDigest( var Digest : T512BitDigest);
22490: SHA384Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22491: SHA384FinalBuf(var Digest:T512BitDigest;const Buf:str;const BufSize:Int;const TotalSize:Int64;var
OutDigest:T384BitDigest);
22492: CalcSHA384( const Buf : Ansistr) : T384BitDigest);
22493: SHA384DigestAsString( const Digest : T384BitDigest) : Ansistr);
22494: SHA384DigestToHex( const Digest : T384BitDigest) : Ansistr);
22495: SHA512InitDigest( var Digest : T512BitDigest);
22496: SHA512Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22497: SHA512FinalBuf(var Digest:T512BitDigest;const Buf:str;const BufSize:Integer;const TotalSize:Int64);
22498: CalcSHA512( const Buf : Ansistr) : T512BitDigest);

```

```

22499: SHA512DigestAsString( const Digest : T512BitDigest) : Ansistr');
22500: SHA512DigestToHex( const Digest : T512BitDigest) : Ansistr');
22501: HMAC_MD5Init(const Key:str;const KeySize:Integer;var Digest:T128BitDigest;var K:T512BitBuf);
22502: HMAC_MD5Buf( var Digest : T128BitDigest; const Buf :Str; const BufSize : Integer);
22503: HMAC_MD5FinalBuf(const K:T512BitBuf;var Digest T128BitDigest;const Buf:str;const BufSize:Integer;const
TotalSize:Int64);
22504: CalcHMAC_MD5( const Key, Buf : Ansistr) : T128BitDigest');
22505: HMAC_SHA1Init(const Key:str;const KeySize:Int;var Digest:T160BitDigest;var K:T512BitBuf);
22506: HMAC_SHA1Buf( var Digest : T160BitDigest; const Buf :Str; const BufSize : Integer);
22507: HMAC_SHA1FinalBuf(const K:T512BitBuf;ar Digest:T160BitDigest;const Buf:Str;const BufSize:Int;const
TotalSize Int64);
22508: CalcHMAC_SHA1( const Key, Buf : Ansistr):T160BitDigest;
22509: HMAC_SHA256Init(const Key:str;const KeySize:Int;var Digest:T256BitDigest;var K:T512BitBuf);
22510: HMAC_SHA256Buf( var Digest : T256BitDigest; const Buf :Str; const BufSize : Integer);
22511: HMAC_SHA256FinalBuf(const K :T512BitBuf;var Digest: T256BitDigest;const Buf:Str; const
BufSize:Integer;const TotalSize:Int64);
22512: CalcHMAC_SHA256( const Key, Buf : Ansistr) : T256BitDigest');
22513: HMAC_SHA512Init(const Key:str;const KeySize:Int;var Digest:T512BitDigest;var K:T1024BitBuf);
22514: HMAC_SHA512Buf( var Digest : T512BitDigest; const Buf :Str; const BufSize : Integer);
22515: HMAC_SHA512FinalBuf(const K:T1024BitBuf;var Digest:T512BitDigest;const Buf:str;const
BufSize:Integer;const TotalSize:Int64);
22516: CalcHMAC_SHA512( const Key, Buf: Ansistr): T512BitDigest');
22517: AHash(CL);
22518: CL.AddTypeS('THashClass', 'class of AHash');
22519: TChecksum32Hash(CL);
22520: TXOR8Hash(CL);
22521: TXOR16Hash(CL);
22522: TXOR32Hash(CL);
22523: TCRC16Hash(CL);
22524: TCRC32Hash(CL);
22525: TAdler32Hash(CL);
22526: TELFHash(CL);
22527: TMD5Hash(CL);
22528: TSHA1Hash(CL);
22529: TSHA256Hash2(CL); //-----!!
22530: TSHA512Hash(CL);
22531: THMAC_MD5Hash(CL);
22532: THMAC_SHA1Hash(CL);
22533: THMAC_SHA256Hash(CL);
22534: THMAC_SHA512Hash(CL);
22535: THashType', ('hashChecksum32,hashXOR8,hashXOR16,hashXOR32,hashCRC16,hashCRC32,hashAdler32, hashELF,
hashMD5,hashSHA1 hashSHA256,hashSHA512,hashHMAC_MD5,hashHMAC_SHA1,hashHMAC_SHA256,hashHMAC_SHA512'+');
22536: GetHashClassByType( const HashType : THashType) : THashClass');
22537: GetDigestSize( const HashType : THashType) : Integer');
22538: CalculateHash(const HashType:THashType;const Buf:AnsiStr;const Digest:str;const Key:AnsiStr;
22539: HashString(const S: Ansistr; const Slots:LongWord; const CaseSensitive:Boolean):LongWord');
22540: SelfTestHash');
22541: end;
22542:
22543: Unit UMatrix
22544: Proc Determinant( Dimen : integer; Data : TNmatrix; var Det : Float; var Error : byte);
22545: Proc Inverse2( Dimen : integer; Data : TNmatrix; var Inv : TNmatrix; var Error : byte);
22546: Proc Gaussian_Elimination(Dimen:integer;Coefficients:TNmatrix;Constants:TNvector;var
Solution:TNvector;var Error:byte);
22547: Proc Partial_Pivoting(Dimen:integer;Coefficients:TNmatrix;Constants:TNvector;var Solution:TNvector;var
Error:byte);
22548: Proc LU_Decompose(Dimen:integer;Coefficients:TNmatrix;var Decomp:TNmatrix;var Permute:TNmatrix;var
Error:byte);
22549: Proc LU_Solve(Dimen:integer;var Decomp TNmatrix;Constants:TNvector;var Permute:TNmatrix;var
Solution:TNvector;var Error:byte);
22550: Proc Gauss_Seidel(Dimen:integer;Coefficients:TNmatrix;Constants:TNvector;Tol:Float;MaxIter:integer;var
Solution:TNvector;var Iter:integer;var Error:byte);
22551: end;
22552:
22553: Proc SIRegister_uTPlb_RSA_Primitives(CL: TPSPascalCompiler);
22554: begin
22555: //CL.AddDelphiFunction('Func I2OSP( x : THugeCardinal; xLen : integer; XStream : TStream; Pool :
TMemoryStreamPool) :Bool');
22556: //Func OS2IP( XStream : TStream; xLen : integer; var x : THugeCardinal; Pool : TMemoryStreamPool;
MaxBits : integer) :Bool');
22557: Proc MGF1( mgfSeed : TStream; maskLen :Card; mask : TStream);
22558: CL.AddTypeS('TLongOpResult', '( opPass, opFail, opAbort );
22559: Func RSAES_OAEP_ENCRYPT( n, e : THugeCardinal; M, C : TMemoryStream) :Bool');
22560: Func RSAES_OAEP_ENCRYPT_MaxByteLen( n : THugeCardinal) : integer');
22561: Func RSAES_OAEP_DECRYPT( d, n : THugeCardinal; C, M : TStream) :Bool');
22562: Func EMSA_PSS_ENCODE(M:TStream;emBits,
sLen:integer;EM:TStream;CheckAbortFunc:TOnHashProgress):TLongOpResult;
22563: Func RSASSA_PSS_SIGN(d,n:THugeCardinal;M,S:TStream;CheckAbortFunc:TOnHashProgress): TLongOpResult;
22564: Func EMSA_PSS_VERIFY(M:TStream;emBits,sLen integer;EM:TStream;
CheckAbortFunc:TOnHashProgress):TLongOpResult;
22565: Func RSASSA_PSS_VERIFY(n,e THugeCardinal; M, S : TStream; CheckAbortFunc : TOnHashProgress) :
TLongOpResult;
22566: Func Generate_RSA_SymmetricKey(n,e:THugeCardinal;CipherStream: TStream;const SymetricCipher:IBlockCipher)
: TSymmetricKey;
22567: Func Extract_RSA_SymmetricKey(d,n:THugeCardinal;CipherStream: TStream;const
SymetricCipher:IBlockCipher):TSymmetricKey;
22568: end;
22569:
22570: Proc SIRegister_DXUtil(CL: TPSPascalCompiler);

```

```

22571: begin
22572: Proc SAFE_RELEASE( var i);
22573: Proc SAFE_DELETE( var Obj);
22574: Func DXUtil_GetDXSDKMediaPathCb( szDest : PChar; cbDest : Integer) : HRESULT;
22575: Func DXUtil_FindMediaFileCch(strDestPath:PChar;cchDest:Integer;strFilename:PChar):HRESULT;
22576: Func DXUtil_FindMediaFileCb(szDestPath: PChar; cbDest Integer;strFilename PChar):HRESULT;
22577: Func DXUtil_WriteStringRegKey(hKey :HKEY; strRegName : PChar; strValue : PChar):HRESULT;
22578: CL.AddTypeS('TIMER_COMMAND', 'DWORD');
22579: 'TIMER_RESET','LongInt').SetInt( 0);
22580: 'TIMER_START','LongInt').SetInt( 1);
22581: 'TIMER_STOP','LongInt').SetInt( 2);
22582: 'TIMER_ADVANCE','LongInt').SetInt( 3);
22583: 'TIMER_GETABSOLUTETIME','LongInt').SetInt( 4);
22584: 'TIMER_GETAPPTIME','LongInt').SetInt( 5);
22585: 'TIMER_GETELAPSEDTIME','LongInt').SetInt( 6);
22586: Func DXUtil_Timer( command : TIMER_COMMAND) : Single;
22587: Func DXUtil_ConvertGenericStringToAnsiCch(strDestination:PAnsiChar;const
tstrSource:PChar;cchDestChar:Intr): HRESULT;
22588: Func DXUtil_ConvertAnsiStrToGenericCch(tstrDestination:PChar;const strSource:PAnsiChar; cchDestChar:Int):
HRESULT;
22589: Func DXUtil_ConvertGenericStringToAnsiCb(strDestination:PAnsiChar;const tstrSource:PChar;cbDestChar:Int:
HRESULT);
22590: Func DXUtil_ConvertAnsiStrToGenericCb(tstrDestination:PChar;const
strSource:PChar;cbDestChar:Integer):HRESULT;
22591: CL.AddTypeS(TArrayListType', '( AL_VALUE, AL_REFERENCE );
22592: SIRegister_CArrayList(CL);
22593: Func GETTIMESTAMP : DWORD');
22594: end;
22595:
22596: Proc SIRegister_DCPbase64(CL: TPSPascalCompiler);
22597: begin
22598: Func Base64EncodeStr(const Value: Ansistr): Ansistr;
22599: Func Base64DecodeStr(const Value: Ansistr): Ansistr;
22600: Func Base64Encode( pInput :Str; pOutput :Str; Size : longint): longint;
22601: Func Base64Decode( pInput :Str; pOutput :Str; Size : longint): longint;
22602: end;
22603:
22604: Proc SIRegister_FlyFilesUtils(CL: TPSPascalCompiler);
22605: begin
22606: Func GetCaseSensitiveFileName( const FileName :Str; RootPath :Str) :Str;
22607: CL.AddConstantN('OTGDeivceCount','LongInt').SetInt( 16);
22608: CL.AddConstantN('UsbDiskStartIndex','LongInt').SetInt( 255);
22609: DeleteDirectories_WaitMinSecond','LongInt').SetInt( 2000);
22610: Func isPathCanUseNow( const PathOrDir :Str; const Default :Bool) :Bool;
22611: Func TestPathCanWrite( const PathOrDir :Str) :Bool;
22612: Func GetSDCardPath( Index : Integer) :Str;
22613: Func FindSDCardSubPath( SubPath :Str; Index : Integer) :Str;
22614: Func GetAppPath :Str;
22615: Func BuildFileListInAPath(const Path:str;const Attr:Int;const List:TStrings;JustFile:Bool):Bool;
22616: Func BuildFileListInAPath1(const Path:str; const Attr:Integer;ustFile:Bool):Str;
22617: Func GetFileNamesFromDirectory(const DirName:str;const SearchFilter:str;const FileAttribs:Int;const
isIncludeSubDirName:Bool;const Recursion:Bool;const FullName:Bool):str;
22618: Func DeleteDirectoryByEcho(const Source:str;AbortOnFailure:Bool;YesToAll:Bool;WaitMinSecond:Integer):Bool;
22619: Func GetTotalSpaceSize( Path :Str) : UInt64;
22620: Func GetAvailableSpaceSize( Path :Str) : UInt64;
22621: Func GetFreeSpaceSize( Path :Str) : UInt64;
22622: Func GetTotalMemorySize : UInt64;
22623: Func GetFreeMemorySize : UInt64;
22624: Func IsPadOrPC :Bool;
22625: Func OpenFileOnExtApp(const FileName:str;Https:Bool):Bool;
22626: Func NowGMT_UTC : TDateTime;
22627: Func EncodeURLWithSchemeOrProtocol( const URL :Str):Str;
22628: Func GetVolumePaths :Str;
22629: Func GetExternalStoragePath :Str;
22630: Func GetExterStoragePath :Str;
22631: Func GetInnerStoragePath :Str;
22632: Func GetIsExternalStorageRemovable :Bool;
22633: Func GetScreenClientInches : Single;
22634: Func IsCanFindJavaClass( const NamePath :Str) :Bool;
22635: Func IsCanFindJavaMethod(const MethodName,Signature:str;const CalssNamePath:str):Bool;
22636: Func IsCanFindJavaStaticMethod(const MethodName,Signature:str;const CalssNamePath:str):Bool;
22637: CL.AddTypeS(TGetFileNameLIsternerMethod','Procedure(const IsOK:Boolean;const FileName:str);
22638: Func OpenFileDialog(Title,FileExtension:str; GetFileNameCallBack: TGetFileNameLIsternerMethod):Boolean;
22639: Func CheckPermission( const APermissionName :Str) :Bool;
22640: CL.AddConstantN('C_android_permission_EXTERNAL_STORAGE',
'String').SetString('android.permission.WRITE_EXTERNAL_STORAGE');
22641: Func CanWriteExterStorage :Bool;
22642: Proc UpdateAlbum( FileNames :Str);
22643: Func ReadNoSizeFileToString( const AFileName :Str) :Str;
22644: Func ReadFileToString( const AFileName :Str) :Str;
22645: end;
22646:
22647: 466_kmean_cluster_correlation_test3.txt
22648: Proc SIRegister_LatLonDist(CL: TPSPascalCompiler);
22649: begin
22650: Func EllipticalDistance(llat1,llon1,llat2,llon2:extended;units:integer): extended;
22651: Proc VDirectLatLon(DGLAT1,DGLON1,DFAZ,S EXTENDED;Units:integer;var DGLAT2,DGLON2,DBAZ : EXTENDED);
22652: Func VInverseDistance(dlat1,dlon1,dlat2,dlon2:ext;var AzimuthInit,AzimuthFinal:ext;units:int):ext;
22653: Func RhumbDistance(llat1,llon1,llat2,llon2:extended;units:integer;var Azimuth:extended):extended;

```

```

22654: Proc RhumbLatLon(LAT1,LON1,Azimuth1,Dist:EXTENDED;Units:integer;var LAT2,LON2:EXTENDED);
22655: Func ApproxEllipticalDistance(llat1,llon1,llat2,llon2: extended;units:integer):extended;
22656: Func ApproxRhumbDistance(llat1,llon1,llat2,llon2: extended;units:integer; var Azimuth: extended):
extended;
22657: Proc ApproxRhumbLatLon(LAT1,LON1,Azimuth1,Dist:EXTENDED;Units integer;var LAT2,LON2: EXTENDED);
22658: Func convertunits( fromindex,ToIndex:integer;fromvalue:extended):extended;
22659: end;
22660:
22661: Proc SIRegister_PXLTiming(CL: TPSPascalCompiler);
22662: begin
22663: CL.AddConstantN('OverrunDeltaLimit','Extended').setExtended( 8.0);
22664: TTimerEvent', 'Proc ( const Sender : TObject);
22665: SIRegister_TMultimediaTimer2(CL);
22666: CL.AddTypeS(TSystemTimerValue, 'UInt64');
22667: Func TimerValueInBetween(const Value1,Value2: TSystemTimerValue):TSystemTimerValue;
22668: Func TickCountInBetween( const Value1, Value2:Card) :Card;
22669: Func GetSystemTimerValue: TSystemTimerValue;
22670: Func GetSystemTickCount:Card;
22671: Func GetSystemTimeValue: Double;
22672: Proc MicroSleep( const Microseconds : UInt64);
22673: end;
22674:
22675: (* === compile-time registration functions === *)
22676: (*-----*)
22677: Proc SIRegister_TMultimediaTimer2(CL: TPSPascalCompiler);
22678: begin
22679: //with RegClassS(CL,'TOBJECT', 'TMultimediaTimer') do
22680: type TMultimediaTimer2 = PXLTiming.TMultimediaTimer;
22681: with CL.AddClassN(CL.FindClass('TOBJECT'),'TMultimediaTimer2') do begin
22682: Constructor Create;
22683: Proc Process;
22684: Proc Reset;
22685: Proc NotifyTick( AllowSleep :Bool);
22686: Delta', 'Double', iptr);
22687: Latency', 'Double', iptr);
22688: FrameRate', 'Integer', iptr);
22689: Speed', 'Double', iptrw);
22690: MaxFPS', 'Integer', iptrw);
22691: Enabled', 'Boolean', iptrw);
22692: SingleCallOnly', 'Boolean', iptrw);
22693: OnTimer', 'TTimerEvent', iptrw);
22694: OnProcess', 'TTimerEvent', iptrw);
22695: end;
22696: end;
22697:
22698: Proc SIRegister_TNetCom(CL: TPSPascalCompiler);
22699: begin
22700: //with RegClassS(CL,'TOBJECT', 'TNetCom') do
22701: with CL.AddClassN(CL.FindClass('TOBJECT'),'TNetCom') do begin
22702: Constructor Create;
22703: Proc Free;
22704: Func Initialize :Bool;
22705: Proc Finalize;
22706: Func ResolveHost( const Host :Str) :Str;
22707: Func ResolveIP( const IPAddress :Str) :Str;
22708: Func SendStr(const Host:Str;const Port:Integer;const Data:str;const Size:Int):Bool;
22709: Func Send2(const Host:str;const Port:Integer;const Data:str;const Size:Int):Bool;
22710: Func Send(const Host:str;const Port:Integer;const Data:str;const Size:Int):Bool;
22711: Proc Update;
22712: Proc ResetStatistics;
22713: LocalIP', 'String', iptr);
22714: Initialized', 'Boolean', iptr);
22715: Broadcast', 'Boolean', iptrw);
22716: LocalPort', 'Integer', iptrw);
22717: //('OnReceive', 'TReceiveEvent', iptrw);
22718: UpdateRefreshTime', 'Integer', iptrw);
22719: BytesReceived', 'Integer', iptr);
22720: BytesSent', 'Integer', iptr);
22721: SentPackets', 'Integer', iptr);
22722: ReceivedPackets', 'Integer', iptr);
22723: BytesPerSec', 'Integer', iptr);
22724: end;
22725: end;
22726:
22727: (*-----*)
22728: Proc SIRegister_PXLNetComs(CL: TPSPascalCompiler);
22729: begin
22730: CL.AddConstantN('MaximumPacketSize','LongInt').SetInt( 8166);
22731: CL.AddTypeS('TSocket', 'Integer'); //delphisocks as longint
22732: CL.AddTypeS('TInAddr', 'record S_addr :Card; end');
22733: SIRegister_TNetCom(CL);
22734: end;
22735:
22736: unit uPSI_cTCPConnection.pas
22737: Proc RIRegister_cTCPConnection_Routines(S: TPSExec);
22738: Func TCPGetTick : LongWord;
22739: Func TCPTickDelta( const D1, D2 : LongWord) : Integer;
22740: Func TCPTickDeltaW( const D1, D2 : LongWord) : LongWord;
22741:

```

```

22742: Proc SIRegister_cTCPConnection(CL: TPSPascalCompiler);
22743: begin
22744:   CL.AddClassN(CL.FindClass('TOBJECT'),'ETCPConnection');
22745:   CL.FindClass('TOBJECT'),'TCPConnection';
22746:   CL.AddTypeS('TTCPLogType', '( tLDebug, tLInfo, tLError );
22747:   TTCPCConnectionProxyState, (prsInit, prsNegotiating, prsFiltering, prsFinished, prsError, prsClosed);
22748:   SIRegister_TTCPCConnectionProxy(CL);
22749:   SIRegister_TTCPCConnectionProxyList(CL);
22750:   TTCPCConnectionState, '(cnsInit, cnsProxyNegotiation, cnsConnected, cnsClosed );
22751: TTCPCConnectionTransferState, record LastUpdate: LongWord; ByteCount: Int64; TransferRate: LongWord; end;
22752: TAnsiCharSet, 'set of AnsiChar';
22753: TTCPCConnectionNotifyEvent, 'Proc ( Sender : TTCPCConnection);
22754: TTCPCConnectionStateChangeEvent, 'Proc ( Sender : TTCPCConnection; State: TTCPCConnectionState);
22755: TTCPCConnectionLogEvent, 'Procedure (Sender: TTCPCConnection; LogType: TTCPLogType; LogMsg: str; LogLevel: Int;
22756: SIRegister_TTCPCConnection(CL);
22757: //CL.AddTypeS('TTCPCConnectionClass', 'class of TTCPCConnection');
22758: Func TCPGetTick : LongWord';
22759: Func TCPTickDelta( const D1, D2 : LongWord) : Integer';
22760: Func TCPTickDeltaW( const D1, D2 : LongWord) : LongWord';
22761: end;
22762:
22763: Proc RIRegister_cTCPConnection_Routines(S: TPSExec);
22764: begin
22765:   S.RegisterDelphiFunction(@TCPGetTick, 'TCPGetTick', cdRegister);
22766:   S.RegisterDelphiFunction(@TCPTickDelta, 'TCPTickDelta', cdRegister);
22767:   S.RegisterDelphiFunction(@TCPTickDeltaW, 'TCPTickDeltaW', cdRegister);
22768: end;
22769:
22770: var HTMLWindow2: IHTMLWindow2;
22771: const HTMLStrGoogle 'string'.SetString(HTMLStrGoogle); //javascript
22772:
22773: Proc GoogleMapsFormCreate(Sender: TObject; aform: TForm; webbrowser1: TWebbrowser; HTMLStr: str);
22774: Proc GoogleMapsGotoLocationClick(Sender: TObject; LatitudeText, LongitudeText: str );
22775: begin
22776:   HTMLWindow2.execScript(Format('GotoLatLng(%s,%s)', [LatitudeText, LongitudeText]), 'JavaScript');
22777: end;
22778:
22779: Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress: str);
22780: var address : str;
22781: begin
22782:   address:= vaddress;
22783:   address:=StringReplace(StringReplace(Trim(address), #13, ' ', [rfReplaceAll]), #10, ' ', [rfReplaceAll]);
22784:   HTMLWindow2.execScript(Format('codeAddress(%s)', [QuotedStr(address)]), 'JavaScript');
22785: end;
22786: Func TaskbarHandle: THandle';
22787: Func TrayHandle: THandle';
22788: Proc DOSWipeFile(fn: str);
22789: Proc DOSFileWipe(fn: str);
22790: Func ExecAndWait(sExe, sCommandLine: str): Bool;
22791: Proc GoogleMapsFormCreate(Sender: TObject; aform: TForm; webbrowser1: TWebbrowser; HTMLStr, navdoc: str);
22792: Proc GoogleMapsGotoLocationClick(Sender: TObject; LatitudeText, LongitudeText: str );
22793: Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress: str);
22794: Proc GoogleMapsGotoAddressClick(Sender: TObject; vaddress: str);
22795: Proc BitmapToMetafile(const Bmp: {Graphics.}TBitmap; const EMF: {Graphics.}TMetafile);
22796:
22797: Proc SIRegister_DrBobCGI(CL: TPSPascalCompiler);
22798: begin
22799:   CL.AddTypeS('TBobRequestMethod', '(bobUnknown, bobGet, bobPost );
22800:   Func bobValue( const Field : ShortString; Convert : Bool) : ShortString';
22801:   Func CookieValue( const Field : ShortString) : ShortString';
22802:   Func getCGIEnvValues: str;
22803: end;
22804:
22805: Func getCGIEnvValues: str;
22806: begin
22807:   result:= intoStr(ContentLength)+';'+RemoteAddress+';'+HttpUserAgent+';'+
22808:   Authorization+';'+ScriptName+'.'
22809: end;
22810:
22811: Proc SIRegister_OverbyteIcsCharsetUtils(CL: TPSPascalCompiler);
22812: begin
22813:   CL.AddConstantN('ERR_CP_NOTMAPPED', 'LongInt').SetInt( MAX_CODEPAGE + 1);
22814:   'ERR_CP_NOTAVAILABLE', 'LongInt').SetInt( MAX_CODEPAGE + 2);
22815:   '^CP_US_ASCII', 'LongInt').SetInt( 20127);
22816:   CL.AddTypeS('CsuString', 'String');
22817:   CL.AddTypeS('TTimeCharset', '( CS_DEFAULT, CS_NOTMAPPED, UTF_8, WIN_1250, WIN_
22818:   +'1251, WIN_1252, WIN_1253, WIN_1254, WIN_1255, WIN_1256, WIN_1257, WIN_125
22819:   +'8, ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, ISO_8859_5, ISO_8859_6,
22820:   +' ISO_8859_7, ISO_8859_8, ISO_8859_8_i, ISO_8859_9, ISO_8859_13, ISO_8859_1
22821:   +'5, ISO_2022_JP, ISO_2022_JP_1, ISO_2022_JP_2, ISO_2022_KR, ISO_2022_CN, X_
22822:   +'CP50227, EUC_JP, GB_2312_80, GB_2312, HZ_GB_2312, GB_18030, EUC_CN, KOI8_R'
22823:   +'1, KOI8_U, UTF_16LE, UTF_16BE, UTF_7, SHIFT_JIS, BIG_5, KOREAN_HANGUL, EUC_
22824:   +'KR, WIN_874, IBM_037, IBM_437, IBM_500, IBM_850, IBM_852, IBM_855, IBM_857'
22825:   +'1, IBM_00858, IBM_860, IBM_861, IBM_862, IBM_863, IBM_864, IBM_865, IBM_866'
22826:   +'1, IBM_869, IBM_870, IBM_1026, IBM_01047, IBM_01140, IBM_01141, IBM_01142, '
22827:   +'IBM_01143, IBM_01144, IBM_01145, IBM_01146, IBM_01147, IBM_01148, IBM_0114
22828:   +'9, MACINTOSH, UTF_32LE, UTF_32BE, US_ASCII, T_6I, CS_LAST_ITEM );
22829:   CL.AddTypeS('TTimeCharsets', 'set of TTimeCharset');
22830:   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);

```



```

22831: 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');
22832: (* _cpinfoexA = record
22833:     MaxCharSize      : UINT;          { max length (bytes) of a char }
22834:     DefaultChar      : array[0..MAX_DEFAULTCHAR - 1] of Byte;    { default character }
22835:     LeadByte         : array[0..MAX_LEADBYTES - 1] of Byte;      { lead byte ranges }
22836:     UnicodeDefaultChar : WideChar;
22837:     CodePage         : UINT;
22838:     CodePageName     : array[0..MAX_PATH] of AnsiChar;
22839: end; *)
22840: //CL.AddTypeS('PCharSetInfo', '^TCharSetInfo // will not work');
22841: CL.AddTypeS('TCharSetInfo', 'record MimeCharSet : TMimeCharSet; CodePage : Lo'
22842: +ngWord; MimeName : CsuString; FriendlyName :Str; end');
22843: CL.AddTypeS('TCharSetInfos', 'array of TCharSetInfo');
22844: //CL.AddTypeS('LPCPINFOEXA', '^CPINFOEXA // will not work');
22845: CL.AddTypeS('CPINFOEXA', '_cpinfoexA');
22846: CL.AddTypeS('TCpInfoExA', 'CPINFOEXA');
22847: //CL.AddTypeS('PCpInfoExA', 'LPCPINFOEXA');
22848: //CL.AddTypeS('LPCPINFOEXW', '^CPINFOEXW // will not work');
22849: CL.AddTypeS('CPINFOEXW', '_cpinfoexW');
22850: CL.AddTypeS('TCpInfoExW', 'CPINFOEXW');
22851: //CL.AddTypeS('PCpInfoExW', 'LPCPINFOEXW');
22852: //CL.AddTypeS('PCpInfoEx', 'PCpInfoExW');
22853: CL.AddTypeS('CPINFOEX', 'CPINFOEXW');
22854: CL.AddTypeS('TCpInfoEx', 'TCpInfoExW');
22855: //CL.AddTypeS('LPCPINFOEX', 'LPCPINFOEXW');
22856: //CL.AddTypeS('PCpInfoEx', 'PCpInfoExA');
22857: CL.AddTypeS('CPINFOEX', 'CPINFOEXA');
22858: CL.AddTypeS('TCpInfoEx', 'TCpInfoExA');
22859: //CL.AddTypeS('LPCPINFOEX', 'LPCPINFOEXA');
22860: SIRegister_TCodePageObj(CL);
22861: CL.AddDelphiFunction('Func CodePageToMimeCharSet( ACodePage : LongWord) : TMimeCharSet');
22862: Func CodePageToMimeCharSetString( ACodePage : LongWord) : CsuString);
22863: Func GetMimeInfo0( AMimeCharSet : TMimeCharSet) : PCharSetInfo;
22864: //&&CL.AddDelphiFunc('Func GetMimeInfo1(const AMimeCharSetStr:CsuString):PCharSetInfo;
22865: //CL.AddDelphiFunction('Func GetMimeInfo2( ACodePage : LongWord) : PCharSetInfo;
22866: Func MimeCharSetToCharSetString( AMimeCharSet : TMimeCharSet) : CsuString');
22867: ('Func ExtractMimeName( PInfo : PCharSetInfo) : CsuString');
22868: Func MimeCharSetToCodePage3( AMimeCharSet : TMimeCharSet) : LongWord;
22869: Func MimeCharSetToCodePage4(const AMimeCharSetStr:CsuString;out ACodePage:LongWord):Bool;
22870: Func MimeCharSetToCodePageDef( const AMimeCharSetString : CsuString) : LongWord);
22871: Func MimeCharSetToCodePageEx5(const AMimeCharSetStr:CsuString;out ACodePage:LongWord):Bool;
22872: Func MimeCharSetToCodePageExDef( const AMimeCharSetString : CsuString) : LongWord);
22873: Func IsValidAnsiCodePage( ACodePage : LongWord) :Bool);
22874: Func IcsIsValidCodePageID( ACodePage : LongWord) :Bool);
22875: Func IsSingleByteCodePage( ACodePage : LongWord) :Bool);
22876: Proc GetSystemCodePageList( AOwnsObjectList : TObjectList);
22877: Func AnsiCodePageFromLocale( ALcid : LCID) : LongWord);
22878: Func OemCodePageFromLocale( ALcid : LCID) : LongWord);
22879: Func GetThreadAnsiCodePage : LongWord);
22880: Func GetThreadOemCodePage : LongWord);
22881: Func GetUserDefaultAnsiCodePage : LongWord);
22882: Func GetUserDefaultOemCodePage : LongWord);
22883: Func GetCPInfoExA( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEXA) : BOOL);
22884: Func GetCPInfoExW( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEXW) : BOOL);
22885: Func GetCPInfoEx( CodePage : UINT; dwFlags : DWORD; var lpCPInfoEx : CPINFOEX) : BOOL);
22886: Proc GetFriendlyCharSetList(Items:TStrings;IncludeList: TMimeCharsets; ClearItems : Bool);
22887: Proc GetMimeCharSetList( Items : TStrings; IncludeList: TMimeCharsets; ClearItems: Bool);
22888: end;
22889:
22890: Proc SIRegister_OverbyteIcsMimeUtils(CL: TPSPascalCompiler);
22891: begin
22892: TMimeUtilsVersion('LongInt').SetInt( 802);
22893: 'mimuutilsCopyRight', 'String').SetString( ' MimeUtils (c) 2003-2014 F. Piette V8.02 ');
22894: 'SmtPDefaultLineLength', 'LongInt').SetInt( 76);
22895: 'SMTP_SND_BUF_SIZE', 'LongInt').SetInt( 2048);
22896: 'icsRegContentType', 'String').SetString( 'MIME\Database\Content Type');
22897: //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)(' . ' ));
22898: //CL.AddConstantN('CrLfSet', 'TSysCharSet').SetString(ord(AnsiChar)('#13) or ord(AnsiChar)('#10));
22899: //CL.AddConstantN('QuotedCharSet', 'TSysCharSet').SetString(ord(AnsiChar)('? ' ) or ord(AnsiChar)('= ' ) or
ord(AnsiChar)(' ' ) or ord(AnsiChar)(' ' ));
22900: //CL.AddConstantN('BreakCharsSet', 'TSysCharSet').SetString(ord(AnsiChar)(' #9 ' ) or ord(AnsiChar)(' #32 ' ) or
ord(AnsiChar)(' ; ' ) or ord(AnsiChar)(' , ' ) or ord(AnsiChar)(' > ' ) or ord(AnsiChar)(' ' ));
22901: Func EncodeQuotedPrintable0( const S : RawByteString) : Str;
22902: Func EncodeQuotedPrintable1(const S:UnicodeString; ACodePage: LongWord) :UnicodeString;
22903: Func EncodeQuotedPrintable2( const S : UnicodeString) : UnicodeString;
22904: Func DecodeQuotedPrintable3( const S : RawByteString) : RawByteString;
22905: Func DecodeQuotedPrintable4(const S : UnicodeString; ACodePage:LongWord):UnicodeString;
22906: Func DecodeQuotedPrintable5( const S : UnicodeString) : UnicodeString;
22907: Func SplitQuotedPrintableString( const S :Str) :Str);
22908: Func FilenameToContentType( FileName :Str) :Str);
22909: Func Base64Encode6( const Input : Ansistr) : Ansistr;
22910: Func Base64Encode7( const Input : PAnsiChar; Len : Integer) : Ansistr;
22911: Func Base64Encode8( const Input : UnicodeString; ACodePage : LongWord):UnicodeString;
22912: Func Base64Encode9( const Input : UnicodeString) : UnicodeString;

```

```

22913: Func Base64Encode10( Input :StrBuilder) :StrBuilder;
22914: Func Base64Decode11( const Input: Ansistr) : Ansistr;
22915: Func Base64Decode12( const Input: UnicodeString; ACodePage : LongWord):UnicodeString;
22916: Func Base64Decode13( const Input: UnicodeString) : UnicodeString;
22917: ('Func Base64Decode14( Input :StrBuilder) :StrBuilder;
22918: Func InitFileEncBase64( const FileName :Str; ShareMode : Word) : TStream'';
22919: Func DoFileEncBase64( var Stream : TStream; var More :Bool) : Ansistr'';
22920: Func DoFileEncQuotedPrintable( var Stream : TStream; var More :Bool) : Ansistr'';
22921: Func DoTextFileReadNoEncoding( var Stream : TStream; var More :Bool) : Ansistr'';
22922: Func DoFileLoadNoEncoding( var Stream : TStream; var More :Bool) :Str'';
22923: Func Base64EncodeEx15(const Input:RawByteString;MaxCol:Int;var
cPos:Integer;CodePage:LongWord;IsMultiByteCP:Bool):RawByteString;
22924: Func Base64EncodeEx16(const Input:UnicodeString;MaxCol:Integer;var cPos:Int):UnicodeString;
22925: Func Base64EncodeEx17(const Input:UnicodeString;MaxCol:Integer;var cPos:Integer;ACodePage:LongWord):
UnicodeString;
22926: Proc EndFileEncBase64( var Stream : TStream);
22927: Proc DotEscape( var S :Str; OnlyAfterCrLf :Bool);
22928: //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;
22929: Func IcsWrapTextEx19(const Line:Str; const BreakStr :Str; const BreakingChars :
TSysCharSet;MaxCol:Integer;QuoteChars:TSysCharSet;var cPos:Integer;ForceBreak Boolean):str;
22930: Func UnFoldHdrLine( const S :Str) :Str'';
22931: Func NeedsEncoding20( const S : Ansistr) :Bool;
22932: Func NeedsEncoding21( const S : UnicodeString) :Bool;
22933: Func NeedsEncodingPChar( S : PChar) :Bool'';
22934: Func HdrEncodeInLine22(const Input:RawByteString;Specials:TSysCharSet; EncType:AnsiChar;const CharSet:
Ansistr; MaxCol:Integer; DoFold :Bool;CodePage:LongWord;IsMultiByteCP:Bool):RawByteString;
22935: Func HdrEncodeInLine23(const Input:UnicodeString;Specials:TSysCharSet;EncType:WideChar;const CharSet:
UnicodeString; MaxCol:Integer;DoFold:Bool;Codepage:LongWord;IsMultiByteCP:Bool):UnicodeString;
22936: Func HdrEncodeInLineEx( const Input : UnicodeString
Specials:TSysCharSet;EncType:WideChar;CodePage:LongWord;MaxCol:Integer;DoFold:Bool;IsMultiByteCP:Bool):RawByteString'';
22937: Func StrEncodeQP24(const Input:RawByteString;MaxCol:Integer;Specials:TSysCharSet;CodePage:LongWord;
IsMultiByteCP:Bool):Str;
22938: Func StrEncodeQP25(const Input:UnicodeString;MaxCol:Integer;Specials:TSysCharSet;ACodePage:LongWord;
IsMultiByteCP:Bool):UnicodeString;
22939: Func StrEncodeQPEx26(const Buf:RawByteString;MaxCol:Integer;Specials:TSysCharSet;ShortSpace:Bool;var
cPos:Int;DoFold:Bool;CodePage:LongWord;IsMultiByteCP:Bool):RawByteString;
22940: Func StrEncodeQPEx27(const Buf:UnicodeString;MaxCol:Integer;Specials:TSysCharSet;ShortSpace:Boolean;var
cPos: Integer;DoFold:Boolean):UnicodeString;
22941: Proc FoldHdrLine28( HdrLines : TStrings; const HdrLine :Str);
22942: Proc FoldHdrLine29(HdrLines:TStrings;const HdrLine:RawByteString;
ACodePage:LongWord;IsMultiByteCP:Boolean);
22943: Func FoldString30(const Input:RawByteString;BreakCharsSet:TSysCharSet;MaxCol:Integer;ACodePage:LongWord;
IsMultiByteCP:Bool):RawByteString;
22944: Func FoldString31(const Input:UnicodeString;BreakCharsSet:TSysCharSet;MaxCol:Int):UnicodeString;
22945: Func CalcBase64AttachmentGrow(FileSize : Int64) : Int64'';
22946: //Func EncodeMbcInline32(CodePage:LongWord;const
CharSet:str;EncType:Char;Body:PWideChar;Len:Integer;DoFold:Boolean;MaxLen Integer): Ansistr;
22947: //Func EncodeMbcInline33(CodePage:LongWord;const
CharSet:str;EncType:Char;Body:PAnsiChar;Len:Integer;DoFold:Boolean;MaxLen: Integer): Ansistr;
22948: Func ContentTypeGetExtn( const Content :Str; var CLSID :Str) :Str'';
22949: 'Func ContentTypeFromExtn( const Extension :Str) :Str'';
22950: CL.AddTypeS('TMimeTypeSrc','( MTypeList, MTypeOS, MTypeMimeFile, MTypeKeyFile, MTypeRes );
22951: SIRegister_TMimeTypeList(CL);
22952: end;
22953:
22954: Returns the MIME name of a Windows code page identifier or an empty }
22955: { string if the code page identifier is not mapped. }
22956: Func CodePageToMimeCharsetString(ACodePage: LongWord): CsuString;
22957:
22958: Proc SIRegister_TMimeTypeList(CL: TPSPascalCompiler);
22959: begin
22960: //with RegClass(CL,'TComponent', 'TMimeTypeList') do
22961: with CL.AddClassN(CL.FindClass('TComponent'),'TMimeTypeList') do begin
22962: Constructor Create( AOwner : TComponent);
22963: Proc Free'';
22964: Func LoadTypeList :Bool'';
22965: Proc Clear'';
22966: Func CountExtn : integer'';
22967: Func CountContent : integer'';
22968: Func AddContentType(const AExtn, AContent:Str) :Bool'';
22969: Func LoadWinReg :Bool'';
22970: Func LoadFromOS :Bool'';
22971: Func LoadFromList :Bool'';
22972: Func LoadMimeFile( const AFileName :Str) :Bool'';
22973: Func LoadFromResource( const AResName :Str) :Bool'';
22974: Func LoadFromFile( const AFileName :Str) :Bool'';
22975: Func SaveToFile( const AFileName :Str) :Bool'';
22976: Proc LoadContentTypes( AList : TStrings);
22977: Proc AddContentTypes( AList : TStrings);
22978: Proc GetContentTypes( AList : TStrings);
22979: Func TypeFromExtn( const AExtn :Str) :Str'';
22980: Func TypeFromFile( const AFileName :Str) :Str'';
22981: Func TypeGetExtn( const AContent :Str) :Str'';
22982: RegisterProperty('LoadOSonDemand', 'boolean', iptrw);
22983: RegisterProperty('MimeTypeFile', 'string', iptrw);
22984: RegisterProperty('DefaultTypes', 'TStringList', iptrw);
22985: RegisterProperty('MimeTypeSrc', 'TMimeTypeSrc', iptrw);

```

```

22986:   RegisterProperty('UnknownType', 'string', iptrw);
22987: end;
22988: end;
22989:
22990: Proc SIRegister_uWebSocket (CL: TPSPascalCompiler);
22991: begin
22992:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TWebSocketConnection');
22993:   CL.AddTypeS(TWebSocketMessageEvent, 'Proc (AConnection:TWebSocketConnection;const AMessage:str);
22994:   TWebSocketConnectEvent', 'Proc ( AConnection : TWebSocketConnection);
22995:   TWebSocketDisconnectEvent', 'Proc ( AConnection : TWebSocketConnection);
22996:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TWebSocketException');
22997:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TWebSocketHandshakeException');
22998:   SIRegister_TWebSocketRequest (CL);
22999:   SIRegister_TWebSocketConnection (CL);
23000:   SIRegister_TWebSocketServer (CL);
23001: end;
23002:
23003: (*-----*)
23004: Proc SIRegister_OverbyteIcsUrl (CL: TPSPascalCompiler);
23005: begin
23006:   CL.AddConstantN('IcsUrlVersion', 'LongInt').SetInt ( 800);
23007:   CL.AddConstantN('uriCopyRight', 'String').SetString(' TicsURL (c) 1997-2012 F. Piette V8.00 ');
23008:   Proc icsParseURL ( const URL :Str; var Proto, User, Pass, Host, Port, Path:Str);
23009:   Func icsPosn ( const s, t :Str; count : Integer ) : Integer);
23010:   Func icsUrlEncode ( const S:str;DstCodePage:LongWord) :Str);
23011:   Func icsUrlDecode0 (const S:str;SrcCodePage:LongWord;DetectUtf8:Boolean):str;
23012:   Func icsUrlDecode1 (const S:RawByteString;SrcCodePage:LongWord;DetectUtf8:Bool):UnicodeString;
23013:   Func icsUrlEncodeToA ( const S:Str; DstCodePage : LongWord ) : Ansistr);
23014: end;
23015:
23016: (*-----*)
23017: Proc SIRegister_OverbyteIcsLogger (CL: TPSPascalCompiler);
23018: begin
23019:   CL.AddConstantN('TicsLoggerVersion', 'LongInt').SetInt ( 800);
23020:   IcsCopyRight, 'String').SetString(' IcsLogger (c) 2005-2012 by François PIETTE V8.00 ');
23021:   CL.AddClassN(CL.FindClass('TOBJECT'), 'ELoggerException');
23022:   CL.AddTypeS('TLogOption', (loDestEvent, loDestFile, loDestOutDebug, loAddStamp, loWsockErr, loWsockInfo,
loWsockDump, loSslErr, loSslInfo, loSslDump, loProtSpecErr, loProtSpecInfo, loProtSpecDump);
23023:   TLogOptions', 'set of TLogOption');
23024:   TLogFileOption', '( lfoAppend, lfoOverwrite );
23025:   TLogFileEncoding', '( lfeUtf8, lfeUtf16 );
23026:   CL.AddConstantN(LogAllOptErr', 'LongInt').Value.ts32:=ord(loWsockErr) or ord(loSslErr) or
ord(loProtSpecErr);
23027:   TNTEventType', '( etError, etWarning, etInformation, etAuditSuccess, etAuditFailure );
23028:   TicsLogEvent', 'Proc ( Sender : TObject; LogOption : TLogOption; const Msg :Str);
23029:   SIRegister_TicsLogger (CL);
23030: end;
23031:
23032: Proc MoveCardinal2String(const Source:Card; SrcStartIdx: Integer; var Dest:Str;
23033:   DstStartIdx: Integer; Count: Integer);
23034: Proc MoveString2Cardinal(const Source:Str; SrcStartIdx: Integer; var Dest:Card;
23035:   DstStartIdx: Integer; Count: Integer);
23036: Proc MoveCardinal(const Source:Card; var Dest:Card; Count: Integer);
23037: Func ExplodeStr(S:Str; Delimiter: Char; Index: Integer):Str; //overload;
23038:   var I: Integer; P, P1: PChar;
23039: Func ImplodeStr(S:Str; Delimiter: Char):Str;
23040: Func ExplodeStr(S:Str; Delimiter: Char; Index: Integer):Str; //overload;
23041: Func InSetWord(n:word):boolean;
23042:
23043:   borland video frame
23044:   <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>
23045:   resourcestrings
23046:   { Charsets, user-friendly names, known from OE and IE }
23047:   sArabicISO = 'Arabic (ISO)'; //28596
23048:   sArabicWindows = 'Arabic (Windows)'; //1256
23049:   sBalticISO = 'Baltic (ISO)'; //28594
23050:   sBalticWindows = 'Baltic (Windows)'; //1257
23051:   sCentralEuropeanISO = 'Central European (ISO)'; //28592
23052:   sCentralEuropeanWindows = 'Central European (Windows)'; //1250
23053:   sChineseTraditionalBig5 = 'Chinese Traditional (Big5)'; //950
23054:   sChineseSimplifiedGB18030 = 'Chinese Simplified (GB18030)'; //54936
23055:   sChineseSimplifiedGB2312 = 'Chinese Simplified (GB2312)'; //936
23056:   sChineseSimplifiedHZ = 'Chinese Simplified (HZ)'; //52936
23057:   sCyrillicISO = 'Cyrillic (ISO)'; //28595
23058:   sCyrillicKOI8R = 'Cyrillic (KOI8-R)'; //20866
23059:   sCyrillicKOI8U = 'Cyrillic (KOI8-U)'; //21866
23060:   sCyrillicWindows = 'Cyrillic (Windows)'; //1251
23061:   sEstonianISO = 'Estonian (ISO)'; //28603
23062:   sGreekISO = 'Greek (ISO)'; //28597
23063:   sGreekWindows = 'Greek (Windows)'; //1253
23064:   sHebrewISOLogical = 'Hebrew (ISO-Logical)'; //38598
23065:   sHebrewISOVisual = 'Hebrew (ISO-Visual)'; //28598
23066:   sHebrewWindows = 'Hebrew (Windows)'; //1255
23067:   sJapaneseJIS = 'Japanese (JIS)'; //932
23068:   sKorean = 'Korean'; //949
23069:   sKoreanEUC = 'Korean (EUC)'; //51949
23070:   sLatin9 = 'Latin 9 (ISO)'; //28605
23071:   sThaiWindows = 'Thai (Windows)'; //874

```

```

23072: sTurkishISO = 'Turkish (ISO)'; //28599
23073: sTurkishWindows = 'Turkish (Windows)'; //1254
23074: sUnicodeUTF7 = 'Unicode (UTF-7)'; //65001
23075: sUnicodeUTF8 = 'Unicode (UTF-8)'; //65000
23076: sVietnameseWindows = 'Vietnamese (Windows)'; //1258
23077: sWesternEuropeanISO = 'Western European (ISO)'; //28591
23078: sWesternEuropeanWindows = 'Western European (Windows)'; //1252
23079:
23080: Proc SRegister_psULib(CL: TPSPascalCompiler); // Prometheus
23081: begin
23082:   'clMoneyGreenps','LongWord').SetUInt( TColor ( $00C0DCC0 ));
23083:   ('clMedGrayps','LongWord').SetUInt( TColor ( $00A4A0A0 ));
23084:   ('clRxCreampss','LongWord').SetUInt( TColor ( $00A6CAFA ));
23085:   ('clCreampss','LongWord').SetUInt( TColor ( $00F0FBFF ));
23086:   ('clRxSkyBlueps','LongWord').SetUInt( TColor ( $00FFFBF0 ));
23087:   ('clSkyBlueps','LongWord').SetUInt( TColor ( $00F0CAA6 ));
23088:   ('clInfoBkps','LongWord').SetUInt( TColor ( $02E1FFFF ));
23089:   ('clNoneps','LongWord').SetUInt( TColor ( $02FFFFFF ));
23090:   ('clSystemColor','LongWord').SetUInt( $FF000000);
23091:   ('MinutesPerDay','LongInt').SetInt( 24 * 60);
23092:   ('DefaultBeepDelay','LongInt').SetInt( 500);
23093:   ('cDoNotBringToTop','String').SetString( '-LaunchInBackground');
23094:   CL.AddTypeS('TFilenameEvent', 'Proc ( Sender : TObject; var Filename :Str);
23095:   //CL.AddTypeS('TWriteDebugProc', 'Procedure(const LogName,RoutineName,Commen:Str;DebugData:array of
string);
23096:   CL.AddTypeS('TMaxIntSize', '( mbsInt8, mbsWord8, mbsInt16, mbsWord16, mbsInt32, mbsWord32 );
23097:   Proc WriteDebuggs(const LogName,RoutineName,Comment:Str;DebugData:array of string);
23098:   Func Betweenps( Value, BoundA, BoundB : integer; AllowFlip :Bool) :Bool);
23099:   Func BooleanMatchsp( Value1 :Bool; Value2 : integer) :Bool;
23100:   Func BooleanMatchlps( Value1 : integer; Value2 :Bool) :Bool;
23101:   Func BooleanMatch2ps( Value1, Value2 : integer) :Bool;
23102:   Func IDMatchps( RequiredID, TestID : integer) :Bool;
23103:   Func IDMatchlps( RequiredID : TField; TestID : integer) :Bool;
23104:   Func IDMatch2ps( RequiredID, TestID : TField) :Bool;
23105:   Func Blendps( Color1, Color2 : TColor; Weight1 : integer; Weight2 : integer) : TColor);
23106:   Func MergeRGBps( Red, Green, Blue : integer) : TColor);
23107:   Proc SplitRGBps( Color : TColor; var Red, Green, Blue : integer);
23108:   Func VisibleContrastps( BackgroundColor : TColor) : TColor);
23109:   Func SortedStrps( Value :Str) :Str);
23110:   Func IntSortStrps( const Value : integer) :Str);
23111:   //Func AllAssignedps( Values : array of pointer) :Bool);
23112:   Func GetTokenps( var SourceStr :Str; Delim :Str) :Str);
23113:   Func URLizeps( SourceStr :Str) :Str);
23114:   Func UnURLizeps( const SourceStr :Str) :Str);
23115:   Func UnQuoteStrps( const Value :Str; QuoteChar : char) :Str);
23116:   Proc SetAndSaveBoolps( var OldVar :Bool; Value :Bool; var SaveVar :Bool);
23117:   Proc SetAndSaveIntps( var OldVar : integer; Value : integer; var SaveVar : integer);
23118:   Proc SetAndSaveStrps( var OldVar :Str; const Value :Str; var SaveVar :Str);
23119:   Func GetCharFromVKeyps( VKey : word) :Str);
23120:   Func IsControlKeyDownps :Bool);
23121:   Proc PushScreenCursorsps( const aCursor : TCursor);
23122:   Proc PopScreenCursorsps);
23123:   Func PeekScreenCursorsps : TCursor);
23124:   Func CharIsNumericSymbolps( aChar : char) :Bool);
23125:   Func PrecisionMultiplierps( DecimalPrecision : byte) :Card);
23126:   Func TextToFloatValps(const aText:Str;const DecimalPrecision:byte;const DefaultValue:extended):extended;
23127:   Func TextToWordValps(const aText:Str;const DefaultVal:cardi;const MaxIntSize:TMaxIntSize):cardi;
23128:   Func TextToIntValps(const aText:Str;const DefaultVal:int;const MaxIntSize:TMaxIntSize):int;
23129:   Func BankersRoundingsps( const Num : currency; const DecimalPrecision : byte) : currency);
23130:   Func RoundDownps( const Num : Extended; const DecimalPrecision : byte) : Extended);
23131:   Func RoundUpss( const Num : Extended; const DecimalPrecision : byte) : Extended);
23132:   Func RoundNearestps( const Num : Extended; const DecimalPrecision : byte) : Extended);
23133:   Func NumDigitpsps( const aNum : integer) : integer);
23134:   Func ZeroToMaxps( Value : integer) : integer);
23135:   Proc CopyIfPrefixMatchps(Source, Dest:TStrings;Prefix:Str;ClearDest:bool;ValuesOnly:bool);
23136:   Func HexToCharps( const aValue : byte) : AnsiChar);
23137:   Func CharToHexps( const aValue : AnsiChar) : byte);
23138:   Func HexToTextps( const Source : Ansistr) : Ansistr);
23139:   Func TextToHexps( const Source : Ansistr) : Ansistr);
23140:   Proc SaveGraphicToStreamps( aGraphic : TGraphic; aStream : TStream);
23141:   Func LoadGraphicFromStreamps( aStream : TStream) : TGraphic);
23142:   Func GraphicToTextps( aGraphic : TGraphic) :Str);
23143:   Func TextToGraphicps( aGraphicAsText :Str) : TGraphic);
23144:   Func GetCustNameStrps( aFirstName, aLastName :Str) :Str);
23145:   Func GetLongCustNameStrps( aTitle, aFirstName, aLastName :Str) :Str);
23146:   Func GetPriceFromMarginps( const aMargin :Str; aCost : currency): currency);
23147:   Func GetMarginFromPriceps( aPrice, aCost : currency) :Str);
23148:   Func GetDriveSerialNumps( Path :Str) : longint);
23149:   Func GetCCCardTypeFromNameps( aCardName :Str) : integer);
23150:   Func GetItemMarginPercentps( Price, Cost : Currency) : double);
23151:   Func ValidateCreditCardps( aCardNum :Str) :Bool);
23152:   Func ValidateCreditCardRange( const aCardNum, aLowNum, aHighNum :Str) :Bool);
23153:   Func HHMMToTimesps( Value :Str) : TTime);
23154:   Func MMDDToDatesps( Value :Str) : TDate);
23155:   Func TimeToHHMMSSps( Value : TTime) :Str);
23156:   Func DateToYYYYMMDDps( Value : TDate) :Str);
23157:   Func FontColorToStringsps( SourceFont : TFont; SourceColor : TColor) :Str);
23158:   Proc StringToFontps( aStr :Str; DestFont : TFont);
23159:   Proc StringToFontColorps( aStr :Str; DestFont : TFont; var DestColor : TColor);

```



```

23160: Proc ShowDebugStringsps( DebugStrings : TStrings);
23161: Proc ShowStackDumpsps('');
23162: Proc RegisterFileExtensionps(const Extension,Description,DefaultActionName,DefaultActionDescription,
DefaultActionCommand:str; IconIndex:integer);
23163: Func LaunchCommandps(aCommand:str; aParams:str; WaitForTerminate:boolean):bool;
23164: Func GetWindowsDirectoryStrps :Str';
23165: Func IsRemoteSessionps :Bool';
23166: Func IsWinXPps :Bool';
23167: Proc AssignTextFileToStreamps( var ATextFile : TextFile; AStream : TStream);
23168: Func DeleteFileMaskps( const Mask :Str) :Bool';
23169: Func FileMaskCountps( const Mask :Str) : integer';
23170: Func FileMaskExistps( const Mask :Str) :Bool';
23171: Func FileMaskListps( const Mask :Str) :Str;
23172: Func FileMaskListps( const Mask:str; Dest : TStrings; ClearList :Bool) :Bool;
23173: Func GetSafeNumericVariantps( Value : Variant) : Variant';
23174: Func GetPasswordCharStrps( aStr :Str) :Str';
23175: Func GetDriversLicenseHeightDisplayTextps( aStr :Str) :Str';
23176: Func IncludePathDelimiterps( const aFilePath :Str) :Str';
23177: Func ExcludePathDelimiterps( const aFilePath :Str) :Str';
23178: //Func GetEnvironmentVariable( const Name :Str) :Str';
23179: Func ArrayToStrps( SourceArray : array of string; Separator : AnsiChar) :Str';
23180: Func psStrReplaces( var S : AnsiStr; const Search, Replace : AnsiStr; Flags : TReplaceFlags):boolean;
23181: Func StrReplaceInStringsps(StrObject:TStrings;const Search,Replace:AnsiStr;Flags:TReplaceFlags):bool;
23182: Proc ShowMessageps( aMsg :Str);
23183: Proc ShowErrorMessageps( aMsg :Str);
23184: Func StandardUserQueryps(aMsg:str;DialogType:Card;DialogCaption:Str):Card;
23185: Func ParamValueps( ParamName :Str) :Str';
23186: Func ParamFlagSetps( ParamName :Str) :Bool';
23187: Func ConcatenateNonBlankps( Strings : array of string) :Str';
23188: Func ConcatenateWithDelimiterps( Strings : array of string; Delimiter : PChar) :Str';
23189: Func GetMaxStringLengthps( Strings : array of string) : integer';
23190: Proc PlayBeeps( BeepActionType : TMsgDlgType; NumBeeps : integer; MSDelay : integer);
23191: Func DelTrees( const DirectoryName :Str) :Bool';
23192: Proc UpdateActionCaptionps( anAction : TAction; aCaption :Str);
23193: Proc GetDirectoryListps( const DirectoryName :Str; StringList : TStrings);
23194: Func DecodePathps( const Path :Str; PathVars : TStrings) :Str';
23195: Func KillEXEps( ExeName :Str) :Bool';
23196: Func GetWindowsDefaultPrinterNameps :Str';
23197: Func GetBackupFilenameps( const OrigFilename :Str) :Str';
23198: Func GetRestoreFilenameps( const BackupFilename :Str) :Str';
23199: Func MakeBackupFiles( const OrigFilename :Str; OverwriteExisting :Bool):Str';
23200: Func RestoreBackupFiles( const BackupFilename:str;OverwriteExisting:boolean):Str';
23201: Func CalcDistanceps( Lat1, Long1, Lat2, Long2 : double) : double';
23202: Func GetDelimiterCountps( aString, aDelimiter :Str) : integer'); //}
23203: end;
23204:
23205: (*-----*)
23206: Proc SIRegister_psUFinancial(CL: TPSPascalCompiler);
23207: begin
23208: CL.AddTypeS(TLoanType', '( ltAmortized, ltSimple );
23209: CL.AddTypeS(TLoanPayType', '( lptMonthly, lptBiMonthly, lptBiWeekly );
23210: TypeS(TLoanPaymentEvent', 'Procedure (PayDate:TDate; PayNum:int; PaymentAmt,Balance,InterestAmt,
TotalInterestAmt,AmortizedAmt,TotalAmortizedAmt:currency;var ExtraPayment: currency);
23211: Func
LoanPaymentAmt(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;LoanType
:TLoanType):currency';
23212: Func LoanNumPayments(Principal,
PaymentAmt:currency;InterestRate:extended;PayType:TLoanPayType;LoanType:TLoanType):int;
23213: Func LoanTotalEstimate(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;
LoanType:TLoanType):currency;
23214: Func
LoanInterestEstimate(Principal:currency;InterestRate:extended;TotalNumPay:integer;PayType:TLoanPayType;
LoanType:TLoanType):currency;
23215: Proc AmortizationSchedule( OnLoanPayment:TLoanPaymentEvent;StartDate:TDate;Principal
:currency;InterestRate:extended;TotalNumPay:integer;PayType TLoanPayType;LoanType:TLoanType);
23216: Proc AmortizationSchedulePaymentAmt(OnLoanPayment TLoanPaymentEvent;StartDate:TDate; Principal,
PaymentAmt:currency;InterestRate:extended;TotalNumPay:int;PayType:TLoanPayType; LoanType:TLoanType);
23217: end;
23218:
23219: (*-----*)
23220: Proc SIRegister_TSimpleRSS(CL: TPSPascalCompiler);
23221: begin
23222: //with RegClassS(CL,'TComponent', 'TSimpleRSS') do
23223: with CL.AddClassN(CL.FindClass('TComponent'),'TSimpleRSS') do begin
23224: Constructor Create( AOwner : TComponent);
23225: Proc Free';
23226: Proc SaveToFile( Filename :Str);
23227: Func SaveToString :Str';
23228: Proc SaveToStream( Stream : TStream);
23229: Func SaveToStrings : TStrings;
23230: Proc LoadFromHTTP( URL :Str);
23231: Proc LoadFromString( S :Str);
23232: Proc LoadFromFile( Filename :Str);
23233: Proc LoadFromStream( Stream : TStream);
23234: Proc LoadFromStrings( Strings : TStrings);
23235: Proc GenerateXML;
23236: Proc GenerateXML1( FeedType : TXMLTypeRSS);
23237: Proc GenerateComponent';
23238: Proc ClearXML';

```



```

23239:   Property('Channel', 'TRSSChannel', iptrw);
23240:   Property('Items', 'TRSSItems', iptrw);
23241:   Property('Version', 'string', iptrw);
23242:   Property('XMLType', 'TXMLTypeRSS', iptrw);
23243:   Property('XMLFile', 'TXMLDocument', iptr);
23244:   Property('IndyHTTP', 'TIdHTTP', iptrw);
23245:   Property('SimpleRSSVersion', 'String', iptr);
23246:   Property('OnCreate', 'TNotifyEvent', iptrw);
23247:   Property('OnGenerateXML', 'TNotifyEvent', iptrw);
23248:   Property('OnParseXML', 'TNotifyEvent', iptrw);
23249: end;
23250: end;
23251:
23252: (*-----*)
23253: Proc SIRegister_SimpleRSSTypes(CL: TPSPascalCompiler);
23254: begin
23255:   CL.AddClassN(CL.FindClass('TOBJECT'), 'ESimpleRSSException');
23256:   CL.AddTypeS('TLanguagesRSS', '( langAF, langSQ, langEU, langBE, langBG, langCA, '
23257:   + 'langZH_CN, langZH_TW, langHR, langCS, langDA, langNL, langNL_BE, langNL_NL'
23258:   + 'langEN, langEN_AU, langEN_BZ, langEN_CA, langEN_IE, langEN_JM, langEN_NZ'
23259:   + 'langEN_PH, langEN_ZA, langEN_TT, langEN_GB, langEN_US, langEN_ZW, langET'
23260:   + 'langFO, langFI, langFR, langFR_BE, langFR_CA, langFR_FR, langFR_LU, lang'
23261:   + 'FR_MC, langFR_CH, langGL, langGD, langDE, langDE_AT, langDE_DE, langDE_LI,'
23262:   + 'langDE_LU, langDE_CH, langEL, langHAW, langHU, langIS, langIN, langGA, la'
23263:   + 'ngIT, langIT_IT, langIT_CH, langJA, langKO, langMK, langNO, langPL, langPT'
23264:   + 'langPT_BR, langPT_PT, langRO, langRO_MO, langRO_RP, langRU, langRU_MO, l'
23265:   + 'angRU_RU, langSR, langSK, langSL, langES, langES_AR, langES_BO, langES_CL,'
23266:   + 'langES_CO, langES_CR, langES_DO, langES_EC, langES_SV, langES_GT, langES_'
23267:   + 'HN, langES_MX, langES_NI, langES_PA, langES_PY, langES_PE, langES_PR, lang'
23268:   + 'ES_ES, langES_UY, langES_VE, langSV, langSV_FI, langSV_SE, langTR, langUK, langX );
23269:   CL.AddTypeS('TXMLTypeRSS', '( xtRDFrss, xtrSSrss, xtAtomrss, xtiTunesrss );
23270:   TContentTypeRSS', '( ctTextrss, ctHTMLrss, ctXHTMLrss );
23271:   TEncodingTypeRSS', '( etBase64rss, etEscapedrss, etXMLrss );
23272:   SIRegister_TRFC822DateTime(CL);
23273:   SIRegister_TRSSChannelReq(CL);
23274:   SIRegister_TRSSImageReq(CL);
23275:   SIRegister_TRSSImageOpt(CL);
23276:   SIRegister_TRSSImage(CL);
23277:   SIRegister_TRSSCloud(CL);
23278:   SIRegister_TRSSTextInput(CL);
23279:   SIRegister_TRSSChannelCategory(CL);
23280:   SIRegister_TRSSChannelCategories(CL);
23281:   SIRegister_TRSSChannelSkipHours(CL);
23282:   SIRegister_TRSSChannelSkipDays(CL);
23283:   SIRegister_TRSSChannelOpt(CL);
23284:   SIRegister_TRSSChannel(CL);
23285:   SIRegister_TRSSItemSource(CL);
23286:   SIRegister_TRSSItemEnclosure(CL);
23287:   SIRegister_TRSSItemCategory(CL);
23288:   SIRegister_TRSSItemCategories(CL);
23289:   SIRegister_TRSSItemGUID(CL);
23290:   SIRegister_TRSSAuthor(CL);
23291:   SIRegister_TRSSItem(CL);
23292:   SIRegister_TRSSItems(CL);
23293:   CL.AddTypeS(TFormatSettingsRSS, 'record ShortDateFormat :Str; LongDateForm'
23294:   + 'at :Str; LongTimeFormat :Str; ShortTimeFormat :Str; end');
23295: end;
23296:
23297: RSS Feed Snippet:
23298: const RSS_NewsFeed = 'http://feeds.bbc.co.uk/news/world/rss.xml';
23299: with TSimpleRSS.create(self) do begin
23300:   XMLType:= xtRDFrss;
23301:   IndyHTTP:= TIdHTTP.create(self);
23302:   LoadFromHTTP(RSS_NewsFeed);
23303:   //LoadFromHTTP(Climatfeed);
23304:   writeln('RSSVersion: '+Version)
23305:   writeln('SimpleRSSVersion: '+SimpleRSSVersion)
23306:   for it:= 0 to items.count-1 do
23307:     writeln(ittoa(it)+' : '+Items[it].title+' : '+items[it].pubdate.getdatetime);
23308: end;
23309:
23310: (*-----*)
23311: Proc SIRegister_JTools(CL: TPSPascalCompiler);
23312: begin
23313:   Func ParseCmdLine( ACmdLine : PChar; List : TStrings; QuoteChar : char) :Bool';
23314:   Func GetCmdSwitchValue(const Switch:str;SwitchChars:TSysCharSet;var Value:str;IgnoreCase:boolean):bool;
23315:   //Proc ShowError( const S :Str; const Args : array of const);
23316: end;
23317:
23318: Proc SIRegister_rfc1213util(CL: TPSPascalCompiler);
23319: begin
23320:   CL.AddConstantN('rfc1213version','String').SetString('0.1.2');
23321:   Func UptimeToDays( const Uptime :Str) :Str';
23322:   Func getSNMP( const mib, community, host :Str; var ErrNo : Integer) :Str';
23323:   Func sysServicesString( const sysServicesStr :Str) :Str';
23324:   Func ipForwardingString( const ipForwarding :Str) :Str';
23325:   Func ipRouteTypeString( const ipRouteType :Str) :Str';
23326:   Func ipRouteProtoString( const ipRouteProto :Str) :Str';
23327:   Func ipNetToMediaTypeString( const ipNetToMediaType :Str) :Str';

```

```

23328: Func GetColFromTableRow( const EntryCols : TStringList; index : integer) :Str');
23329: Func ifTypeString( ifTypeStringNo :Str) :Str;
23330: Func ifStatusString( ifStatusStringNo :Str) :Str;
23331: Func tcpRtoAlgorithmString( const tcpRtoAlgorithm :Str) :Str;
23332: Func tcpConnStateString( const tcpConnState :Str) :Str;
23333: Func egpNeighStateString( const egpNeighState :Str) :Str;
23334: Func egpNeighModeString( const egpNeighMode :Str) :Str;
23335: Func egpNeighEventTriggerString( const egpNeighEventTrigger :Str) :Str;
23336: Func snmpEnableAuthenTrapsString( const snmpEnableAuthenTraps :Str) :Str;
23337: Func FormatMac( const Mac :Str) :Str;
23338: Proc JumpToRegistryKey(Key:Str);
23339: end;
23340:
23341: Proc SIRegister_neuralab(CL: TPSPascalCompiler);
23342: begin
23343:   SIRegister_TABHash(CL);
23344:   Func ABKey( S : array of byte; Divisor : longint) : longint');
23345:   Func ABCmp( var X, Y : array of byte) :Bool');
23346:   Func ABGetNext1( var AB : array of byte; ST : word) : word');
23347:   Func ABCountDif( var X, Y : array of byte) : longint');
23348:   Func ABCountDifZero( var X : array of byte) : longint');
23349:   Proc ABAnd( var A, B : array of byte);
23350:   Func ABGetEqual( var Equal, X, Y : array of byte) : longint');
23351:   Proc ABShiftLogicalLeft( var X : array of byte);
23352:   Proc ABShiftLogicalRight( var X : array of byte);
23353:   Func ABGetDif( var Dif, X, Y : array of byte) : longint');
23354:   Func ABToString( var AB : array of byte) :Str');
23355:   Func ABToStringR( var AB : array of byte) :Str');
23356:   Proc ABClear( var AB : array of byte);
23357:   Proc ABFull( var AB : array of byte);
23358:   Proc ABBitOnPos( var AB : array of byte; POS : longint);
23359:   Proc ABBitOnPosAtPos( var AB : array of byte; X, Start, Len : longint);
23360:   Func ABReadBitOnPosAtPos( var AB : array of single; Start, Len:longint) : longint');
23361:   Proc ABCopy( var A, B : array of byte);
23362:   Proc ABTriPascal( var A, B : array of byte);
23363:   Proc ABSet( var A : array of byte; B : array of byte);
23364: end;
23365:
23366: Proc SIRegister_neuralbit(CL: TPSPascalCompiler);
23367: begin
23368:   //CL.AddTypeS('TArrOf2BytesPtr', '^TArrOf2Bytes // will not work');
23369:   //CL.AddTypeS('TArrOf3BytesPtr', '^TArrOf3Bytes // will not work');
23370:   //CL.AddTypeS('TArrOf4BytesPtr', '^TArrOf4Bytes // will not work');
23371:   //CL.AddTypeS('TArrBytePtr', '^TLongByteArray // will not work');
23372:   Func POT( numero, elevado : extended) : extended');
23373:   Func LongintBitTest( Data : longint; P : longint) :Bool');
23374:   Func LongintBitFlip( Data : longint; P : longint) : longint');
23375:   Proc BAClear( var VARS : array of byte);
23376:   Proc BAMake1( var VARS : array of byte);
23377:   Func BARead( var A : array of byte; P : longint) : byte');
23378:   Proc BAFlip( var A : array of byte; P : longint);
23379:   Proc BAWrite( var A : array of byte; P : longint; Data : byte);
23380:   Func BATest( var A : array of byte; P : longint) :Bool');
23381:   Proc BASum( var x, y : array of byte);
23382:   Proc BASub( var x, y : array of byte);
23383:   Proc BAIncPos( var x : array of byte; POS : longint);
23384:   Proc BADecPos( var x : array of byte; POS : longint);
23385:   Proc BAInc( var x : array of byte);
23386:   Proc BADec( var x : array of byte);
23387:   Func BAToString( VARS : array of byte) :Str');
23388:   Func BAToFloat( var VARS : array of byte) : extended');
23389:   Proc PFloatToBA( var VARS : array of byte; Valor : extended);
23390:   Proc BANot( var VARS : array of byte);
23391:   Proc BAAnd( var r, x, y : array of byte);
23392:   Proc BAOOr( var r, x, y : array of byte);
23393:   Proc BAXOr( var r, x, y : array of byte);
23394:   Func BAGrater( var x, y : array of byte) :Bool;
23395:   Func BALower( var x, y : array of byte) :Bool;
23396:   Func BAEqual( var x, y : array of byte) :Bool;
23397:   Proc BAPMul( var r, x, y : array of byte);
23398:   Func nnRAnd( A, B : extended) : extended;
23399:   Func nnROr( A, B : extended) : extended;
23400:   Func nnRNot( A : extended) : extended;
23401:   Func nnRXor( A, B : extended) : extended;
23402:   Func REqual( A, B : extended) : extended;
23403:   Proc RSum( x, y, z : extended; var R, C : extended);
23404:   Proc RegSum( var x, y : array of extended);
23405:   Func RegEqual( var x, y : array of extended) : extended');
23406:   Func RegOrdEqual( var x, y : array of extended) : extended');
23407:   Func RegToString( var VARS : array of extended) :Str');
23408:   Func RORer( var VARS : array of extended) : extended');
23409:   Func RAnd( var VARS : array of extended) : extended');
23410:   Func RNot( X : extended; var VARS : array of extended) : extended');
23411:   Func RORMaxTerm( var VARS : array of extended; NumMaxTerm : longint) : extended');
23412:   Func RORMaxTermStr( NumVars : longint; NumMaxTerm : longint) :Str');
23413:   Func RSatFunc( var VARS : array of extended; NumFunc : longint) : extended');
23414:   Func RSatFuncStr( NumVars : longint; NumFunc : longint) :Str');
23415:   Proc RRegen( var VARS : array of extended);
23416:   Proc RDeGen( var VARS : array of extended);

```

```

23417: Proc RDegeN( var VARS : array of extended; P : extended);
23418: Proc nnClear( var VARS : array of extended);
23419: Proc BARAnd( var R, A, B : array of byte);
23420: Proc BAROr( var R, AUX, A, B : array of byte);
23421: Proc BARNot( var R, A : array of byte);
23422: end;
23423:
23424: (*-----*)
23425: Proc SIRegister_winsvc2(CL: TPSPascalCompiler);
23426: begin
23427:   CL.AddConstantN('SERVICE_AUTO_START', 'LongWord').SetUInt( $00000002);
23428:   CL.AddConstantN('SERVICE_CONFIG_DELAYED_AUTO_START_INFO', 'LongInt').SetInt( 3);
23429:   ('SERVICE_CONFIG_FAILURE_ACTIONS_FLAG', 'LongInt').SetInt( 4);
23430:   ('SERVICE_CONFIG_PREFERRED_NODE', 'LongInt').SetInt( 9);
23431:   ('SERVICE_CONFIG_PRESHUTDOWN_INFO', 'LongInt').SetInt( 7);
23432:   ('SERVICE_CONFIG_REQUIRED_PRIVILEGES_INFO', 'LongInt').SetInt( 6);
23433:   ('SERVICE_CONFIG_SERVICE_SID_INFO', 'LongInt').SetInt( 5);
23434:   ('SERVICE_CONFIG_TRIGGER_INFO', 'LongInt').SetInt( 8);
23435:   ('SC_MANAGER_ALL_ACCESS', 'LongWord').SetUInt($F003F);
23436:   ('SERVICE_ALL_ACCESS', 'LongWord').SetUInt($F01FF);
23437:   Func ChangeServiceType( ServiceName :Str; TypeID : DWord) :Bool;
23438:   Func GetServiceStatus2( ServiceName :Str; ErrorState :Bool) :Bool;
23439:   Func StartService2( ServiceName :Str) :Bool;
23440:   Func StopService2( ServiceName :Str) :Bool;
23441: end;
23442:
23443: Proc SIRegister_neuralbyteprediction(CL: TPSPascalCompiler);
23444: begin
23445:   CL.AddTypeS('TNeuralCountings', 'array of longint');
23446:   SIRegister_TNeuronGroupBase(CL);
23447:   SIRegister_TNeuronGroup(CL);
23448:   CL.AddTypeS('TNeuralNetwork', 'array of TNeuronGroup');
23449:   SIRegister_TStatePredictionClass(CL);
23450:   SIRegister_TLabeledState(CL);
23451:   SIRegister_TClassifier(CL);
23452:   SIRegister_TEasyLearnAndPredictClass(CL);
23453: end;
23454:
23455: Proc SIRegister_neuralcache(CL: TPSPascalCompiler);
23456: begin
23457:   CL.AddConstantN('NeuralMaxStates', 'LongInt').SetInt( 400000);
23458:   CL.AddTypeS('TNeuralState', 'TBytes');
23459:   TProcPred, 'Proc ( var ST : TBytes; Acao : byte);
23460:   SIRegister_TCacheMem(CL);
23461: end;
23462:
23463: (*-----*)
23464: Proc SIRegister_USolarSystem(CL: TPSPascalCompiler);
23465: begin
23466:   CL.AddTypeS('TOrbitalElements', record N:Double; i:Double; w:Double; a:Double; e:Double; M:Double; end;
23467:   TOrbitalElementsData, 'record NConst : Double; NVar : Double; i'
23468:   +Const : Double; iVar : Double; wConst : Double; wVar : Double; aConst : Do'
23469:   +uble; aVar : Double; eConst : Double; eVar : Double; MConst : Double; MVar:Double; end');
23470:   CL.AddConstantN('cAUToKilometers', 'Extended').setExtended( 149.6e6);
23471:   CL.AddConstantN('cEarthRadius', 'LongInt').SetInt( 6400);
23472:   Func GMTDateTimeToJulianDay( const dt : TDateTime) : Double;
23473:   Func ComputeOrbitalElements(const oeData:TOrbitalElementsData;const d:Double): TOrbitalElements;
23474:   Func ComputePlanetPosition( const orbitalElements : TOrbitalElements) : TAffineVector;
23475:   Func ComputePlanetPosition1(const orbitalElementsData:TOrbitalElementsData;const d:Double):TAffineVector;;
23476: end;
23477:
23478: Proc SIRegister_TSearchAnagrams(CL: TPSPascalCompiler);
23479: begin
23480:   //with RegClassS(CL, 'tcontrol', 'TSearchAnagrams') do
23481:   with CL.AddClassN(CL.FindClass('tcontrol'), 'TSearchAnagrams') do begin
23482:     RegisterProperty('useabbrevs', 'boolean', iptrw);
23483:     RegisterProperty('useforeign', 'boolean', iptrw);
23484:     RegisterProperty('usecaps', 'boolean', iptrw);
23485:     RegisterProperty('tag', 'integer', iptrw);
23486:     Proc Init(newletters:str;NewMinLen,NewMaxLen:word;newa,newf,newc:bool;pubDic:TDic);
23487:     Func FindMissingLetter( const s :Str) : TTestWords;
23488:     Proc Findallwords( const s :Str; list : Tstrings);
23489:   end;
23490: end;
23491:
23492: unit U_Invertedtext;
23493: Proc InitInvertedText(canvas:TCanvas; pagewidth,pageheight:integer);
23494: Proc DrawInvertedText(Canvas:TCanvas;pagewidth,pageheight:integer;linenbr:ints:str);
23495: Proc MemoTextFixUp(memo:TMemo {var text:str});
23496:
23497: unit hashunit
23498: Proc SIRegister_THashStr(CL: TPSPascalCompiler);
23499: begin
23500:   //with RegClassS(CL, 'TObject', 'THashStr') do
23501:   with CL.AddClassN(CL.FindClass('TObject'), 'THashStr') do begin
23502:     y('test', 'array of THashObject', iptrw);
23503:     y('maxhash', 'integer', iptrw);
23504:     y('maxloadfactor', 'single', iptrw);
23505:     y('maxused', 'integer', iptrw);

```

```

23506:   y('used', 'integer', iptrw);
23507:   y('nbrcollisions', 'integer', iptrw);
23508:   y('maxcollisions', 'integer', iptrw);
23509:   Constructor create( newmaxhash : integer; newmaxloading : single);
23510:   Proc Free');
23511:   Func exists( s :Str; var T : TObjet ) :Bool');
23512:   Func AddIfNotDup( s :Str; t : Tobjet ) :Bool');
23513:   Proc resethash');
23514:   Proc rehash');
23515:   Func hash( s :Str ) : integer');
23516: end;
23517: end;
23518:
23519: Proc SortString(var SortSt : Ansistr);
23520: SortSt:= 'this is maXbox4'; SortString2(SortSt); >>> 4Xabhiimosstx
23521: * === compile-time registration functions === *)
23522: (*-----*)
23523: Proc SIRegister_JsonsUtilsEx(CL: TPSPascalCompiler);
23524: begin
23525:   Func FixedFloatToStr( const Value : Extended) :Str');
23526:   Func FixedTryStrToFloat( const S :Str; out Value : Extended) :Bool');
23527:   Func FixedStrToFloat( const S :Str) : Extended');
23528:   Func __ObjectToJson( aObjet : TObjet ) :Str');
23529:   Proc __jsonToObject( const aJSONString :Str; var aObjet : TObjet);
23530:   CL.AddTypeS('TObjectDynArray', 'array of TObjet');
23531:   TStringDynArray2', 'array of string');
23532:   TIntegerDynArray2', 'array of Integer');
23533:   CL.AddConstantN('GLB_JSON_STD_DECIMALSEPARATOR', 'String').SetString( '.');
23534: end;
23535:
23536: Proc SIRegister_TJson(CL: TPSPascalCompiler);
23537: begin
23538:   //with RegClassS(CL, 'TJsonBase', 'TJson') do
23539:   with CL.AddClassN(CL.FindClass('TJsonBase'), 'TJson') do begin
23540:     Constructor Create');
23541:     Proc Free');
23542:     Proc Parse( JsonString :Str);
23543:     Func Stringify :Str');
23544:     Proc Assign( Source : TJsonBase);
23545:     Proc Delete21( const Index : Integer);
23546:     Proc Delete22( const Name :Str);
23547:     Proc Clear');
23548:     Func Get23( const Index : Integer) : TJsonValue;
23549:     Func Get24( const Name :Str) : TJsonValue;
23550:     Func Put25( const Value : TJsonEmpty) : TJsonValue;
23551:     Func Put26( const Value : TJsonNull) : TJsonValue;
23552:     Func Put27( const Value :Bool) : TJsonValue;
23553:     Func Put28( const Value : Integer) : TJsonValue;
23554:     Func Put29( const Value : Extended) : TJsonValue;
23555:     Func Put30( const Value :Str) : TJsonValue;
23556:     Func Put31( const Value : TJsonArray2) : TJsonValue;
23557:     Func Put32( const Value : TJsonObject2) : TJsonValue;
23558:     Func Put33( const Value : TJsonValue) : TJsonValue;
23559:     Func Put34( const Value : TJson) : TJsonValue;
23560:     Func Putbool( const Value :Bool) : TJsonValue; ----> alias
23561:     Func Putint( const Value : Integer) : TJsonValue;
23562:     Func Putext( const Value : Extended) : TJsonValue;
23563:     Func Putstr( const Value :Str) : TJsonValue;
23564:     Func Putarr( const Value : TJsonArray2) : TJsonValue;
23565:     Func Putobj( const Value : TJsonObject2) : TJsonValue;
23566:     Func Putval( const Value : TJsonValue) : TJsonValue;
23567:     Func Putjson( const Value : TJson) : TJsonValue;
23568:     Func Put35( const Name :Str; const Value : TJsonEmpty) : TJsonValue;
23569:     Func Put36( const Name :Str; const Value : TJsonNull) : TJsonValue;
23570:     Func Put37( const Name :Str; const Value :Bool) : TJsonValue;
23571:     Func Put38( const Name :Str; const Value : Integer) : TJsonValue;
23572:     Func Put39( const Name :Str; const Value : Extended) : TJsonValue;
23573:     Func Put40( const Name :Str; const Value :Str) : TJsonValue;
23574:     Func Put41( const Name :Str; const Value : TJsonArray2) : TJsonValue;
23575:     Func Put42( const Name :Str; const Value : TJsonObject2) : TJsonValue;
23576:     Func Put43( const Name :Str; const Value : TJsonValue) : TJsonValue;
23577:     Func Put44( const Name :Str; const Value : TJson) : TJsonValue;
23578:     Func Put45( const Value : TJsonPair) : TJsonValue;
23579:     RegisterProperty('StructType', 'TJsonStructType', iptr);
23580:     'JsonObject', 'TJsonObject2', iptr);
23581:     'JsonArray', 'TJsonArray2', iptr);
23582:     'Count', 'Integer', iptr);
23583:     'Values', 'TJsonValue String', iptr);
23584:     SetDefaultProperty('Values');
23585:   end;
23586: end;
23587:
23588: (*-----*)
23589: Proc SIRegister_TJsonBase(CL: TPSPascalCompiler);
23590: begin
23591:   //with RegClassS(CL, 'TObjet', 'TJsonBase') do
23592:   with CL.AddClassN(CL.FindClass('TObjet'), 'TJsonBase') do begin
23593:     Constructor Create( AOwner : TJsonBase);
23594:     Proc Free');

```

```

23595: Proc Parse( jsonString :Str);
23596: Func Stringify :Str';
23597: Proc Assign( Source : TJsonBase);
23598: Func Encode( const S :Str) :Str';
23599: Func Decode( const S :Str) :Str';
23600: Proc Split( const S :Str; const Delimiter: Char; Strings:TStrings);
23601: Func IsJsonObject( const S :Str) :Bool';
23602: Func IsJsonArray( const S :Str) :Bool';
23603: Func IsJsonString( const S :Str) :Bool';
23604: Func IsJsonNumber( const S :Str) :Bool';
23605: Func IsJsonBoolean( const S :Str) :Bool';
23606: Func IsJsonNull( const S :Str) :Bool';
23607: Func AnalyzeJsonValue( const S :Str) : TJsonValue;
23608: RegisterProperty('Owner', 'TJsonBase', iptr);
23609: end;
23610: end;
23611:
23612: (*-----*)
23613: Proc SIRegister_Jsons(CL: TPSPascalCompiler);
23614: begin
23615:   TJsonValue(' ( jvNone, jvNull, jvString, jvNumber, jvBoolean, jvObject, jvArray );
23616:   TJsonStructType(' ( jsNone, jsArray, jsObject );
23617:   //CL.AddTypeS('TJsonNull', ' ( null );
23618:   //CL.AddTypeS('TJsonEmpty', ' ( empty );
23619:   SIRegister_TJsonBase(CL);
23620:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TJsonObject2');
23621:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TJsonArray2');
23622:   SIRegister_TJsonValue(CL);
23623:   SIRegister_TJsonArray2(CL);
23624:   SIRegister_TJsonPair(CL);
23625:   SIRegister_TJsonObject2(CL);
23626:   SIRegister_TJson(CL);
23627: end;
23628: var job:= ajt.JsonObject;
23629: for cnt:= 2 to job.count-2 do begin
23630:   Clabel:= job.items[cnt].name;
23631:   writeln('iterate '+clabel)
23632:   JsArr:= job.values[clabel].asArray;
23633:   for cnt2:= 0 to jsarr.count-1 do
23634:     jsobj:= jsarr.items[cnt2].asobject;
23635:     for cnt3:= 0 to jsobj.count do
23636:       writeln(jsobj['elements'].asarray[0].asobject.items[cnt3].name)
23637:     end;
23638:   end;
23639: writeln('elements status:
'+ajt['rows'].asarray[0].asobject['elements'].asarray[0].asobject['status'].asstring);
23640:
23641: Proc SIRegister_Bricks(CL: TPSPascalCompiler);
23642: begin
23643:   CL.AddConstantN('kError101', 'String').SetString( 'Runtime error: Negative parameter(s);
23644:   kError102', 'String').SetString( 'Runtime error: Parameter(s) out of range');
23645:   kError103', 'String').SetString( 'Runtime error: min > max');
23646:   kError104', 'String').SetString( 'Runtime error: max = 0');
23647:   kError105', 'String').SetString( 'Runtime error: Denominator is zero');
23648:   CL.AddTypeS('TVectorE', 'array of extended');
23649:   complexreal', 'record re : real; im : real; end');
23650:   TFR', 'record M : extended; phi : extended; F : complexreal; end');
23651:   //CL.AddTypeS('complexreal', 'record re : real; im : real; end');
23652:   SIRegister_TBlock(CL);
23653:   SIRegister_TP2(CL);
23654:   SIRegister_TPT0(CL);
23655:   SIRegister_TPT1(CL);
23656:   SIRegister_TPT2(CL);
23657:   SIRegister_TInt2(CL);
23658:   SIRegister_TIT1(CL);
23659:   SIRegister_TDT1(CL);
23660:   SIRegister_TIT2(CL);
23661:   SIRegister_TPAdd(CL);
23662:   SIRegister_TPSub(CL);
23663:   SIRegister_TPMul(CL);
23664:   SIRegister_TPDIV(CL);
23665: end;
23666:
23667: Proc SIRegister_lifeblocks(CL: TPSPascalCompiler);
23668: begin
23669:   SIRegister_TASIA(CL);
23670:   SIRegister_TMiMe(CL);
23671:   SIRegister_TNoCoDI(CL);
23672: end;
23673:
23674: *****
23675: Release Notes maXbox 4.7.4.64 June 2020 mX47
23676: *****
23677: 1254 unit uPSI_MaskEdit.pas FCL
23678: 1255 unit uPSI_SimpleRSSTypes; BlueHippo
23679: 1256 unit uPSI_SimpleRSS; BlueHippo
23680: 1257 unit uPSI_psULib.pas Prometheus
23681: 1258 unit uPSI_psUFinancial; Prometheus
23682: 1259 uPSI_PsAPI_2.pas mX4

```



```

23683: 1260 uPSI_PersistSettings_2      mX4
23684: 1261 uPSI_rfc1213util2.pas        IP
23685: 1262 uPSI_JTools.pas               JCL
23686: 1263 unit uPSI_neuralbit.pas        CAI
23687: 1264 unit uPSI_neuralab.pas         CAI
23688: 1265 unit uPSI_winsvc2.pas          TEK
23689: 1266 unit uPSI_wmiserv2.pas          TEK
23690: 1267 uPSI_neuralcache.pas           CAI
23691: 1268 uPSI_neuralbyteprediction      CAI
23692: 1269 unit uPSI_USolarSystem;         glscene.org
23693: 1270 uPSI_USearchAnagrams.pas        DFF
23694: 1271 uPSI_JsonsUtilsEx.pas           Randolph
23695: 1272 unit uPSI_Jsons.pas             Randolph
23696: 1273 unit uPSI_HashUnit;             DFF
23697: 1274 uPSI_U_Invertedtext.pas         DFF
23698: 1275 unit uPSI_Bricks;               Dendron
23699: 1276 unit uPSI_lifeblocks.pas         Dendron
23700:
23701: Totals of Func Calls: 32633
23702: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
23703: CRC32: 3EB27A87: 28.2 MB (29,608,248) bytes
23704: Totals of Func Calls: 32614
23705:
23706: Functions_max hex in the box maxbox
23707: functionslist.txt
23708: FunctionsList1
23709: 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
23709:
23710: Proc SIRegister_JclStringLists(CL: TPSPascalCompiler);
23711: begin
23712:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclStringListError');
23713:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IJclStringList,'IJclStringList');
23714:   CL.AddTypeS('TJclStringListObjectsMode','( omNone, omObjects, omVariants, omInterfaces );
23715:   SIRegister_IJclStringList(CL);
23716:   SIRegister_TJclInterfacedStringList(CL);
23717:   SIRegister_TJclStringList(CL);
23718:   Func JclStringList : IJclStringList;
23719:   Func JclStringListStrings( AStrings : TStrings ) : IJclStringList;
23720:   Func JclStringListStrings1( const A : array of string ) : IJclStringList;
23721:   Func JclStringList1( const A : array of const ) : IJclStringList;
23722:   Func JclStringList2( const AText :Str ) : IJclStringList;
23723: end;
23724:
23725: (*-----*)
23726: Proc SIRegister_cInternetUtils(CL: TPSPascalCompiler);
23727: begin
23728:   ('ctHTML','String').SetString( 'text/html');
23729:   ('ctText','String').SetString( 'text/plain');
23730:   ('ctXML','String').SetString( 'text/xml');
23731:   ('ctJPG','String').SetString( 'image/jpeg');
23732:   ('ctGIF','String').SetString( 'image/gif');
23733:   ('ctBMP','String').SetString( 'image/bmp');
23734:   ('ctPNG','String').SetString( 'image/png');
23735:   ('ctTIFF','String').SetString( 'image/tiff');
23736:   ('ctMPG','String').SetString( 'video/mpeg');
23737:   ('ctAVI','String').SetString( 'video/avi');
23738:   ('ctQT','String').SetString( 'video/quicktime');
23739:   ('ctBinary','String').SetString( 'application/binary');
23740:   ('ctPDF','String').SetString( 'application/pdf');
23741:   ('ctPostscript','String').SetString( 'application/postscript');
23742:   ('ctBasicAudio','String').SetString( 'audio/basic');
23743:   ('ctMP3','String').SetString( 'audio/mpeg');
23744:   ('ctRA','String').SetString( 'audio/x-realaudio');
23745:   ('ctURLEncoded','String').SetString( 'application/x-www-form-urlencoded');
23746:   ('ctZIP','String').SetString( 'application/zip');
23747:   ('ctJavaScript','String').SetString( 'application/javascript');
23748:   ('ctPascal','String').SetString( 'text/x-source-pascal');
23749:   ('ctCPP','String').SetString( 'text/x-source-cpp');
23750:   ('ctINI','String').SetString( 'text/x-windows-ini');
23751:   ('ctBAT','String').SetString( 'text/x-windows-bat');
23752:   Func flcMIMEContentTypeFromExtention( const Extention :Str):Str;
23753:   ('protoHTTP','String').SetString( 'http');
23754:   ('protoNNTP','String').SetString( 'news');
23755:   ('protoFTP','String').SetString( 'ftp');
23756:   ('protoGopher','String').SetString( 'gopher');
23757:   ('protoEMail','String').SetString( 'mailto');
23758:   ('protoHTTPS','String').SetString( 'https');
23759:   ('protoIRC','String').SetString( 'irc');
23760:   ('protoFile','String').SetString( 'file');
23761:   ('protoTelnet','String').SetString( 'telnet');
23762:   Proc flcDecodeURL( const URL :Str; var Protocol, Host, Path :Str);
23763:   Func flcEncodeURL( const Protocol, Host, Path :Str ) :Str;
23764:   Proc flcDecodeHost(const Address:str;var Host,Port:str;const DefaultPort:str);
23765:   Func flcEncodeDotLineTerminated( const S : Ansistr ) : Ansistr;
23766:   Func flcEncodeEmptyLineTerminated( const S : Ansistr ) : Ansistr;
23767:   Func flcDecodeDotLineTerminated( const S : Ansistr ) : Ansistr;
23768:   Func flcDecodeEmptyLineTerminated( const S : Ansistr ) : Ansistr;
23769:   //CL.AddDelphiFunction('Proc StreamDotLineTerminated0(const Source, Destination: AStream; const
ProgressCallback:TCopyProgressProcedure);

```

```

23770: //CL.AddDelphiFunction('Proc StreamDotLineTerminated1(const Source:str;const Destination:AStream;const
ProgressCallback:TCopyProgressProcedure);
23771: Func flchtmlCharRef( const CharVal : LongWord; const UseHex :Bool):Str');
23772: Func flchtmlSafeAsciiText( const S :Str) :Str');
23773: Proc flchtmlSafeWideText( var S : WideString);
23774: Func flchtmlSafeQuotedText( const S :Str) :Str');
23775: Func flcEncodeHeaderField( const Name, Body :Str) :Str');
23776: Func flcDecodeHeaderField( const S :Str; var Name, Body :Str) :Bool');
23777: Proc flcDecodeEmailAddress( const S :Str; var User, Domain :Str);
23778: Proc flcDecodeEMailField( const S :Str; var EMailAddress, Name :Str);
23779: Func flcDateFieldBody :Str');
23780: Func flcDateField :Str');
23781: Func flcMessageIDFieldBody( const ID :Str; const Host :Str) :Str');
23782: SIRegister_AHeaderField(CL);
23783: //CL.AddTypeS('AHeaderFieldClass', 'class of AHeaderField');
23784: CL.AddTypeS('AHeaderFieldArray', 'array of AHeaderField');
23785: SIRegister_THeaderCls(CL);
23786: //CL.AddTypeS('THeaderClass', 'class of THeader');
23787: CL.AddTypeS('THeaderArray', 'array of THeaderCls');
23788: SIRegister_THeaderField(CL);
23789: SIRegister_TInvalidField(CL);
23790: SIRegister_TDateFieldClass(CL);
23791: SIRegister_TEMailField(CL);
23792: CL.AddDelphiFunction('Proc SelfTestInternetUtils');
23793: maxform1.JumptoOutput1Click(self)
23794: end;
23795:
23796: Proc SIRegister_cWindows(CL: TPSPascalCompiler);
23797: Func InvMod(const A, N: Integer): Integer;
23798: Func ExpMod(A, Z: Integer; const N: Integer): Integer;
23799: Func CopyFrom(const S:Str; const Index: Integer):Str;
23800: Func CopyEx(const S:Str; const Start, Count: Integer):Str;
23801: Func loadForm2(vx, vy: smallint; acolor: TColor; aname:Str): TForm;
23802: PlaySound('INDIGO WAV', hInstance, SND_RESOURCE or SND_ASYNC);
23803: Proc flcRaiseLastOSError';
23804: Func flcStrEqualNoCase( const A,B:Str; const AsciiCaseSensitive :Bool):Bool');
23805: Func flcStrMatchNoAsciiCase( const S, M :Str; const Index : Integer) :Bool');
23806: Func flcStrMatchLeft( const S, M :Str; const AsciiCaseSensitive :Bool):Bool');
23807: Func flcStrMatchRight( const S,M:Str; const AsciiCaseSensitive :Bool) :Bool');
23808: Func flcPosChar( const F : Char; const S :Str; const Index : Integer) : Integer');
23809: Func flcPosStr(const F,S:Str;const Index:Integer;const AsciiCaseSensitive:Bool):nteger;
23810: Func flcStrMatch( const S, M :Str; const Index : Integer) :Bool');
23811: Func flcStrPMatch( const A, B : PChar; const Len : Integer) :Bool');
23812: Func flcflcCopyFrom( const S :Str; const Index : Integer) :Str');
23813: Func flcCopyRange( const S :Str; const StartIndex, StopIndex : Integer) :Str');
23814: Func flcCopyLeft( const S :Str; const Count : Integer) :Str');
23815: Func flcCopyRight( const S :Str; const Count : Integer) :Str');
23816: Func flcStrIsNumeric( const S :Str) :Bool');
23817: Func flcStrMatchChar( const S :Str; const M : TSYsCharSet) :Bool');
23818: Func flcStrSplitAtChar(const S:Str;const C:Char;var Left,Right:Str const Option1:Bool):Bool;
23819: Func flcStrReplaceChar( const Find, Replace : Char; const S :Str) :Str');
23820: Func flcStrInclSuffix(const S:Str;const Suffix:Str;const AsciiCaseSensitive:Bool):str;
23821: Func flcStrRemoveCharDelimited(var S:str;const FirstDelim,SecondDelim:Char) :Str');
23822: Func flcStrSplit( const S, D :Str) : TStringArray');
23823: Proc flcTrimInPlace( var S :Str; const C : TSYsCharSet);
23824: Func flcTrimIn( S :Str; const C : TSYsCharSet) :Str');
23825: Func flcStrAfterCharSet( const S :Str; const D : TSYsCharSet) :Str');
23826: Func flcStrAfterChar( const S :Str; const D : Char) :Str');
23827: Proc StrSplit(Delimiter: Char; Str:Str; ListOfStrings: TString) ;
23828: (*-----*)
23829: Proc SIRegister_cWindows(CL: TPSPascalCompiler);
23830: begin
23831: CL.AddTypeS('TWindowsVersionflc', '( Win16_31, Win32_95, Win32_95R2, Win32_98, W'
23832: '+in32_98SE, Win32_ME, Win32_Future, WinNT_31, WinNT_35, WinNT_351, WinNT_40'
23833: '+', WinNT5_2000, WinNT5_XP, WinNT5_2003, WinNT5_Future, WinNT_Future, Win_Future );
23834: CL.AddTypeS('TWindowsVersions', 'set of TWindowsVersionflc');
23835: Func flcGetWindowsVersion : TWindowsVersionflc);
23836: Func flcIsWinPlatform95 :Bool');
23837: Func flcIsWinPlatformNT :Bool');
23838: Func flcGetWindowsProductID :Str');
23839: Func flcGetWindowsTemporaryPath :Str');
23840: Func flcGetWindowsPath :Str');
23841: Func flcGetWindowsSystemPath :Str');
23842: Func flcGetProgramFilesPath :Str');
23843: Func flcGetCommonFilesPath :Str');
23844: Func flcGetApplicationPath :Str');
23845: Func flcGetUserName :Str');
23846: Func flcGetLocalComputerName :Str');
23847: Func flcGetLocalHostName :Str');
23848: CL.AddTypeS('TVersionInfoflc', (viFileVersion,viFileDescription,viLegalCopyrght,viComments,viCompanyName,
viInternalNme,viLegalTrademarks,viOriginalFilename,viProductNme,viProductVersion;
23849: Func flcGetAppVersionInfo( const VersionInfo : TVersionInfoflc) :Str');
23850: Func flcWinExecute(const ExeName,Params:str;const ShowWin:Word;const Wait:Bool):Bool);
23851: Func flcGetEnvironmentStrings : TStringArray');
23852: Func flcContentTypeFromExtention( Extention :Str) :Str');
23853: Func flcIsApplicationAutoRun( const Name :Str) :Bool');
23854: Proc flcSetApplicationAutoRun( const Name :Str; const AutoRun :Bool);
23855: Func flcGetWinPortNames : TStringArray');
23856: Func flcGetKeyPressed( const VKeyCode : Integer) :Bool');

```

```

23857: Func flcGetHardDiskSerialNumber( const DriveLetter : Char) :Str');
23858: Func flcReboot :Bool');
23859: //CL.AddDelphiFunction('Func ConvertThreadToFiber( lpParameter : Pointer) : Pointer');
23860: //Func CreateFiber(dwStackSize:DWORD;lpStartAddress:TFNfiberStartRoutine;lpParameter:Ptr):Ptr');
23861: CL.AddTypeS('TIEProxy', '( iepHTTP, iepHTTPS, iepFTP, iepGOPHER, iepSOCKS );
23862: CL.AddDelphiFunction('Func GetIEProxy( const Protocol : TIEProxy) :Str');
23863: CL.AddTypeS('TWindowHandleMessageEvent', 'Func( const Msg:Card;const wParam,lParam:Int;var
Handled:Bool):Int;
23864: CL.AddClassN(CL.FindClass('TOBJECT'),'TWindowHandle');
23865: TWindowHandleEvent', 'Proc ( const Sender : TWindowHandle);
23866: (TWindowHandleErrorEvent', 'Procedure( const Sender:TWindowHandle;const E:Exception);
23867: SIRegister_TWindowHandle(CL);
23868: CL.AddClassN(CL.FindClass('TOBJECT'),'EWindowHandle');
23869: SIRegister_TfndWindowHandle(CL);
23870: CL.AddClassN(CL.FindClass('TOBJECT'),'TTimerHandle');
23871: TTimerEventflc', 'Proc ( const Sender : TTimerHandle);
23872: SIRegister_TTimerHandle(CL);
23873: SIRegister_TfndTimerHandle(CL);
23874: Proc flcRaiseLastOSError');
23875: Func flcStrEqualNoCase(const A,B:Str; const AsciiCaseSensitive :Bool):Bool');
23876: Func flcStrMatchNoAsciiCase( const S, M :Str; const Index : Integer) :Bool');
23877: Func flcStrMatchLeft( const S, M :Str; const AsciiCaseSensitive :Bool):Bool');
23878: Func flcStrMatchRight(const S,M :Str; const AsciiCaseSensitive :Bool):Bool');
23879: Func flcPosChar( const F : Char; const S :Str; const Index : Integer): Integer');
23880: Func flcPosStr(const F,S:str;const Index:Integer;const AsciiCaseSensitive:Bool):Integer;
23881: Func flcStrMatch( const S, M :Str; const Index : Integer):Bool');
23882: Func flcStrPMatch( const A, B : PChar; const Len : Integer):Bool');
23883: Func flcflcCopyFrom( const S :Str; const Index : Integer):Str');
23884: Func flcCopyRange( const S :Str; const StartIndex, StopIndex :Integer):str');
23885: Func flcCopyLeft( const S :Str; const Count : Integer):Str');
23886: Func flcCopyRight( const S :Str; const Count : Integer):Str');
23887: Func flcStrIsNumeric( const S :Str) :Bool');
23888: Func flcStrMatchChar( const S :Str; const M : TSYsCharSet) :Bool');
23889: Func flcStrSplitAtChar(const S Str;const C:Char;var Left,Right:Str;const Optional:Bool):Bool
23890: Func flcStrReplaceChar( const Find, Replace : Char; const S:Str) :Str');
23891: Func flcStrInclSuffix(const S:Str;const Suffix:Str;const AsciiCaseSensitive:Bool):str;
23892: Func flcStrRemoveCharDelimited(var S:str;const FirstDelim,SecondDelim:Char) :Str');
23893: Func flcStrSplit( const S, D :Str) : TStringArray');
23894: Proc flcTrimInPlace( var S :Str; const C : TSYsCharSet);
23895: Func flcTrimIn( S :Str; const C : TSYsCharSet) :Str');
23896: Func flcStrAfterCharSet( const S :Str; const D : TSYsCharSet) :Str');
23897: Proc StrSplit(Delimiter: Char; Str:Str; ListOfStrings: TStrings) ;
23898: Func StrSplitF(Delimiter: Char; Str:Str): TStringlist;
23899: Func Int32toBytes(const Cint: Integer): TBytes;
23900: Proc FGPlayASound(const AResName:Str);
23901: //CL.AddDelphiFunction('Proc FGPlayASound(const AResName:Str);
23902: Proc FGPlayASound2(const AResName:Str; atype:Str);
23903: Proc PlayResWav(const AResName:Str; atype:Str);
23904: Proc PlayResWav2(const AResName:Str; atype:Str); //sync tick1.wav
23905: procedure PlaySoundStreamSync(const AName: TMemoryStream; atype: string);
23906: procedure PlaySoundStreamAsync(const AName: TMemoryStream; atype: string);
23907: procedure PlaySoundStreamAsyncNoStop(const AName: TMemoryStream; atype: string);
23908: Func queryPerformanceCounter2(var mse: int64):Bool;
23909: Func AddQuantumToDateTime(const dt: TDateTime): TDateTime;
23910: Func AddQuantumToDateTime(const dt: TDateTime): TDateTime;
23911: var overlay: Int64 absolute Result;
23912: begin
23913: Result:= dt;
23914: overlay:= overlay+1;
23915: end;
23916: Func GetRTFText(Const RichEdit: TRichEdit):Str;
23917: Func StrRemoveCharSet(const S:Str; const C: CharSet):Str;
23918: Func GetTextBetween(const Start, Stop:Str;var S, Between:Str):Boolean;
23919: end;
23920:
23921: Proc SIRegister_flcSysUtils(CL: TPSPascalCompiler);
23922: begin
23923: SIRegister_TfreqObj(CL);
23924: CL.AddTypeS('MSArray','array[0..1] of tmemorystream;
23925: Func GetLastOSErrorCode : NativeInt');
23926: Func GetLastOSErrorMessage :Str');
23927: Proc TFormlmsPlaySound( MS : MSArray; streaminuse : integer; aloop:boolean);
23928: Proc BytesSetLengthAndZero( var V : TBytes; const NewLength : NativeInt);
23929: Proc BytesInit( var V : TBytes; const R : Byte);
23930: Proc BytesInit1( var V : TBytes; const S :Str);
23931: Func BytesAppend( var V : TBytes; const R : Byte) : NativeInt');
23932: Func BytesAppend1(var V : TBytes; const R : TBytes) : NativeInt');
23933: Func BytesAppend2(var V : TBytes; const R : array of Byte) : NativeInt');
23934: Func BytesAppend3(var V : TBytes; const R :Str) : NativeInt');
23935: Func BytesCompare(const A, B : TBytes) : Integer');
23936: Func BytesEqual( const A, B : TBytes) :Bool');
23937: Func StringRefCount2(const S: Ansistr): Integer;
23938: Func hton16(const A:Word): Word;
23939: Func ntoh16(const A:Word): Word;
23940: Func hton32(const A:LongWord): LongWord;
23941: Func ntoh32(const A:LongWord): LongWord;
23942: Func hton64(const A:Int64): Int64;
23943: Func ntoh64(const A:Int64): Int64;
23944: CL.AddTypeS('TPoint3D2', 'record x : single; y : single; z : single; end;

```

```

23945: Func SphereTPoint3D(Phi, Lambda: Double): TPoint3D2;
23946: Func RotateAroundXTPoint3D(const P: TPoint3D2; Alfa: Double): TPoint3D2;
23947: Func RotateAroundYTPoint3D(const P: TPoint3D2; Beta: Double): TPoint3D2;
23948: Func LinkerTimeStamp(const FileName: Str): TDateTime; overload;
23949: Func ByteCharDigitToInt(const A: Char): Integer;
23950: Func WideCharDigitToInt(const A: WideChar): Integer;
23951: Func CharDigitToInt(const A: Char): Integer;
23952: Func IntToByteCharDigit(const A: Integer): Char;
23953: Func IntToWideCharDigit(const A: Integer): WideChar;
23954: Func IntToCharDigit(const A: Integer): Char;
23955: Func flcMinInt(const A, B: Int64): Int64;
23956: Func flcMaxInt(const A, B: Int64): Int64;
23957: end;
23958:
23959: Proc SIRegister_RotImg(CL: TPSPascalCompiler);
23960: begin
23961:   SIRegister_TRotateImage(CL);
23962:   Func CreateRotatedBitmap(Bitmap: TBitmap; const Angle: Extended; bgColor: TColor): TBitmap;
23963: end;
23964:
23965: *-----*)
23966: Proc SIRegister_HSLUtils(CL: TPSPascalCompiler);
23967: begin
23968:   Func ARGB(a, r, g, b: Byte): Card';
23969:   Func GetAValue2( argb: DWORD ): Byte';
23970:   Func GetRValue2( argb: DWORD ): Byte';
23971:   Func GetGValue2( argb: DWORD ): Byte';
23972:   Func GetBValue2( argb: DWORD ): Byte';
23973:   Func GetABGRAValue(abgr: DWORD): Byte';
23974:   Func GetABGRRValue(abgr: DWORD): Byte';
23975:   Func GetABGRGValue(abgr: DWORD): Byte';
23976:   Func GetABGRBValue(abgr: DWORD): Byte';
23977:   Func HSLtoRGB2( H, S, L: double): TColor';
23978:   Func AHSLtoARGB( A, H, S, L: double): Card';
23979:   Func HSLRangeToRGB( H, S, L: integer): TColor';
23980:   Proc ARGBtoHSL( ARGB: TColor; var A, H, S, L: double);
23981:   Proc ABGRtoHSL( ABGR: TColor; var A, H, S, L: double);
23982:   Proc RGBtoHSLRange( RGB: TColor; var H, S, L: integer);
23983: end;
23984:
23985: Proc SIRegister_GraphicsMathLibrary(CL: TPSPascalCompiler);
23986: begin
23987:   gmlsizeUndefined('LongInt').SetInt( 1);
23988:   gmlsize2D('LongInt').SetInt( 3);
23989:   gmlsize3D('LongInt').SetInt( 4);
23990:   CL.AddClassN(CL.FindClass('TOBJECT'),'EVectorError');
23991:   CL.AddClassN(CL.FindClass('TOBJECT'),'EMatrixError');
23992:   gmlTAxis, '( axisX, axisY, axisZ );
23993:   gmlTCoordinate, '( coordCartesian, coordSpherical, coordCylindrical );
23994:   gmlTDimension, '( dimen2D, dimen3D );
23995:   gmlTIndex, 'Integer';
23996:   gmlTrotation, '( rotateClockwise, rotateCounterClockwise );
23997:   gmlTVector, 'record size: gmlTIndex; vector: ARRAY[1..4] OF DOUBLE; x,y,z,h:double; end;
23998:   gmlTMatrix, 'record size: gmlTIndex; matrix: ARRAY[1..4] of Array[1..4] OF DOUBLE; end;
23999:   Func gmlVector2D( const xValue, yValue: DOUBLE): gmlTVector';
24000:   Func gmlVector3D( const xValue, yValue, zValue: DOUBLE): gmlTVector';
24001:   Func gmlAddVectors(const u, v: gmlTVector): gmlTVector';
24002:   Func gmlTransform( const u: gmlTVector; const a: gmlTMatrix): gmlTVector';
24003:   Func gmlMatrix2D(const m11,m12,m13,m21,m22,m23,m31,m32,m33: DOUBLE): gmlTMatrix';
24004:   Func gmlMatrix3D(const m11,m12,m13,m14,m21,m22,m23,m24,m31,m32,m33,m34,m41,m42,m43,m44:DOUBLE):gmlTMatrix;
24005:   Func gmlMultiplyMatrices( const a, b: gmlTMatrix): gmlTMatrix';
24006:   Func gmlInvertMatrix( const a, b: gmlTMatrix; var determinant: DOUBLE): gmlTMatrix';
24007:   Func gmlRotateMatrix(const dimension:gmlTDimension;const xyz:gmlTAxis;const angle:DOUBLE;const rotation
24008:     :gmlTrotation):gmlTMatrix;
24009:   Func gmlScaleMatrix( const s: gmlTVector): gmlTMatrix';
24010:   Func gmlTranslateMatrix( const t: gmlTVector): gmlTMatrix';
24011:   Func gmlViewTransformMatrix(const coordinate:gmlTCoordinate;const azimuth,elevation,distance:DOUBLE;const
ScreenX,ScreenY,ScreenDistance:DOUBLE):gmlTMatrix';
24012:   Func gmlFromCartesian(const ToCoordinate:gmlTCoordinate;const u:gmlTVector): gmlTVector;
24013:   Func gmlToCartesian(const FromCoordinate: gmlTCoordinate;const u:gmlTVector): gmlTVector;
24014:   Func gmlToDegrees( const angle: DOUBLE): DOUBLE';
24015:   Func gmlToRadians( const angle: DOUBLE): DOUBLE';
24016:   Func gmlDefuzz( const x: DOUBLE): DOUBLE';
24017:   Func gmlGetFuzz: DOUBLE';
24018:   Proc gmlSetFuzz( const x: DOUBLE);
24019: end;
24020:
24021: Proc SIRegister_umodels(CL: TPSPascalCompiler);
24022: begin
24023:   CL.AddTypeS('TRegType', '( REG_LIN, REG_MULT, REG_POL, REG_FRAC, REG_EXPO, RE'
24024:     +'G_IEXPO, REG_EXLIN, REG_LOGIS, REG_POWER, REG_GAMMA, REG_MICH, REG_MINT, R'
24025:     +'EG_HILL, REG_PK, REG_EVAL );
24026:   //CL.AddTypeS('TModel', 'record [...] end');
24027:   Func FirstParam( Model: TModel): Integer';
24028:   Func LastParam( Model: TModel): Integer';
24029:   Func FuncName( Model: TModel): Str';
24030:   Func ParamName( Model: TModel; I: Integer): Str';
24031:   Func RegFunc( Model: TModel; X: Float; B: TVector): Float';
24032:   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);

```

```

24033: 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);
24034: end;
24035:
24036: (*-----*)
24037: Proc SIRegister_TStatisticClass(CL: TPSPascalCompiler);
24038: begin
24039:   //with RegClassS(CL,'TOBJECT', 'TStatisticClass') do
24040:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TStatisticClass') do begin
24041:     Proc Assign( const S : TStatisticClass);
24042:     Func Duplicate : TStatisticClass';
24043:     Proc Clear';
24044:     Func IsEqual( const S : TStatisticCLASS:Bool');
24045:     Proc Add( const V : MFloat);
24046:     Proc Add1( const V : array of MFloat);
24047:     Proc Add2( const V : TStatisticClass);
24048:     Proc AddNegated( const V : TStatisticClass);
24049:     Proc Negate';
24050:     Count', 'Integer', iptr);
24051:     Min', 'MFloat', iptr);
24052:     Max', 'MFloat', iptr);
24053:     Sum', 'MFloat', iptr);
24054:     SumOfSquares', 'MFloat', iptr);
24055:     SumOfCubes', 'MFloat', iptr);
24056:     SumOfQuads', 'MFloat', iptr);
24057:     Func Range : MFloat');
24058:     Func Mean : MFloat');
24059:     Func PopulationVariance : MFloat');
24060:     Func PopulationStdDev : MFloat');
24061:     Func Variance : MFloat');
24062:     Func StdDev : MFloat');
24063:     Func M1 : MFloat');
24064:     Func M2 : MFloat');
24065:     Func M3 : MFloat');
24066:     Func M4 : MFloat');
24067:     Func Skew : MFloat');
24068:     Func Kurtosis : MFloat');
24069:     Func GetAsString :Str');
24070:   end;
24071: end;
24072:
24073: (*-----*)
24074: Proc SIRegister_flcStatistics(CL: TPSPascalCompiler);
24075: begin
24076:   CL.AddClassN(CL.FindClass('TOBJECT'),'EStatistics');
24077:   CL.FindClass('TOBJECT'),'EStatisticsInvalidArgument');
24078:   CL.FindClass('TOBJECT'),'EStatisticsOverflow');
24079:   //CL.AddTypeS('MFloat', 'double');
24080:   Proc SIRegister_flcStatistics(CL: TPSPascalCompiler); begin
24081:     TOBJECT'),'EStatistics');
24082:     TOBJECT'),'EStatisticsInvalidArgument');
24083:     TOBJECT'),'EStatisticsOverflow');
24084:     //CL.AddTypeS('MFloat', 'double'); //check
24085:     CL.AddTypeS('MFloat', 'extended'); //checked
24086:     Func flcBinomialCoeff( N, R : Integer) : MFloat');
24087:     Func flcerf( x : MFloat) : MFloat');
24088:     Func flcerfc( const x : MFloat) : MFloat');
24089:     Func flcCummNormal( const u, s, X : MFloat) : MFloat');
24090:     Func flcCummNormal01( const X : MFloat) : MFloat');
24091:     Func flcInvCummNormal01( Y0 : MFloat) : MFloat');
24092:     Func flcInvCummNormal( const u, s, Y0 : MFloat) : MFloat');
24093:     Func flcCummChiSquare( const Chi, Df : MFloat) : MFloat');
24094:     Func flcCumF( const f, Df1, Df2 : MFloat) : MFloat');
24095:     Func flcCummPoisson( const X : Integer; const u : MFloat) : MFloat');
24096:     SIRegister_TStatisticClass(CL);
24097:     CL.AddClassN(CL.FindClass('TOBJECT'),'EStatistic');
24098:     CL.FindClass('TOBJECT'),'EStatisticNoSample');
24099:     CL.FindClass('TOBJECT'),'EStatisticDivisionByZero');
24100:     Proc TestStatisticClass';
24101:   end;
24102:
24103: Proc Include(var aset: Charset; achar: Char); ---> cFundamentUtils
24104: Proc Exclude(var aset: Charset; achar: Char);
24105: Proc SIRegister_StdFuncs(CL: TPSPascalCompiler);
24106: begin
24107:   CL.AddClassN(CL.FindClass('TOBJECT'),'EParserError');
24108:   CL.AddTypeS('TCharSet', 'set of Char');
24109:   Proc SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
24110:   CL.AddTypeS('CharSet', 'set of Char'); ////
24111:   CL.AddTypeS('AnsiCharSet', 'TCharSet');
24112:   CL.AddTypeS('ByteSet', 'set of Byte');
24113:   CL.AddTypeS('AnsiChar', 'Char');
24114:
24115: Proc SIRegister_flcMaths(CL: TPSPascalCompiler);
24116: begin
24117:   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);

```



```

24118: 'flcPi2','Extended').setExtended( 6.283185307179586476925286766559006);
24119: 'flcPi3','Extended').setExtended( 9.424777960769379715387930149838509);
24120: 'flcPi4','Extended').setExtended( 12.56637061435917295385057353311801);
24121: 'flcPiOn2','Extended').setExtended( 1.570796326794896619231321691639751);
24122: 'flcPiOn3','Extended').setExtended( 1.047197551196597746154214461093168);
24123: 'flcPiOn4','Extended').setExtended( 0.785398163397448309615660845819876);
24124: 'flcPiSq','Extended').setExtended( 9.869604401089358618834490999876151);
24125: 'flcPiE','Extended').setExtended( 22.45915771836104547342715220454374);
24126: 'flcLnPi','Extended').setExtended( 1.144729885849400174143427351353059);
24127: 'flcLogPi','Extended').setExtended( 0.497149872694133854351268288290899);
24128: 'flcSqrtPi','Extended').setExtended( 1.772453850905516027298167483341145);
24129: 'flcSqrt2Pi','Extended').setExtended( 2.506628274631000502415765284811045);
24130: 'flcLnSqrt2Pi','Extended').setExtended( 0.918938533204672741780329736405618);
24131: 'flcDegPerRad','Extended').setExtended( 57.29577951308232087679815481410517);
24132: 'flcDegPerGrad','Extended').setExtended( 0.9);
24133: 'flcDegPerCycle','Extended').setExtended( 360.0);
24134: 'flcGradPerCycle','Extended').setExtended( 400.0);
24135: 'flcGradPerDeg','Extended').setExtended( 1.1111111111111111111111111111111);
24136: 'flcGradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
24137: 'flcRadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
24138: 'flcRadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
24139: 'flcRadPerCycle','Extended').setExtended( 6.283185307179586476925286766559);
24140: 'flcCyclePerDeg','Extended').setExtended( 0.00277777777777777777777777777778);
24141: 'flcCyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
24142: 'flcCyclePerGrad','Extended').setExtended( 0.0025);
24143: 'flcE','Extended').setExtended( 2.718281828459045235360287471352663);
24144: 'flcE2','Extended').setExtended( 7.389056098930650227230427460575008);
24145: 'flcExpM2','Extended').setExtended( 0.135335283236612691893999494972484);
24146: 'flcLn2','Extended').setExtended( 0.693147180559945309417232121458177);
24147: 'flcLn10','Extended').setExtended( 2.302585092994045684017991454684364);
24148: 'flcLogE','Extended').setExtended( 0.434294481903251827651128918916605);
24149: 'flcLog2','Extended').setExtended( 0.301029995663981195213738894724493);
24150: 'flcLog3','Extended').setExtended( 0.477121254719662437295027903255115);
24151: 'flcSqrt2','Extended').setExtended( 1.414213562373095048801688724209698);
24152: 'flcSqrt3','Extended').setExtended( 1.732050807568877293527446341505872);
24153: 'flcSqrt5','Extended').setExtended( 2.236067977499789696409173668731276);
24154: 'flcSqrt7','Extended').setExtended( 2.645751311064590590501615753639260);
24155: //CL.AddTypeS('MFloat', 'Double');
24156: //CL.AddTypeS('MFloatArray', 'DoubleArray');
24157: //CL.AddTypeS('MFloat', 'Extended'); // check t<pe
24158: CL.AddTypeS('ExtendedArray', 'array of MFloat');
24159: //ExtendedArray = array of Extended;
24160: CL.AddTypeS('MFloatArray', 'ExtendedArray');
24161: CL.AddTypeS('Int64Array', 'array of Int64');
24162: flcfx', 'Func (const x: Float): Float;
24163: MFloatArray2', 'array of Mfloat');
24164: //Int64Array = array of Int64;
24165: // CL.AddTypeS('PMFloat', 'MFloat // will not work');
24166: Proc SetFPUPrecisionSingle);
24167: Proc SetFPUPrecisionDouble);
24168: Proc SetFPUPrecisionExtended);
24169: Proc SetFPURoundingNearest);
24170: Proc SetFPURoundingDown);
24171: Proc SetFPURoundingUp);
24172: Proc SetFPURoundingTruncate);
24173: Func flcPolyEval(const X:MFloat;const Coef:array of MFloat;const N:Integer): MFloat;
24174: Func flcSign(const R : Integer) : Integer;
24175: Func flcSign1(const R : Int64) : Integer;
24176: Func flcSign2(const R : Single) : Integer;
24177: Func flcSign3(const R : Double) : Integer;
24178: Func flcSign4(const R : Extended) : Integer;
24179: Func flcFloatMod(const A, B : MFloat) : MFloat);
24180: Func flcATan360(const X, Y : MFloat) : MFloat);
24181: Func flcInverseTangentDeg(const X, Y : MFloat) : MFloat);
24182: Func flcInverseTangentRad(const X, Y : MFloat) : MFloat);
24183: Func flcInverseSinDeg(const Y, R : MFloat) : MFloat);
24184: Func flcInverseSinRad(const Y, R : MFloat) : MFloat);
24185: Func flcInverseCosDeg(const X, R : MFloat) : MFloat);
24186: Func flcInverseCosRad(const X, R : MFloat) : MFloat);
24187: Func flcDMSToReal(const Degr, Mins, Secs : MFloat) : MFloat);
24188: Proc flcRealToDMS(const X : MFloat; var Degr, Mins, Secs : MFloat);
24189: Proc flcPolarToRectangular(const R, Theta : MFloat; var X, Y : MFloat);
24190: Proc flcRectangularToPolar(const X, Y : MFloat; var R, Theta : MFloat);
24191: Func flcDistance(const X1, Y1, X2, Y2 : MFloat) : MFloat);
24192: Func flcIsPrime(const N : Int64) : Bool);
24193: Func flcIsPrimeFactor(const N, F : Int64) : Bool);
24194: Func flcPrimeFactors(const N : Int64) : Int64Array);
24195: Func flcGCD(const N1, N2 : Integer) : Integer;
24196: Func flcGCD1(const N1, N2 : Int64) : Int64;
24197: Func flcIsRelativePrime(const X, Y : Int64) : Bool);
24198: Func flcInvMod(const A, N : Integer) : Integer;
24199: Func flcInvMod1(const A, N : Int64) : Int64;
24200: Func flcExpMod(A, Z : Integer; const N : Integer) : Integer;
24201: Func flcExpMod1(A, Z : Int64; const N : Int64) : Int64;
24202: Func flcJacobi(const A, N : Integer) : Integer);
24203: CL.AddConstantN('FactorialMaxN','LongInt').SetInt( 1754);
24204: //CL.AddConstantN('FactorialMaxN','LongInt').SetInt( 170);
24205: Func flcFactorial(const N : Integer) : MFloat);
24206: Func flcCombinations(const N, C : Integer) : MFloat);

```

```

24207: Func flcPermutations( const N, P : Integer) : MFloat');
24208: Func flcFibonacci( const N : Integer) : Int64');
24209: Func flcGammaLn( X : Extended) : Extended');
24210: Proc flcFourierTransform(const AngleNumerator:MFloat;const RealIn,ImagIn:array of MFloat;var RealOut,
    ImagOut: MFloatArray);
24211: Proc flcFFT( const RealIn,ImagIn:array of MFloat; var RealOut, ImagOut : MFloatArray);
24212: Proc flcInverseFFT(const RealIn,ImagIn:array of MFloat;var RealOut,ImagOut:MFloatArray);
24213: Proc flcCalcFrequency(const FrequencyIndex:Int;const RealIn,ImagIn:array of MFloat;var RealOut,
    ImagOut:MFloat);
24214: Func flcSecantSolver( const f : flcfx; const y, Guess1, Guess2 : MFloat) : MFloat');
24215: Func flcNewtonSolver( const f, df : flcfx; const y, Guess : MFloat) : MFloat');
24216: Func flcFirstDerivative( const f : flcfx; const x : MFloat) : MFloat');
24217: Func flcSecondDerivative( const f : flcfx; const x : MFloat) : MFloat');
24218: Func flcThirdDerivative( const f : flcfx; const x : MFloat) : MFloat');
24219: Func flcFourthDerivative( const f : flcfx; const x : MFloat) : MFloat');
24220: Func flcSimpsonIntegration( const f : flcfx; const a, b : MFloat; N : Integer): MFloat');
24221: Proc TestMathClass');
24222: end;
24223:
24224: Proc SIRegister_flcCharSet(CL: TPSPascalCompiler);
24225: begin
24226: //type ByteSet = set of Byte;
24227: //ByteCharSet = CharSet; /
24228: CL.AddTypeS('ByteCharSet', 'CharSet');
24229: // CL.AddTypeS('ByteCharSet', 'charset');
24230: Func flcAsAnsiCharSet( const C : array of Char) : ByteCharSet');
24231: Func flcAsByteSet( const C : array of Byte) : ByteSet');
24232: Func flcAsByteCharSet( const C : array of Char) : ByteCharSet');
24233: Proc flcComplementChar( var C : ByteCharSet; const Ch : Char);
24234: Proc flcClearCharSet( var C : ByteCharSet);
24235: Proc flcFillCharSet( var C : ByteCharSet);
24236: Proc flcComplementCharSet( var C : ByteCharSet);
24237: Proc flcAssignCharSet( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24238: Proc flcUnion( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24239: Proc flcDifference( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24240: Proc flcIntersection( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24241: Proc flcXORCharSet( var DestSet : ByteCharSet; const SourceSet : ByteCharSet);
24242: Func flcIsSubSet( const A, B : ByteCharSet) :Bool');
24243: Func flcIsEqual( const A, B : ByteCharSet) :Bool');
24244: Func flcIsEmpty( const C : ByteCharSet) :Bool');
24245: Func flcIsComplete( const C : ByteCharSet) :Bool');
24246: Func flcCharCount( const C : ByteCharSet) : Integer');
24247: Proc flcConvertCaseInsensitive( var C : ByteCharSet);
24248: Func flcCaseInsensitiveCharSet( const C : ByteCharSet) : ByteCharSet');
24249: Func flcCharSetToStrB( const C : ByteCharSet) : RawByteString');
24250: Func flcStrToCharSetB( const S :Str) : ByteCharSet');
24251: Func flcCharSetToCharClassStr( const C : ByteCharSet) : Ansistr');
24252: Proc TestCharSet');
24253: end;
24254:
24255: Proc SIRegister_flcTimers(CL: TPSPascalCompiler);
24256: begin
24257: TOBJECT('ETimerError');
24258: Func GetHighResolutionTick : Word64');
24259: Func GetHighResolutionFrequency : Word64');
24260: Func HighResolutionTickDelta( const T1, T2 : Word64) : Int64');
24261: Func HighResolutionTickDeltaU( const T1, T2 : Word64) : Word64');
24262: 'MicroTickFrequency','LongInt').SetInt( 1000000);
24263: Func GetMicroTick : Word64');
24264: Func MicroTickDelta( const T1, T2 : Word64) : Int64');
24265: Func MicroTickDeltaU( const T1, T2 : Word64) : Word64');
24266: 'MilliTickFrequency','LongInt').SetInt( 1000);
24267: Func GetMilliTick : Word64');
24268: Func MilliTickDelta( const T1, T2 : Word64) : Int64');
24269: Func MilliTickDeltaU( const T1, T2 : Word64) : Word64');
24270: Func DateTimeToMicroDateTime( const DT : TDateTime) : Word64');
24271: Func MicroDateTimeToDateTime( const DT : Word64) : TDateTime');
24272: Func GetMicroDateTimeNow : Word64');
24273: Func GetNowUT : TDateTime');
24274: Func GetMicroDateTimeNowUT : Word64');
24275: Func GetNowUTC( const ReInit :Bool) : TDateTime');
24276: Func GetMicroDateTimeNowUTC( const ReInit :Bool) : Word64');
24277: Proc TestTimerClass;
24278: end;
24279:
24280: Proc SIRegister_TBlaiseLexer(CL: TPSPascalCompiler);
24281: begin
24282: //with RegClassS(CL,'TOBJECT', 'TBlaiseLexer') do
24283: with CL.AddClassN(CL.FindClass('TOBJECT'),'TBlaiseLexer') do begin
24284: TokenCount', 'Integer', iptrw);
24285: IdenCount', 'Integer', iptrw);
24286: SymbolCount', 'Integer', iptrw);
24287: IgnoreCount', 'Integer', iptrw);
24288: ReservedCount', 'Integer', iptrw);
24289: Constructor Create');
24290: Proc Reset');
24291: Proc SetData( const Data : pchar; const Size : Integer);
24292: Proc SetText( const S :Str; const StartPosition : Integer);
24293: Func EOF :Bool');

```

```

24294: Proc GetToken');
24295: TokenLen', 'Integer', iptr);
24296: TokenType', 'Integer', iptr);
24297: Func TokenText :Str);
24298: TokenPos', 'Integer', iptr);
24299: LastTokenEnd', 'Integer', iptr);
24300: Position', 'Integer', iptrw);
24301: LineNr', 'Integer', iptr);
24302: Column', 'Integer', iptr);
24303: CurrentLine', 'String', iptr);
24304: Func MatchType( const TokenType : Integer; const Skip :Bool) :Bool);
24305: Func MatchTokenText(const Token:str;const Skip:Bool;const CaseSensitive:Bool):Bool;
24306: Func TokenAsInteger : Int64);
24307: end;
24308: end;
24309:
24310: Proc SIRegister_TRationalClass(CL: TPSPascalCompiler);
24311: begin
24312: //with RegClassS(CL,'TOBJECT', 'TRationalClass') do
24313: with CL.AddClassN(CL.FindClass('TOBJECT'),'TRationalClass') do begin
24314:   Constructor Create;
24315:   Constructor Create1( const Numerator : Int64; const Denominator: Int64);
24316:   Constructor Create2( const R : Extended);
24317:   Numerator', 'Int64', iptrw);
24318:   Denominator', 'Int64', iptrw);
24319:   AsString', 'String', iptrw);
24320:   AsStringB', 'RawByteString', iptrw);
24321:   AsStringU', 'UnicodeString', iptrw);
24322:   AsFloat', 'MFloat', iptrw);
24323:   Func Duplicate : TRationalClass);
24324:   Proc Assign( const R : TRationalClass);
24325:   Proc Assign1( const R : MFloat);
24326:   Proc Assign2( const Numerator : Int64; const Denominator : Int64);
24327:   Proc AssignZero');
24328:   Proc AssignOne');
24329:   Func IsEqual( const R : TRationalClass) :Bool;
24330:   Func IsEqual1(const Numerator:Int64;const Denominator:Int64):Bool;
24331:   Func IsEqual2( const R : Extended) :Bool;
24332:   Func IsZero :Bool);
24333:   Func IsOne :Bool);
24334:   Proc Add( const R : TRationalClass);
24335:   Proc Add1( const V : Extended);
24336:   Proc Add2( const V : Int64);
24337:   Proc Subtract( const R : TRationalClass);
24338:   Proc Subtract1( const V : Extended);
24339:   Proc Subtract2( const V : Int64);
24340:   Proc Negate');
24341:   Proc Abs');
24342:   Func Sgn : Integer');
24343:   Proc Multiply( const R : TRationalClass);
24344:   Proc Multiply1( const V : Extended);
24345:   Proc Multiply2( const V : Int64);
24346:   Proc Divide( const R : TRationalClass);
24347:   Proc Divide1( const V : Extended);
24348:   Proc Divide2( const V : Int64);
24349:   Proc Reciprocal');
24350:   Proc Sqrt');
24351:   Proc Sqr');
24352:   Proc Power( const R : TRationalClass);
24353:   Proc Power1( const V : Int64);
24354:   Proc Power2( const V : Extended);
24355:   Proc Exp');
24356:   Proc Ln');
24357:   Proc Sin');
24358:   Proc Cos');
24359: end;
24360: end;
24361:
24362: Proc SIRegister_TComplexClass(CL: TPSPascalCompiler);
24363: begin
24364: //with RegClassS(CL,'TOBJECT', 'TComplexClass') do
24365: with CL.AddClassN(CL.FindClass('TOBJECT'),'TComplexClass') do begin
24366:   Constructor Create( const ARealPart : MFloat; const AImaginaryPart : MFloat);
24367:   RealPart', 'MFloat', iptrw);
24368:   ImaginaryPart', 'MFloat', iptrw);
24369:   AsString', 'String', iptrw);
24370:   AsStringB', 'RawByteString', iptrw);
24371:   AsStringU', 'UnicodeString', iptrw);
24372:   Proc Assign( const C : TComplex);
24373:   Proc Assign1( const V : MFloat);
24374:   Proc AssignZero');
24375:   Proc AssignI');
24376:   Proc AssignMinI');
24377:   Func Duplicate : TComplexClass);
24378:   Func IsEqual2( const C : TComplexClass) :Bool;
24379:   Func IsEqual3( const R, I : MFloat) :Bool;
24380:   Func IsReal :Bool);
24381:   Func IsZero :Bool);
24382:   Func IsI :Bool);

```

```

24383: Proc Add4( const C : TComplex);
24384: Proc Add5( const V : MFloat);
24385: Proc Subtract6( const C : TComplexClass);
24386: Proc Subtract7( const V : MFloat);
24387: Proc Multiply8( const C : TComplexClass);
24388: Proc Multiply9( const V : MFloat);
24389: Proc MultiplyI';
24390: Proc MultiplyMinI';
24391: Proc Divide10( const C : TComplexClass);
24392: Proc Divide11( const V : MFloat);
24393: Proc Negate';
24394: Func Modulo : MFloat';
24395: Func Denom : MFloat';
24396: Proc Conjugate';
24397: Proc Inverse';
24398: Proc Sqrt';
24399: Proc Exp';
24400: Proc Ln';
24401: Proc Sin';
24402: Proc Cos';
24403: Proc Tan';
24404: Proc Power( const C : TComplexClass);
24405: end;
24406: end;
24407:
24408: Proc SIRegister_TMatrixClass(CL: TPSPascalCompiler);
24409: begin
24410: //with RegClassS(CL,'TOBJECT', 'TMatrixClass') do
24411: with CL.AddClassN(CL.FindClass('TOBJECT'),'TMatrixClass') do begin
24412: Constructor CreateSize( const RowCount, ColCount : Integer);
24413: Constructor CreateSquare( const N : Integer; const Identity :Bool);
24414: //RegisterMethod('Constructor CreateDiagonal( const D : TVector);
24415: ColCount', 'Integer', iptrw);
24416: RowCount', 'Integer', iptrw);
24417: Proc SetSize( const RowCount, ColCount : Integer);
24418: Proc Clear';
24419: Item', 'MFloat Integer Integer', iptrw);
24420: SetDefaultProperty('Item');
24421: Proc AssignZero';
24422: Proc AssignIdentity';
24423: Proc Assign( const Value : MFloat);
24424: Proc Assign1( const M : TMatrix);
24425: //RegisterMethod('Proc Assign2( const V : TVector);
24426: Func Duplicate : TMatrixClass;
24427: Func DuplicateRange( const R1, C1, R2, C2 : Integer) : TMatrixClass;
24428: //RegisterMethod('Func DuplicateRow( const Row : Integer) : TVector');
24429: //RegisterMethod('Func DuplicateCol( const Col : Integer) : TVector');
24430: //RegisterMethod('Func DuplicateDiagonal : TVector');
24431: Func IsEqual( const M : TMatrixClass) :Bool;
24432: //RegisterMethod('Func IsEqual6( const V : TVector) :Bool;
24433: Func IsSquare :Bool';
24434: Func IsZero :Bool';
24435: Func IsIdentity :Bool';
24436: AsString', 'String', iptr);
24437: AsStringB', 'RawByteString', iptr);
24438: AsStringU', 'UnicodeString', iptr);
24439: Func Trace : MFloat';
24440: //RegisterMethod('Proc SetRow7(const Row: Integer; const V : TVector);
24441: Proc SetRow( const Row : Integer; const Values : array of MFloat);
24442: //RegisterMethod('Proc SetCol( const Col: Integer; const V : TVector);
24443: Func Transpose : TMatrixClass);
24444: Proc TransposeInPlace';
24445: Proc Add( const M : TMatrixClass);
24446: Proc Subtract( const M : TMatrixClass);
24447: Proc Negate';
24448: Proc MultiplyRow( const Row : Integer; const Value : MFloat);
24449: Proc Multiply( const Value : MFloat);
24450: Func Multiply1( const M : TMatrixClass) : TMatrixClass;
24451: Proc MultiplyInPlace( const M : TMatrixClass);
24452: Func IsOrthogonal :Bool';
24453: Func IsIdempotent :Bool';
24454: Func Normalise( const M : TMatrixClass) : MFloat';
24455: Func SolveMatrix( var M : TMatrixClass) : MFloat';
24456: Func Determinant : MFloat';
24457: Func Inverse : TMatrixClass';
24458: Proc InverseInPlace';
24459: //RegisterMethod('Func SolveLinearSystem( const V : TVector) : TVector');
24460: end;
24461: end;
24462:
24463: Proc SIRegister_TStringBuilder(CL: TPSPascalCompiler);
24464: begin
24465: //with RegClassS(CL,'TOBJECT', 'TStringBuilder') do
24466: with CL.AddClassN(CL.FindClass('TOBJECT'),'TStringBuilder') do begin
24467: Constructor Create( const S :Str);
24468: Constructor Create2( const Capacity : Integer);
24469: RegisterProperty('Length', 'Integer', iptr);
24470: RegisterProperty('AsString', 'String', iptrw);
24471: Proc Clear';

```

```

24472: Proc Assign( const S : TStringBuilder);
24473: Proc Append( const S :Str);
24474: Proc AppendCRLF');
24475: Proc AppendLn( const S :Str);
24476: Proc Append1( const S :Str; const Count : Integer);
24477: Proc AppendCh( const C : Char);
24478: Proc AppendCh1( const C : Char; const Count : Integer);
24479: Proc Append2( const S : TStringBuilder);
24480: Proc Pack');
24481: end;
24482: Proc TestStringBuilderClass
24483: end;
24484:
24485: * === compile-time registration functions === *)
24486: (*-----*)
24487: Proc SRegister_flgASCII(CL: TPSPascalCompiler);
24488: begin
24489: 'AsciiNULL','Char').SetString( ByteChar ( #0 ));
24490: 'AsciiSOH','Char').SetString( ByteChar ( #1 ));
24491: 'AsciiSTX','Char').SetString( ByteChar ( #2 ));
24492: 'AsciiETX','Char').SetString( ByteChar ( #3 ));
24493: 'AsciiEOT','Char').SetString( ByteChar ( #4 ));
24494: 'AsciiENQ','Char').SetString( ByteChar ( #5 ));
24495: 'AsciiACK','Char').SetString( ByteChar ( #6 ));
24496: 'AsciiBEL','Char').SetString( ByteChar ( #7 ));
24497: 'AsciiBS','Char').SetString( ByteChar ( #8 ));
24498: 'AsciiHT','Char').SetString( ByteChar ( #9 ));
24499: 'AsciiLF','Char').SetString( ByteChar ( #10 ));
24500: 'AsciiVT','Char').SetString( ByteChar ( #11 ));
24501: 'AsciiFF','Char').SetString( ByteChar ( #12 ));
24502: 'AsciiCR','Char').SetString( ByteChar ( #13 ));
24503: 'AsciiSO','Char').SetString( ByteChar ( #14 ));
24504: 'AsciiSI','Char').SetString( ByteChar ( #15 ));
24505: 'AsciiDLE','Char').SetString( ByteChar ( #16 ));
24506: 'AsciiDC1','Char').SetString( ByteChar ( #17 ));
24507: 'AsciiDC2','Char').SetString( ByteChar ( #18 ));
24508: 'AsciiDC3','Char').SetString( ByteChar ( #19 ));
24509: 'AsciiDC4','Char').SetString( ByteChar ( #20 ));
24510: 'AsciiNAK','Char').SetString( ByteChar ( #21 ));
24511: 'AsciiSYN','Char').SetString( ByteChar ( #22 ));
24512: 'AsciiETB','Char').SetString( ByteChar ( #23 ));
24513: 'AsciiCAN','Char').SetString( ByteChar ( #24 ));
24514: 'AsciiEM','Char').SetString( ByteChar ( #25 ));
24515: 'AsciiEOF','Char').SetString( ByteChar ( #26 ));
24516: 'AsciiESC','Char').SetString( ByteChar ( #27 ));
24517: 'AsciiFS','Char').SetString( ByteChar ( #28 ));
24518: 'AsciiGS','Char').SetString( ByteChar ( #29 ));
24519: 'AsciiRS','Char').SetString( ByteChar ( #30 ));
24520: 'AsciiUS','Char').SetString( ByteChar ( #31 ));
24521: 'AsciiSP','Char').SetString( ByteChar ( #32 ));
24522: 'AsciiDEL','Char').SetString( ByteChar ( #127 ));
24523: 'AsciiXON','char').SetString( AsciiDC1);
24524: 'AsciiXOFF','char').SetString( AsciiDC3);
24525: 'AsciiDecimalPoint','Char').SetString( ByteChar ( #46 ));
24526: 'AsciiComma','Char').SetString( ByteChar ( #44 ));
24527: 'AsciiBackSlash','Char').SetString( ByteChar ( #92 ));
24528: 'AsciiForwardSlash','Char').SetString( ByteChar ( #47 ));
24529: 'AsciiPercent','Char').SetString( ByteChar ( #37 ));
24530: 'AsciiAmpersand','Char').SetString( ByteChar ( #38 ));
24531: 'AsciiPlus','Char').SetString( ByteChar ( #43 ));
24532: 'AsciiMinus','Char').SetString( ByteChar ( #45 ));
24533: 'AsciiEqualSign','Char').SetString( ByteChar ( #61 ));
24534: 'AsciiSingleQuote','Char').SetString( ByteChar ( #39 ));
24535: 'AsciiDoubleQuote','Char').SetString( ByteChar ( #34 ));
24536: 'AsciiDigit0','Char').SetString( ByteChar ( #48 ));
24537: 'AsciiDigit9','Char').SetString( ByteChar ( #57 ));
24538: 'AsciiUpperA','Char').SetString( ByteChar ( #65 ));
24539: 'AsciiUpperZ','Char').SetString( ByteChar ( #90 ));
24540: 'AsciiLowerA','Char').SetString( ByteChar ( #97 ));
24541: 'AsciiLowerZ','Char').SetString( ByteChar ( #122 ));
24542: 'WideNULL','Char').SetString( WideChar ( #0 ));
24543: 'WideSOH','Char').SetString( WideChar ( #1 ));
24544: 'WideSTX','Char').SetString( WideChar ( #2 ));
24545: 'WideETX','Char').SetString( WideChar ( #3 ));
24546: 'WideEOT','Char').SetString( WideChar ( #4 ));
24547: 'WideENQ','Char').SetString( WideChar ( #5 ));
24548: 'WideACK','Char').SetString( WideChar ( #6 ));
24549: 'WideBEL','Char').SetString( WideChar ( #7 ));
24550: 'WideBS','Char').SetString( WideChar ( #8 ));
24551: 'WideHT','Char').SetString( WideChar ( #9 ));
24552: 'WideLF','Char').SetString( WideChar ( #10 ));
24553: 'WideVT','Char').SetString( WideChar ( #11 ));
24554: 'WideFF','Char').SetString( WideChar ( #12 ));
24555: 'WideCR','Char').SetString( WideChar ( #13 ));
24556: 'WideSO','Char').SetString( WideChar ( #14 ));
24557: 'WideSI','Char').SetString( WideChar ( #15 ));
24558: 'WideDLE','Char').SetString( WideChar ( #16 ));
24559: 'WideDC1','Char').SetString( WideChar ( #17 ));
24560: 'WideDC2','Char').SetString( WideChar ( #18 ));

```



```

24561: 'WideDC3', 'Char').SetString( WideChar ( #19 ));
24562: 'WideDC4', 'Char').SetString( WideChar ( #20 ));
24563: 'WideNAK', 'Char').SetString( WideChar ( #21 ));
24564: 'WideSYN', 'Char').SetString( WideChar ( #22 ));
24565: 'WideETB', 'Char').SetString( WideChar ( #23 ));
24566: 'WideCAN', 'Char').SetString( WideChar ( #24 ));
24567: 'WideEM', 'Char').SetString( WideChar ( #25 ));
24568: 'WideEOF', 'Char').SetString( WideChar ( #26 ));
24569: 'WideESC', 'Char').SetString( WideChar ( #27 ));
24570: 'WideFS', 'Char').SetString( WideChar ( #28 ));
24571: 'WideGS', 'Char').SetString( WideChar ( #29 ));
24572: 'WideRS', 'Char').SetString( WideChar ( #30 ));
24573: 'WideUS', 'Char').SetString( WideChar ( #31 ));
24574: 'WideSP', 'Char').SetString( WideChar ( #32 ));
24575: 'WideDEL', 'Char').SetString( WideChar ( #127 ));
24576: 'WideXON', 'char').SetString( WideDC1);
24577: 'WideXOFF', 'char').SetString( WideDC3);
24578: 'WideCRLF', 'String').SetString( WideString ( #13#10 ));
24579: Func IsAsciiCharB( const C : Char ) : Bool';
24580: Func IsAsciiCharW( const C : WideChar ) : Bool';
24581: Func IsAsciiChar( const C : Char ) : Bool';
24582: //Func IsAsciiBufB( const Buf: Pointer; const Len: NativeInt):Bool';
24583: //Func IsAsciiBufW( const Buf: Pointer; const Len: NativeInt):Bool';
24584: Func IsAsciiStringB( const S : Str ) : Bool';
24585: Func IsAsciiStringU( const S : UnicodeString ) : Bool';
24586: Func IsAsciiString( const S : Str ) : Bool';
24587: Func AsciiHexCharValueB( const C : Char ) : Integer';
24588: Func AsciiHexCharValueW( const C : WideChar ) : Integer';
24589: Func AsciiIsHexCharB( const C : Char ) : Bool';
24590: Func AsciiIsHexCharW( const C : WideChar ) : Bool';
24591: Func AsciiDecimalCharValueB( const C : Char ) : Integer';
24592: Func AsciiDecimalCharValueW( const C : WideChar ) : Integer';
24593: Func AsciiIsDecimalCharB( const C : Char ) : Bool';
24594: Func AsciiIsDecimalCharW( const C : WideChar ) : Bool';
24595: Func AsciiOctalCharValueB( const C : Char ) : Integer';
24596: Func AsciiOctalCharValueW( const C : WideChar ) : Integer';
24597: Func AsciiIsOctalCharB( const C : Char ) : Bool';
24598: Func AsciiIsOctalCharW( const C : WideChar ) : Bool';
24599: Func CharCompareNoAsciiCaseB( const A, B : Char ) : Integer';
24600: Func CharCompareNoAsciiCaseW( const A, B : WideChar ) : Integer';
24601: Func CharCompareNoAsciiCase( const A, B : Char ) : Integer';
24602: Func CharEqualNoAsciiCaseB( const A, B : Char ) : Bool';
24603: Func CharEqualNoAsciiCaseW( const A, B : WideChar ) : Bool';
24604: Func CharEqualNoAsciiCase( const A, B : Char ) : Bool';
24605: //Func StrPCompareNoAsciiCaseB( const A, B: PByteChar; const Len: NativeInt):Integer';
24606: //Func StrPCompareNoAsciiCaseW( const A, B: PWideChar; const Len: NativeInt):Integer';
24607: Func StrPCompareNoAsciiCase( const A, B : PChar; const Len : NativeInt ) : Integer';
24608: Func StrCompareNoAsciiCaseA( const A, B : Ansistr ) : Integer';
24609: Func StrCompareNoAsciiCaseB( const A, B : Str ) : Integer';
24610: Func StrCompareNoAsciiCaseU( const A, B : UnicodeString ) : Integer';
24611: Func StrCompareNoAsciiCase( const A, B : Str ) : Integer';
24612: Func AsciiLowerCaseB( const C : Char ) : Char';
24613: Func AsciiLowerCaseW( const C : WideChar ) : WideChar';
24614: Func AsciiLowerCase( const C : Char ) : Char';
24615: Func AsciiUpperCaseB( const C : Char ) : Char';
24616: Func AsciiUpperCaseW( const C : WideChar ) : WideChar';
24617: Func AsciiUpperCase( const C : Char ) : Char';
24618: Proc AsciiConvertUpperB( var S : Str);
24619: Proc AsciiConvertUpperU( var S : UnicodeString);
24620: Proc AsciiConvertUpper( var S : Str);
24621: Proc AsciiConvertLowerB( var S : Str);
24622: Proc AsciiConvertLowerU( var S : UnicodeString);
24623: Proc AsciiConvertLower( var S : Str);
24624: Func AsciiUpperCaseB( const A : Str ) : Str';
24625: Func AsciiUpperCaseU( const A : UnicodeString ) : UnicodeString';
24626: Func AsciiUpperCase( const A : Str ) : Str';
24627: Func AsciiLowerCaseB( const A : Str ) : Str';
24628: Func AsciiLowerCaseU( const A : UnicodeString ) : UnicodeString';
24629: Func AsciiLowerCase( const A : Str ) : Str';
24630: Proc AsciiConvertFirstUpB( var S : RawByteString);
24631: Proc AsciiConvertFirstUpU( var S : UnicodeString);
24632: Proc AsciiConvertFirstUp( var S : Str);
24633: Func AsciiFirstUpB( const S : RawByteString ) : RawByteString';
24634: Func AsciiFirstUpU( const S : UnicodeString ) : UnicodeString';
24635: Func AsciiFirstUp( const S : Str ) : Str';
24636: Proc AsciiConvertArrayUpper( var S : TStringArray);
24637: Proc AsciiConvertArrayLower( var S : TStringArray);
24638: Proc TestAscii';
24639: end;
24640:
24641: Proc SIRegister_flgStringPatternMatcher(CL: TPSPascalCompiler);
24642: begin
24643: TMatchPatternGreed', '( mpgLazy, mpgGreedy, mpgGreedyNoBacktrack);
24644: //CL.AddDelphiFunction('Func StrZMatchPatternA(M, S: PAnsiChar; const G : TMatchPatternGreed): Integer');
24645: //CL.AddDelphiFunction('Func StrZMatchPatternW(M, S: PWideChar; const G : TMatchPatternGreed): Integer');
24646: Func StrEqualPatternA(const M,S: RawByteString; const G : TMatchPatternGreed) : Bool;
24647: Func StrEqualPatternU(const M,S: UnicodeString; const G : TMatchPatternGreed) : Bool;
24648: Func StrEqualPattern( const M, S : Str; const G : TMatchPatternGreed ) : Bool';
24649: Func StrPosPatternA(const F,S:RawByteString; var Len:Int;const StartIndex:Int;const
G:TMatchPatternGreed):Int;

```

```

24650: Func StrPosPatternU(const F,S:UnicodeString;var Len:Int;const StartIndex:Int const
G:TMatchPatternGreed):Int;
24651: Func MatchFileMaskB(const Mask,Key:RawByteString;const AsciiCaseSensitive:Bool):Bool;
24652: TMatchQuantifier','( mqOnce, mqAny, mqLeastOnce, mqOptional );
24653: TMatchQuantSeqOptions', '(moDeterministic, moNonGreedy );
24654: //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');
24655: Proc TestPatternmatcher';
24656: Func CharMatch(const A, B: Char; const AsciiCaseSensitive:Bool):Bool;
24657: end;
24658:
24659: https://github.com/fundamentalslib/fundamentals5/blob/master/Source/Utils/flcStrings.pas
24660: https://github.com/fundamentalslib/fundamentals5/blob/master/Source/Utils/flcUtils.pas
24661: https://raw.githubusercontent.com/fundamentalslib/fundamentals5/master/Source/Utils/flcUnicodeChar.pas
24662: 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).
24663: 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.
24664: 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.
24665: 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:
24666:
24667: Func HexToBits(C: Char): Byte;
24668: begin
24669:   case C of
24670:     '0'..'9': Result:= Byte(Ord(C) - Ord('0'));
24671:     'a'..'f': Result:= Byte(10 + (Ord(C) - Ord('a')));
24672:     'A'..'F': Result:= Byte(10 + (Ord(C) - Ord('A')));
24673:   else
24674:     raise Exception.Create('Invalid encoding detected');
24675:   end;
24676: end;
24677:
24678: https://stackoverflow.com/questions/1410771/decode-utf-8-encoded-cyrillic-with-delphi-2007
24679: (* === compile-time registration functions === *)
24680: (*-----*)
24681: Proc SIRegister_flcUnicodeChar(CL: TPSPascalCompiler);
24682: begin
24683:   WideStringQuote','(String').SetString( WideString ( ''' ));
24684:   WideStringDoubleQuote','(String').SetString( WideString ( '" ' ));
24685:   WideStringNoBreakSpace','(Char').SetString( WideString ( #00A0 ));
24686:   WideStringLineSeparator','(Char').SetString( WideString ( #2028 ));
24687:   WideStringParagraphSeparator','(Char').SetString( WideString ( #2029 ));
24688:   WideStringBOM_MSB_First','(Char').SetString( WideString ( #FFFE ));
24689:   WideStringBOM_LSB_First','(Char').SetString( WideString ( #FEFF ));
24690:   WideStringObjectReplacement','(Char).SetString( WideString ( #FFFC ));
24691:   WideStringCharReplacement','(Char).SetString( WideString ( #FFFD ));
24692:   WideStringInvalid','(Char).SetString( WideString ( #FFFF ));
24693:   WideStringCopyrightSign','(Char).SetString( WideString ( #00A9 ));
24694:   WideStringRegisteredSign','(Char).SetString( WideString ( #00AE ));
24695:   WideStringHighSurrogateFirst','(Char).SetString( WideString ( #D800 ));
24696:   WideStringHighSurrogateLast','(Char).SetString( WideString ( #DB7F ));
24697:   WideStringLowSurrogateFirst','(Char).SetString( WideString ( #DC00 ));
24698:   WideStringLowSurrogateLast','(Char).SetString( WideString ( #DFFF ));
24699:   WideStringPrivateHighSurrogateFirst','(Char).SetString( WideString ( #DB80 ));
24700:   WideStringPrivateHighSurrogateLast','(Char).SetString( WideString ( #DBFF ));
24701:   flcWideNULL','(Char).SetString( WideString ( #0 ));
24702:   flcWideBS','(Char).SetString( WideString ( #8 ));
24703:   flcWideHT','(Char).SetString( WideString ( #9 ));
24704:   flcWideLF','(Char').SetString( WideString ( #10 ));
24705:   flcWideVT','(Char').SetString( WideString ( #11 ));
24706:   flcWideFF','(Char').SetString( WideString ( #12 ));
24707:   flcWideCR','(Char').SetString( WideString ( #13 ));
24708:   flcWideEOF','(Char').SetString( WideString ( #26 ));
24709:   flcWideSP','(Char').SetString( WideString ( #32 ));
24710:   'UCS4_STRING_TERMINATOR','(LongWord').SetUInt( #9C);
24711:   'UCS4_LF','(LongWord').SetUInt( #0A);
24712:   'UCS4_CR','(LongWord').SetUInt( #0D);
24713: Func UnicodeIsAsciiChar( const Ch : WideString ) : Bool';
24714: CL.AddTypeS('TUnicodeSpaceWidth','(uswNone,uswZero,uswAdjustableEmOneOver'
+ 'Four,uswEmOneOverTwo,uswEmOneOverThree,uswEmOneOverFour,uswEmFourOverEighteen,uswEmOneOverFive,
uswEmOneOverSix,uswEmOne,uswWidthDigit,uswWidthPeriod,uswWidthIdeographic);
24716: Func UnicodeSpaceWidth( const Ch : WideString ) : TUnicodeSpaceWidth';
24717: Func UnicodeIsNoBreakSpace( const Ch : WideString ) : Bool';
24718: Func UnicodeIsSpaceFixedWidth( const Ch : WideString ) : Bool';
24719: Func UnicodeIsSpaceAdjustableWidth( const Ch : WideString ) : Bool';
24720: Func UnicodeIsSpace( const Ch : WideString ) : Bool';
24721: Func UnicodeIsLineBreak( const Ch : WideString ) : Bool';
24722: Func UnicodeIsWhiteSpace( const Ch : WideString ) : Bool';
24723: Func UnicodeIsControl( const Ch : WideString ) : Bool';
24724: Func UnicodeIsControlOrWhiteSpace( const Ch : WideString ) : Bool';
24725: Func UnicodeIsIgnorable( const Ch : UCS4Char ) : Bool';

```

```

24726: Func UnicodeIsDash( const Ch : WideChar ) :Bool';
24727: Func UnicodeIsHyphen( const Ch : WideChar ) :Bool';
24728: Func UnicodeIsDashOrHyphen( const Ch : WideChar ) :Bool';
24729: Func UnicodeIsFullStop0( const Ch : UCS4Char ) :Bool;
24730: Func UnicodeIsFullStop1( const Ch : WideChar ) :Bool;
24731: Func UnicodeIsComma( const Ch : WideChar ) :Bool';
24732: Func UnicodeIsExclamationMark( const Ch : WideChar ) :Bool';
24733: Func UnicodeIsQuestionMark( const Ch : WideChar ) :Bool';
24734: Func UnicodeIsLeftParenthesis( const Ch : WideChar ) :Bool';
24735: Func UnicodeIsLeftBracket( const Ch : WideChar ) :Bool';
24736: Func UnicodeGetRightParenthesis( const LeftParenthesis : WideChar ) : WideChar';
24737: Func UnicodeGetRightBracket( const LeftBracket : WideChar ) : WideChar';
24738: Func UnicodeIsSingularQuotationMark( const Ch : WideChar ) :Bool';
24739: Func UnicodeIsOpeningQuotationMark( const Ch : WideChar ) :Bool';
24740: Func UnicodeIsClosingQuotationMark( const Ch : WideChar ) :Bool';
24741: Func UnicodeGetClosingQuotationMark( const OpeningQuote : WideChar ) : WideChar';
24742: Func UnicodeGetOpeningQuotationMark( const ClosingQuote : WideChar ) : WideChar';
24743: Func UnicodeIsPunctuation( const Ch : WideChar ) :Bool';
24744: Func UnicodeIsDecimalDigit2( const Ch : UCS4Char ) :Bool;
24745: Func UnicodeIsDecimalDigit3( const Ch : WideChar ) :Bool;
24746: Func UnicodeIsAsciiDecimalDigit( const Ch : WideChar ) :Bool';
24747: Func UnicodeDecimalDigitValue4( const Ch : UCS4Char ) : Integer;
24748: Func UnicodeDecimalDigitValue5( const Ch : WideChar ) : Integer;
24749: Func UnicodeFractionCharacterValue( const Ch : WideChar; var A, B :Integer):Bool';
24750: Func UnicodeRomanNumeralValue( const Ch : WideChar ) : Integer';
24751: Func UnicodeIsAsciiHexDigit( const Ch : WideChar ) :Bool';
24752: Func UnicodeIsHexDigit6( const Ch : UCS4Char ) :Bool;
24753: Func UnicodeIsHexDigit7( const Ch : WideChar ) :Bool;
24754: Func UnicodeHexDigitValue8( const Ch : UCS4Char ) : Integer;
24755: Func UnicodeHexDigitValue9( const Ch : WideChar ) : Integer;
24756: Func UnicodeIsUpperCase( const Ch : WideChar ) :Bool';
24757: Func UnicodeIsLowerCase( const Ch : WideChar ) :Bool';
24758: Func UnicodeIsTitleCase( const Ch : WideChar ) :Bool';
24759: Func UnicodeIsLetter( const Ch : WideChar ) :Bool';
24760: Func UnicodeIsAlphabetic( const Ch : WideChar ) :Bool';
24761: Func UnicodeUpCase( const Ch : WideChar ) : WideChar';
24762: Func UnicodeLowCase( const Ch : WideChar ) : WideChar';
24763: Func UnicodeCharIsEqualNoCase( const A, B : WideChar ) :Bool';
24764: Func UnicodeGetCombiningClass( const Ch : WideChar ) : Byte';
24765: Func UnicodeLocateFoldingUpperCase( const Ch : WideChar ) : UnicodeString';
24766: Func UnicodeLocateFoldingTitleCase( const Ch : WideChar ) : UnicodeString';
24767: Func UnicodeUpCaseFoldingU( const Ch : WideChar ) : UnicodeString';
24768: Func UnicodeLowCaseFoldingU( const Ch : WideChar ) : UnicodeString';
24769: Func UnicodeTitleCaseFoldingU( const Ch : WideChar ) : UnicodeString';
24770: Func UnicodeGetCharacterDecompositionU10( const Ch : WideChar ) : UnicodeString;
24771: Func UnicodeGetCharacterDecompositionU11( const Ch : UCS4Char ) : UnicodeString;
24772: Func UnicodeUpperCaseFoldingU( const S : UnicodeString ) : UnicodeString';
24773: Func UnicodeLowerCaseFoldingU( const S : UnicodeString ) : UnicodeString';
24774: Proc TestUnicodeChar';
24775: Func HexToBits(C: Char): Byte;
24776: Func Utf8ToStr(const Source :Str) :Str; end;
24777: unit PJResFile_Routines;
24778: Func IsIntResource( const ResID : PChar ) :Bool';
24779: Func IsEqualResID2( const R1, R2 : PChar ) :Bool';
24780: Func ResIDToStr( const ResID : PChar ) :Str';
24781:
24782: 15 CLF_Fundamentals Testroutines 47520
24783: -----
24784: 1 TestMathClass;
24785: 2 TestStatisticClass;
24786: 3 TestBitClass;
24787: 4 TestCharset;
24788: 5 TestTimerClass
24789: 6 TestRationalClass
24790: 7 TestComplexClass
24791: 8 TestMatrixClass;
24792: 9 TestStringBuilderClass
24793: 10 TestASCII;
24794: 11 TestASCIIRoutines;
24795: 12 TestPatternmatcher;
24796: 13 TestUnicodeChar;
24797: 14 unit uPSI_AfUtils;
24798: 15 unit uPSI_PsAPI;
24799: 16 TestVectorClass;
24800: 17 TestVectorClassExtended;
24801: 18 TestKMeans;
24802: 19 TestKMeans2;
24803: 20 TestTNNNetVolume;
24804: 21 TestConvolutionAPI();
24805: 22 TestDataParallelism(NN: TNNNet);
24806: //TestTNNNetVolume;
24807:
24808: *****
24809: Release Notes maXbox 4.7.5.20 Jan 2021 mX47
24810: *****
24811: Add 25 Units + 4 Tutorials
24812: 1277 unit uPSI_SystemsDiagram.pas Dendron
24813: 1278 unit uPSI_qsFoundation.pas Dendron
24814: 1279 uPSI_JclStringLists2 JCL

```

```

24815: 1280 uPSI_cInternetUtils2          FLC
24816: 1281 uPSI_cWindows.pas              FLC
24817: 1282 uPSI_flcSysUtils.pas +TBytes    utils
24818: 1283 unit uPSI_RotImg.pas            DA
24819: 1284 uPSI_SimpleImageLoader.pas     LAZ
24820: 1285 uPSI_HSLUtils.pas              LAZ
24821: 1286 uPSI_GraphicsMathLibrary.pas   EF
24822: 1287 unit uPSI_umodels.pas          DMATH
24823: 1288 uPSI_flcStatistics.pas         FLC5
24824: 1289 uPSI_flcMaths.pas              FLC5
24825: 1290 uPSI_flcCharSet.pas
24826: 1291 uPSI_flcBits32.pas
24827: 1292 uPSI_flcTimers.pas
24828: 1293 uPSI_cBlaiseParserLexer.pas
24829: 1294 uPSI_flcRational.pas
24830: 1295 uPSI_flcComplex.pas
24831: 1296 unit uPSI_flcMatrix (uPSI_flcVectors.pas)
24832: 1297 unit uPSI_flcStringBuilder.pas
24833: 1298 unit PResFile_Routines;
24834: 1299 uPSI_flcASCII.pas
24835: 1300 uPSI_flcStringPatternMatcher;
24836: 1301 unit uPSI_flcUnicodeChar.pas
24837: Totals of Func Calls: 33282
24838: SHA1: of 4.7.5.20 D82EAD01C58738887661428F94B207DB1D8FAEB5
24839: CRC32: 203C82F0 29.5 MB (31,012,768 bytes)
24840: newunits: 01 RotImg.pas + uModel : TModel lib dmath.dll
24841: 02 SimpleImageLoader.pas
24842: 03 systemsdiagram.pas + fpc switch
24843: 04 qsFoundation.pas NO FPC Vector operator
24844: 05 prediction.pas SimulationEngine missing
24845: 06 HSLUtils.pas - color model
24846: 07 cInternetUtils.pas - header information
24847: 08 cWindows.pas - cstrings routines as flcSysUtils
24848: 09 flcSysUtils.pas //2 functions with cwindows possible+ freqObj +TBytes utils
24849: 10 GraphicsMathLibrary.PAS gml-prefix
24850: 11 flcBits32.pas //IFDEF DEBUG {IFDEF TEST} Proc Test;
24851: 12 flcFloats.pas No Floats instead: uPSI_cBlaiseParserLexer.pas
24852: 13 flcDecimal.pas + TestClass
24853: 14 flcCharSet.pas //test include wrapper + -
24854: 15 flcComplex.pas -Class
24855: 16 flcMaths.pas //{IFDEF MATHS_TEST} Proc Test
24856: 17 flcMatrix.pas - less TVectors
24857: 18 flcRational.pas -Class
24858: 19 flcStatistics.pas -Class
24859: 20 flcStringBuilder.pas - less Unicode
24860: 21 flcVectors.pas No Vectors cause compatibility
24861: 22 flcTimers.pas {DEFINE TIMERS_TEST}
24862:
24863: *****
24864: Release Notes maxbox 4.7.5.90 October 2021 mX475
24865: *****
24866:
24867: Add 26 Units + 10 Tutorials
24868: 1403 unit uPSI_SemaphorGrids;
24869: 1404 unit uXmlDates2;
24870: 1405 unit uPSI_JclTimeZones;
24871: 1406 unit uPSI_XmlDocRssParser.pas
24872: 1407 unit uPSI_RssParser.pas
24873: 1408 uPSI_SimpleParserRSS.pas
24874: 1409 unit uPSI_SimpleRSSUtils;
24875: 1410 unit uPSI_RssModel; _BlueHippo
24876: 1411 unit uPSI_StrUtil; _FIBPlus
24877: 1412 unit uPSI_TChartUtils; _TEE
24878: 1413 unit uPSI_PythonEngine.pas _P4D_Beta
24879: 1414 unit uPSI_VclPythonGUIInputOutput;
24880: 1415 unit uPSI_VarPyth;
24881: 1416 unit JclUsesUtils;
24882: 1417 unit uPSI_cParameters;
24883: 1418 unit uPSI_WDCCMisc; (uPSI_cFileTemplates);
24884: 1419 uPSI_WDCColeVariantEnum.pas
24885: 1420 unit uPSI_WDCCWinInet.pas _WDCC
24886: 1421 uPSI_PythonVersions.pas
24887: 1422 unit uPSI_PythonAction.pas
24888: 1423 uPSI_SingleList.pas
24889: 1424 unit uPSI_AdMeter.pas; Async Professional
24890: 1425 unit uPSI_neuralplanbuilder; CAI
24891: 1426 unit uPSI_neuralvolume.pas; CAI
24892: 1427 unit uPSI_neuralvolumev.pas; CAI
24893: 1428 unit uPSI_DoubleList4; CapJack
24894: 1429 unit uPSI_ByteListClass; CapJack
24895: 1430 unit uPSI_flcVectors4; FLC5
24896: 1431 unit uPSI_flcMatrix4;
24897: 1432 uPSI_CurlHttpCodes.pas
24898: 1433 unit uPSI_NeuralNetwork.pas; CAI
24899: 1434 unit uPSI_neuralfit; CAI
24900: 1435 unit uPSI_neuraldatasets; CAI
24901: 1436 unit uPSI_neuraldatasetsv.pas CAI
24902: 1437 uPSI_flcFloats.pas FLC5
24903:

```

```

24904: //Async Professional
24905:
24906: SIRegister_TFastStringStream(CL);
24907: Proc LoadJPEGResource(imagel: TImage; aJpgImage:Str);
24908: Func CreateDOSProcessRedirected3(const CommandLine, InputFile, OutputFile, ErrMsg:Str):Bool;
24909: //Type TSysCharSet = set of Char;
24910: Func SysCharSetToStr(const C: TSysCharSet): Ansistr;
24911: Func StrToSysCharSet(const S: Ansistr): TSysCharSet;
24912: Proc MaskFPUEExceptions( ExceptionsMasked :Bool; MatchPythonPrecision :Bool);
24913: Func GetOleVariantEnum( Collection : OLEVariant ) : IGetOleVariantEnum';
24914: Func GetOleVariantArrEnum( Collection : OLEVariant ) : IGetOleVariantEnum';
24915:
24916:
24917: Proc SIRegister_TSimpleParserRSS(CL: TPSPascalCompiler);
24918: begin
24919:   //with RegClassS(CL, 'TSimpleParserBase', 'TSimpleParserRSS') do
24920:   with CL.AddClassN(CL.FindClass('TSimpleParserBase'), 'TSimpleParserRSS') do begin
24921:     RegisterMethod('Proc Parse');
24922:     RegisterMethod('Proc Generate');
24923:   end;
24924: end;
24925:
24926: SERIES_MARK_FORMATS: array [TSeriesMarksStyle] of String = (
24927:   ' ',
24928:   '%0:.9g', // smsValue
24929:   '%1:.2f%%', // smsPercent
24930:   '%2:s', // smsLabel
24931:   '%2:s %1:.2f%%', // smsLabelPercent
24932:   '%2:s %0:.9g', // smsLabelValue
24933:   '%2:s', // smsLegend: not sure what it means, left for Delphi compatibility
24934:   '%1:.2f%% of %3:g', // smsPercentTotal
24935:   '%1:.2f%% of %3:g', // smsLabelPercentTotal
24936:   '%4:.9g' // smsXValue
24937: );
24938:
24939: Proc SIRegister_uXmlDates(CL: TPSPascalCompiler);
24940: begin
24941:   CL.AddTypeS('TXMLDateZeroOptions', '( dzoBlank, dzoZero, dzoNow );
24942:   Func GetTimeZoneOffset :Str');
24943:   Func GetTimestampWithTimeZone( const ADateTime : TDateTime ) :Str');
24944:   Func GetXmlDate( const ADateTime : TDateTime; const DateZeroOption : TXMLDateZeroOptions ) :Str');
24945:   Func ConvertToDelphiDateFromXml( const ADateTime :Str ) : TDateTime');
24946: end;
24947:
24948: Proc SIRegister_JclTimeZones(CL: TPSPascalCompiler);
24949: begin
24950:   CL.AddTypeS('TJclTZIValueInfo', 'record Bias : Longint; StandardBias : Integer;
24951:   + 'r; DaylightBias : Integer; StandardDate : TSystemTime; DaylightDate : TSystemTime; end');
24952:   //CL.AddTypeS('PJclTimeZoneRegInfo', '^TJclTimeZoneRegInfo // will not work');
24953:   TJclTimeZoneRegInfo', 'record DisplayDesc :Str; StandardName
24954:   + :Str; DaylightName :Str; SortIndex : Integer; MapID :Str; TZI: TJclTZIValueInfo; end');
24955:   TJclTimeZoneCallBackFunc', 'Func ( const TimeZoneRec : TJclTimeZoneRegInfo ) :Bool');
24956:   SIRegister_TJclTimeZoneInfo(CL);
24957:   SIRegister_TJclTimeZones(CL);
24958:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EDaylightSavingsNotSupported');
24959:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EAutoAdjustNotEnabled');
24960:   Func EnumTimeZones( CallBackFunc : TJclTimeZoneCallBackFunc ) :Bool);
24961:   Func IsAutoAdjustEnabled :Bool);
24962:   Func CurrentTimeZoneSupportsDaylightSavings :Bool);
24963:   Func DateCurrentTimeZoneClocksChangeToStandard : TDateTime);
24964:   Func DateCurrentTimeZoneClocksChangeToDaylightSavings : TDateTime);
24965:   Func GetCurrentTimeZoneDescription :Str);
24966:   Func GetCurrentTimeZoneDaylightSavingsPeriod :Str);
24967:   Func GetCurrentTimeZoneGMTOffset :Str);
24968:   Func GetCurrentTimeZoneUTCBias : Integer);
24969:   Func GetWMScheduledJobUTCtime( Time : TDateTime ) :Str);
24970:   Func UTCNow : TDateTime);
24971: end;
24972:
24973: type IRSSParser = interface
24974:   ['{263159EC-F94D-4BA6-9D34-5D93277D4DDC}']
24975:   Func ParserRSSFeed(XML:Str) : TRSSFeed;
24976: end;
24977:
24978: Proc SIRegister_TXmlDocRssParser(CL: TPSPascalCompiler);
24979: begin
24980:   //with RegClassS(CL, 'TInterfacedObject', 'TXmlDocRssParser') do
24981:   with CL.AddClassN(CL.FindClass('TInterfacedObject'), 'TXmlDocRssParser') do begin
24982:     RegisterMethod('Func ParserRSSDate( DateStr :Str ) : TDateTime');
24983:     RegisterMethod('Func ParserRSSFeed( XML :Str ) : TRSSFeed');
24984:   end;
24985: end;
24986:
24987: (*-----*)
24988: Proc SIRegister_RssParser(CL: TPSPascalCompiler);
24989: begin
24990:   Func ParserRSSDate( DateStr :Str ) : TDateTime);
24991:   Func ParserRSSFeed( XML :Str ) : TRSSFeed);
24992: end;

```



```

24993:
24994: unit SimpleRSSUtils;
24995: type TXMLTypeRSS = (xtRDFrss, xtRSSrss, xtAtomrss, xtiTunesrss);
24996: type TContentTypeRSS = (ctTextrss, ctHTMLrss, ctXHTMLrss);
24997: type TEncodingTypeRSS = (etBase64rss, etEscapedrss, etXMLrss);
24998:
24999: Func StringToLanguage(Language:Str): TLanguages;
25000: Func LanguageToString(Language: TLanguages):Str;
25001: Func DecodeStringRSS(Encoding: TEncodingTypeRSS; Data:Str):Str;
25002: Proc GetSkipHours(RootNode: IXMLNode; var aSimpleRSS: TSimpleRSS);
25003: Proc GetSkipDays(RootNode: IXMLNode; var aSimpleRSS: TSimpleRSS);
25004: Proc SIRegister_SimpleRSSUtils(CL: TPSPascalCompiler);
25005: Func StringToLanguage( Language :Str) : TLanguages);
25006: Func LanguageToString( Language : TLanguages) :Str);
25007: Func DecodeStringRSS( Encoding : TEncodingTypeRSS; Data :Str) :Str);
25008: Proc GetSkipHours( RootNode : IXMLNode; var aSimpleRSS : TSimpleRSS);
25009: Proc GetSkipDays( RootNode : IXMLNode; var aSimpleRSS : TSimpleRSS);
25010: Func GetPageByURL(URL:Str):Str;
25011: end;
25012:
25013: (*-----*)
25014: Proc SIRegister_StrUtil(CL: TPSPascalCompiler);
25015: begin
25016:   CL.AddTypeS('TPosition', 'record X : Longint; Y : Longint; end');
25017:   CL.AddTypeS('TFastStringStream', 'TStringStream');
25018:   SIRegister_TFastStringStream(CL);
25019:   CL.AddTypeS('TCharSet', 'set of AnsiChar');
25020:   Func _Position( AX, AY : Integer) : TPosition);
25021:   Func _MakeStr( C : Char; N : Integer) :Str);
25022:   Func _StrAsFloat( S :Str) : double);
25023:   Func _ToClientDateFmt( D :Str; caseFmt : byte) :Str);
25024:   Func _ExtractWord( Num : integer; const Str :Str; const WordDelims : TCharSet):Str);
25025:   Func _WordCount( const S :Str; const WordDelims : TCharSet) : Integer);
25026:   Func _AnsiUpperCase( const s : Ansistr) : Ansistr);
25027:   Func _CompareStrWA( const S1 :Str; const S2 : Ansistr) : Integer);
25028:   Func _AnsiCompareTextAA( const S1, S2 : Ansistr) : Integer);
25029:   Func _WildStringCompare( FirstString, SecondString :Str) :Bool);
25030:   Func _MaskCompare( const aString, Mask :Str) :Bool);
25031:   Func _SQLMaskCompare( const aString, Mask :Str) :Bool);
25032:   Func _SQLMaskCompareAW( const aString : Ansistr; const Mask :Str) :Bool);
25033:   Func _SQLMaskCompareAA( const aString, Mask : Ansistr) :Bool);
25034:   Func _SQLMaskCompare1( const aString, Mask : WideString) :Bool);
25035:   Func _TrimCLRF( const s :Str) :Str);
25036:   Func _ReplaceStr( const S, Srch, Replace :Str) :Str);
25037:   Func _ReplaceStrInSubstr( const Source String; const Srch, Replace:str; BeginPos, EndPos:integer):str);
25038:   Func _ReplaceCIStr( const S, Srch, Replace:Str; Ansi :Bool):str);
25039:   Func _ReplaceStrCIInSubstr(const Source:str; const Srch, Replace:str; BeginPos, EndPos:integer):str);
25040:   Func _FastUpperCase( const S :Str) :Str);
25041:   Proc _DoUpperCase( var S :Str; FirstPos : integer; LastPos : integer);
25042:   Proc _DoAnsiUpperCase( var S : Ansistr);
25043:   Proc _DoAnsiUpperCase1( var S :Str);
25044:   Proc _DoWideUpperCase( var S : WideString);
25045:   Proc _DoUtf8Decode( const s : AnsiChar; StrLen : integer; var ws: WideString);
25046:   Proc _DoUtf8Decode1( const s : Ansistr; var ws : WideString);
25047:   Proc _DoUtf8Decode2( const s : variant; var ws : WideString);
25048:   Func _EqualStrings( const s, sl :Str; CaseSensitive :Bool) :Bool);
25049:   Func _IifStr( Condition :Bool; const Str1, Str2 :Str) :Str);
25050:   Func _IifVariant( Condition :Bool; const Var1, Var2 : Variant) : Variant);
25051:   Func _StrOnMask( const StrIn, MaskIn, MaskOut :Str) :Str);
25052:   Func _StrIsInteger( const Str :Str) :Bool);
25053:   Func _FormatIdentifierValue( Dialect : Integer; const Value :Str) :Str);
25054:   Func _FormatIdentifier( Dialect : Integer; const Value :Str) :Str);
25055:   Func _NormalizeName( Dialect : Integer; const Name :Str) :Str);
25056:   Func _EasyFormatIdentifier( Dialect : Integer; const Value :Str; DoEasy :Bool) :Str);
25057:   Func _EqualNames( CI :Bool; const Value, Value1 :Str) :Bool);
25058:   Func _NeedQuote( const Name :Str) :Bool);
25059:   Func _EasyNeedQuote( const Name :Str) :Bool);
25060:   Func _PosCh( aCh : Char; const s :Str) : integer);
25061:   Func _PosCh1( aCh : Char; const s :Str; StartPos : integer) : integer;
25062:   Func _PosCh2( aCh : AnsiChar; const s : Ansistr; StartPos : integer) : integer;
25063:   Func _PosCI( const Substr, Str :Str) : integer);
25064:   Func _PosExt( const Substr, Str :Str; BegSub, EndSub : TCharSet) : integer);
25065:   Func _PosExtCI( const Substr, Str :Str; BegSub, EndSub : TCharSet; AnsiUpper :Bool) : integer);
25066:   Func _PosInSubstr( const substr, Str :Str; BeginPos, EndPos : integer) : integer);
25067:   Func _PosInRight( const substr, Str :Str; BeginPos : integer) : integer);
25068:   Func _PosInSubstrCI( const SearchStr, Str :Str; BeginPos, EndPos : integer) : integer);
25069:   Func _PosInSubstrExt(const SearchStr:str;Str:str;BeginPos,EndPos:integer; BegSub, EndSub:TCharSet):int);
25070:   Func _PosInSubstrCIExt(const SearchStr,Str:Str;BeginPos, EndPos : integer; BegSub,EndSub :TCharSet):int);
25071:   Func _FirstPos( Search : array of Char; const InString :Str) : integer;
25072:   Func _FirstPos1( Search : array of AnsiChar; const InString : Ansistr) : integer;
25073:   Func _LastChar( const Str :Str) : Char);
25074:   Func _QuotedStr( const S :Str) :Str);
25075:   Func _CutQuote( const S :Str) :Str);
25076:   Func _StringInArray( const s :Str; const a : array of String) :Bool);
25077:   Func _IsBlank( const Str :Str) :Bool);
25078:   Func _IsBeginPartStr11( const PartStr, TargetStr : Ansistr) :Bool;
25079:   Func _IsBeginPartStrWA( const PartStr :Str; const TargetStr : Ansistr) :Bool);
25080:   Func _IsBeginPartStrVarA( const PartStr : Variant; const TargetStr : Ansistr) :Bool);
25081:   Func _IsBeginPartStr12( const PartStr, TargetStr : WideString) :Bool;

```

```

25082: Proc _DoLowerCase13( var Str :Str);
25083: Proc _DoLowerCase14( var Str : Ansistr);
25084: Proc _DoTrim( var Str :Str);
25085: Func _VarTrimRight( const str : Variant) : Variant);
25086: Proc _DoTrimRight15( var Str :Str);
25087: Proc _DoTrimRight16( var Str : WideString);
25088: Proc _DoTrimRight17( var Str : Ansistr);
25089: Proc _DoCopy18( const Source :Str; var Dest :Str; Index, Count : Integer);
25090: Proc _DoCopy19( const Source : Ansistr; var Dest : Ansistr; Index, Count : Integer);
25091: Func _FastTrim( const S :Str) :Str);
25092: Func _FastCopy20( const S :Str; Index : integer; Count : Integer) :Str;
25093: Func _FastCopy21( const S : Ansistr; Index : integer; Count : Integer) : Ansistr;
25094: Proc _SLDifference( ASL, BSL : TStringList; ResultSL : TStringList);
25095: Func _EmptyStrings( SL : TStringList) :Bool);
25096: Proc _DeleteEmptyStr( Src : TStringList);
25097: Func _NonAnsiSortCompareStrings( SL : TStringList; Index1, Index2 : Integer) : Integer);
25098: Func _FindInDiapazon(SL:TStrList;const S:str;const StartIndex,EndIndex:int;AnsiCompare:bool;var
Index:Int):bool);
25099: Func _NonAnsiIndexOf( SL : TStringList; const S :Str) : integer);
25100: Proc _GetNameAndValue( const s :Str; var Name, Value : Ansistr);
25101: Func _StrToDateFmt( const ADate, Fmt :Str) : TDateTime);
25102: Func _DateToSQLStr( const ADate : TDateTime) :Str);
25103: Func _ValueFromStr( const Str :Str) :Str);
25104: Func _WideUpperCase( const S : WideString) : WideString);
25105: Func _Q_StrLen( P : AnsiChar) :Card);
25106: Func _IsOldParamName( const ParamName :Str) :Bool);
25107: Func _IsNewParamName( const ParamName :Str) :Bool);
25108: Func _IsMasParamName( const ParamName :Str) :Bool);
25109: Func _GUIDAsString( const AGUID : TGUID) : Ansistr);
25110: Proc _GUIDAsStringToPChar( const AGUID : TGUID; Dest : AnsiChar);
25111: Func _StringAsGUID( const AStr : Ansistr) : TGUID);
25112: Func _CompareStrAndGuid( GUID : TGUID; const Str : Ansistr) : integer);
25113: Func _CompareStrAndGuidP( GUID : TGUID; const Str : Ansistr) : integer);
25114: Func _IsEqualGUIDs( const guid1, guid2 : TGUID) :Bool);
25115: Func _IsNumericStr( const Str :Str) :Bool);
25116: Func _IsEmptyStr( const Str :Str) :Bool);
25117: Func _CompareStrTrimmed( const Str1, Str2 : PChar; Len : integer) : integer);
25118: Func _CharInSet( C : AnsiChar; const CharSet : TSysCharSet) :Bool);
25119: Func DecodeBase58(const Input:Str): TByteArray;
25120: Func StreamToByteArray(Stream: TStream): TBytes;
25121: Func InterpolateRGB(AColor1, AColor2: Integer; ACoeff: Double): Integer;
25122: Proc SetPropDefaults(AObject: TPersistent; APropNames: array of String);
25123: //Hashed := HashSHA256(HashSHA256(Copy(Decoded, 0, 21)));
25124: Proc ValidateBitcoinAddress(const Address:Str; DHashSHA256: TByteArray);
25125: end;
25126:
25127: //type TPyObject = PPyObject; this is all a part of prerelease to get a decent bufferpointer layout!
25128: //type PPyObject = TPyObject;
25129: The engine has a property, PythonEngine.InitScript:
25130: PythonEngine.InitScript.Add('import sys');
25131: PythonEngine.InitScript.Add('sys.path.insert(0, 'path/to/script');
25132:
25133: (*-----*)
25134: Proc SIRegister_PythonEngine(CL: TPSPascalCompiler);
25135: begin
25136: CL.AddTypeS('TPythonVersionProp', 'record DllName:Str; RegVersion:Str; APIVersion: Integer; end');
25137: 'PYT_METHOD_BUFFER_INCREASE', 'LongInt').SetInt( 10);
25138: 'PYT_MEMBER_BUFFER_INCREASE', 'LongInt').SetInt( 10);
25139: 'PYT_GETSET_BUFFER_INCREASE', 'LongInt').SetInt( 10);
25140: 'METH_VARARGS', 'LongWord').SetUInt( $0001);
25141: 'METH_KEYWORDS', 'LongWord').SetUInt( $0002);
25142: 'CO_OPTIMIZED', 'LongWord').SetUInt( $0001);
25143: 'CO_NEWLOCALS', 'LongWord').SetUInt( $0002);
25144: 'CO_VARARGS', 'LongWord').SetUInt( $0004);
25145: 'CO_VARKEYWORDS', 'LongWord').SetUInt( $0008);
25146: Py_LT', 'LongInt').SetInt( 0);
25147: Py_LE', 'LongInt').SetInt( 1);
25148: Py_EQ', 'LongInt').SetInt( 2);
25149: Py_NE', 'LongInt').SetInt( 3);
25150: Py_GT', 'LongInt').SetInt( 4);
25151: Py_GE', 'LongInt').SetInt( 5);
25152: CL.AddTypeS('TRichComparisonOpcode', '( pyLT, pyLE, pyEQ, pyNE, pyGT, pyGE );
25153: C_Long', 'Integer'); C_ULong', 'Cardinal');
25154: C_Long', 'NativeInt'); C_ULong', 'NativeUInt');
25155: TPFlag', '( tpfHeapType, tpfBaseType, tpfReady, tpfReadyng, tpf
25156: +HaveGC, tpVectorCall, tpMethodDescriptor, tpHaveVersionTag, tpValidVersion'
25157: +Tag, tpIsAbstract, tpLongSubclass, tpListSubClass, tpTupleSubclass, tpByte'
25158: +sSubclass, tpBaseExcSubclass, tpTypeSubclass );
25159: CL.AddTypeS('TPFlags', 'set of TPFlag');
25160: CL.AddConstantN('TPFLAGS_DEFAULT', 'LongInt').Value.ts32 := ord(tpfBaseType) or ord(tpHaveVersionTag);
25161: 'single_input', 'LongInt').SetInt( 256);
25162: 'file_input', 'LongInt').SetInt( 257);
25163: 'eval_input', 'LongInt').SetInt( 258);
25164: 'PyUnicode_WCHAR_KIND', 'LongInt').SetInt( 0);
25165: 'PyUnicode_1BYTE_KIND', 'LongInt').SetInt( 1);
25166: 'PyUnicode_2BYTE_KIND', 'LongInt').SetInt( 2);
25167: 'PyUnicode_4BYTE_KIND', 'LongInt').SetInt( 4);
25168: T_SHORT', 'LongInt').SetInt( 0);
25169: T_INT', 'LongInt').SetInt( 1);

```

```

25170: 'T_LONG', 'LongInt').SetInt( 2);
25171: 'T_FLOAT', 'LongInt').SetInt( 3);
25172: 'T_DOUBLE', 'LongInt').SetInt( 4);
25173: 'T_STRING', 'LongInt').SetInt( 5);
25174: 'T_OBJECT', 'LongInt').SetInt( 6);
25175: 'T_CHAR', 'LongInt').SetInt( 7);
25176: 'T_BYTE', 'LongInt').SetInt( 8);
25177: 'T_UBYTE', 'LongInt').SetInt( 9);
25178: 'T_USHORT', 'LongInt').SetInt( 10);
25179: 'T_UINT', 'LongInt').SetInt( 11);
25180: 'T_ULONG', 'LongInt').SetInt( 12);
25181: 'T_STRING_INPLACE', 'LongInt').SetInt( 13);
25182: 'T_OBJECT_EX', 'LongInt').SetInt( 16);
25183: 'PY_READONLY', 'LongInt').SetInt( 1);
25184: //CL.AddConstantN('RO', '').SetString( READONLY);
25185: CL.AddConstantN('READ_RESTRICTED', 'LongInt').SetInt( 2);
25186: CL.AddConstantN('PY_WRITE_RESTRICTED', 'LongInt').SetInt( 4);
25187: CL.AddTypeS('TPyMemberType', '( mtShort, mtInt, mtLong, mtFloat, mtDouble, mt'
25188:   + 'String, mtObject, mtChar, mtByte, mtUShort, mtUInt, mtULong, mtStringInplace, mtObjectEx );
25189: CL.AddTypeS('TPyMemberFlag', '( mfDefault, mfReadOnly, mfReadRestricted, mfWriteRestricted, mfRestricted
);
25190: CL.AddTypeS('TBasicService', '( pbsGetAttr, pbsSetAttr, pbsRepr, pbsCompa'
25191:   + 're, pbsHash, pbsStr, pbsGetAttrO, pbsSetAttrO, pbsCall, pbsTraverse, pbsClear, pbs'
25192:   + 'RichCompare, pbsIter, pbsIterNext );
25193: CL.AddTypeS('TBasicServices', 'set of TBasicService);
25194: CL.AddTypeS('TNumberService', '( pnsAdd, pnsSubtract, pnsMultiply, pnsRem'
25195:   + 'ainder, pnsDivmod, pnsPower, pnsNegative, pnsPositive, pnsAbsolute, pnsInvert, pn'
25196:   + 'sLShift, pnsRShift, pnsAnd, pnsXor, pnsOr, pnsInt, pnsFloat, pnsFloorDivide, pnsTr'
25197:   + 'ueDivide, pnsMatrixMultiply, pnsBool );
25198: CL.AddTypeS('TNumberServices', 'set of TNumberService');
25199: CL.AddTypeS('TInplaceNumberService', '( pnsInplaceAdd, pnsInplaceSubtra'
25200:   + 'ct, pnsInplaceMultiply, pnsInplaceRemainder, pnsInplacePower, pnsInplaceLShift'
25201:   + ', pnsInplaceRShift, pnsInplaceAnd, pnsInplaceXor, pnsInplaceOr, pnsInplaceFloor'
25202:   + 'Divide, pnsInplaceTrueDivide, pnsInplaceMatrixMultiply );
25203: CL.AddTypeS('TInplaceNumberServices', 'set of TInplaceNumberService');
25204: CL.AddConstantN('PY_CR', 'Char').SetString( #13);
25205: 'PY_LF', 'Char').SetString( #10);
25206: 'PY_TAB', 'Char').SetString( #09);
25207: //TypeS('PP_frozen', '^P_frozen // will not work');
25208: //TypeS('P_frozen', '^_frozen // will not work');
25209: //TypeS('PPPyObject', '^PyObject // will not work');
25210: //TypeS('PPPyObject', '^PyObject // will not work');
25211: //TypeS('PPPyObject', '^PPPyObject // will not work');
25212: //TypeS('PPPyObject', '^PyObject // will not work');
25213: //TypeS('PPPyObject', '^PySliceObject // will not work');
25214: //TypeS('binaryfunc', 'integer// will not work');
25215: CL.AddTypeS('Py_complex', 'record real : double; imag : double; end');
25216: CL.AddTypeS('PyMethodDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25217:   + 'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_method : PPyMethodDef; end');
25218: CL.AddTypeS('PyMemberDescrObject', '^PyMemberDescrObject // will not work');
25219: CL.AddTypeS('PyMemberDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25220:   + 'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_member : PPyMemberDef; end');
25221: //CL.AddTypeS('PPyGetSetDescrObject', '^PyGetSetDescrObject // will not work');
25222: CL.AddTypeS('PyGetSetDescrObject', 'record ob_refcnt : NativeInt; ob_type : P'
25223:   + 'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_getset : PPyGetSetDef; end');
25224: //CL.AddTypeS('PyWrapperDescrObject', '^PyWrapperDescrObject // will not work');
25225: CL.AddTypeS('PyWrapperDescrObject', 'record ob_refcnt : NativeInt; ob_type : '
25226:   + 'PyTypeObject; d_type : PPyTypeObject; d_name : PPyObject; d_base : pwrapperbase; d_wrapped : Pointer;
end');
25227: //CL.AddTypeS('PPyModuleDef_Base', '^PyModuleDef_Base // will not work');
25228: CL.AddTypeS('PyModuleDef_Base', 'record ob_refcnt : NativeInt; ob_type : PPyT'
25229:   + 'ypeObject; m_index : NativeInt; m_copy : PPyObject; end');
25230: //CL.AddTypeS('PPyModuleDef', '^PyModuleDef // will not work');
25231: CL.AddTypeS('PyModuleDef', 'record m_base : PyModuleDef_Base; m_name : PAnsiC'
25232:   + 'har; m_doc : PAnsiChar; m_size : NativeInt; m_methods : PPyMethodDef; m_re'
25233:   + 'load : inquiry; m_traverse : traverseproc; m_clear : inquiry; m_free : inquiry; end');
25234: CL.AddTypeS('PyTypeObject', 'record ob_refcnt : NativeInt; ob_type : PPyTypeO'
25235:   + 'bject; ob_size : NativeInt; tp_name : PAnsiChar; tp_basicsize : NativeInt;'
25236:   + 'tp_itemsize : NativeInt; tp_dealloc : pydestructor; tp_vectorcall_offset '
25237:   + ': NativeInt; tp_getattr : getattrfunc; tp_setattr : setattrfunc; tp_as_asy'
25238:   + 'nc : Pointer; tp_repr : reprfunc; tp_as_number : PPyNumberMethods; tp_as_s'
25239:   + 'equence : PPySequenceMethods; tp_as_mapping : PPyMappingMethods; tp_hash : '
25240:   + 'hashfunc; tp_call : ternaryfunc; tp_str : reprfunc; tp_getattro : getattr'
25241:   + 'rofunc; tp_setattro : setattrofunc; tp_as_buffer : Pointer; tp_flags : C_UL'
25242:   + 'ong; tp_doc : PAnsiChar; tp_traverse : traverseproc; tp_clear : inquiry; t'
25243:   + 'p_richcompare : richcmpfunc; tp_weaklistoffset : NativeInt; tp_iter : geti'
25244:   + 'terfunc; tp_iternext : iternextfunc; tp_methods : PPyMethodDef; tp_members'
25245:   + ' : PPyMemberDef; tp_getset : PPyGetSetDef; tp_base : PPyTypeObject; tp_dic'
25246:   + 't : PPyObject; tp_descr_get : descrgetfunc; tp_descr_set : descrsetfunc; t'
25247:   + 'p_dictoffset : NativeInt; tp_init : initproc; tp_alloc : allocfunc; tp_new'
25248:   + ' : newfunc; tp_free : pydestructor; tp_is_gc : inquiry; tp_bases : PPyObje'
25249:   + 'ct; tp_mro : PPyObject; tp_cache : PPyObject; tp_subclasses : PPyObject; t'
25250:   + 'p_weaklist : PPyObject; tp_del : PyDestructor; tp_version_tag : Card; '
25251:   + 'tp_finalize : PyDestructor; tp_vectorcall : Pointer; tp_xxx1 : NativeInt; '
25252:   + 'tp_xxx2 : NativeInt; tp_xxx3 : NativeInt; tp_xxx4 : NativeInt; tp_xxx5 : N'
25253:   + 'ativeInt; tp_xxx6 : NativeInt; tp_xxx7 : NativeInt; tp_xxx8 : NativeInt; t'
25254:   + 'p_xxx9 : NativeInt; tp_xxx10 : NativeInt; end');
25255: CL.AddTypeS('PPyInterpreterState', '^_Pointer');
25256: CL.AddTypeS('PPyThreadState', '^_Pointer');

```

```

25257: //CL.AddTypeS('PNode', '^node // will not work');
25258: CL.AddTypeS('node', 'record n_type : smallint; n_str : PAnsiChar; n_lineno : '
25259: + 'integer; n_col_offset : integer; n_nchildren : integer; n_child : PNode; end');
25260: //CL.AddTypeS('PPyCompilerFlags', '^PyCompilerFlags // will not work');
25261: CL.AddTypeS('PyCompilerFlags', 'record flags : integer; cf_feature_version:integer; end');
25262: CL.AddConstantN('PyDateTime_DATE_DATASIZE', 'LongInt').SetInt( 4);
25263: ('PyDateTime_TIME_DATASIZE', 'LongInt').SetInt( 6);
25264: ('PyDateTime_DATETIME_DATASIZE', 'LongInt').SetInt( 10);
25265: CL.AddTypeS('PyGILState_STATE', '( PyGILState_LOCKED, PyGILState_UNLOCKED );
25266: CL.AddClassN(CL.FindClass('TOBJECT'), 'EDLLLoadError');
25267: SIRegister_EDLLImportError(CL);
25268: SIRegister_EPythonError(CL);
25269: TOBJECT', 'EPyExecError');
25270: TOBJECT', 'EPyException');
25271: TOBJECT', 'EPyStandardError');
25272: TOBJECT', 'EPyArithmeticError');
25273: TOBJECT', 'EPyLookupError');
25274: TOBJECT', 'EPyAssertionError');
25275: TOBJECT', 'EPyAttributeError');
25276: TOBJECT', 'EPyEOFError');
25277: TOBJECT', 'EPyFloatingPointError');
25278: TOBJECT', 'EPyEnvironmentError');
25279: TOBJECT', 'EPyIOError');
25280: TOBJECT', 'EPyOSError');
25281: TOBJECT', 'EPyImportError');
25282: TOBJECT', 'EPyIndexError');
25283: TOBJECT', 'EPyKeyError');
25284: TOBJECT', 'EPyKeyboardInterrupt');
25285: TOBJECT', 'EPyMemoryError');
25286: TOBJECT', 'EPyNameError');
25287: TOBJECT', 'EPyOverflowError');
25288: TOBJECT', 'EPyRuntimeError');
25289: TOBJECT', 'EPyNotImplementedError');
25290: PySyntaxError(CL);
25291: TOBJECT', 'EPyIndentationError');
25292: TOBJECT', 'EPyTabError');
25293: TOBJECT', 'EPySystemError');
25294: TOBJECT', 'EPySystemExit');
25295: TOBJECT', 'EPyTypeError');
25296: TOBJECT', 'EPyUnboundLocalError');
25297: TOBJECT', 'EPyValueError');
25298: TOBJECT', 'EPyUnicodeError');
25299: TOBJECT', 'UnicodeEncodeError');
25300: TOBJECT', 'UnicodeDecodeError');
25301: TOBJECT', 'UnicodeTranslateError');
25302: TOBJECT', 'EPyZeroDivisionError');
25303: TOBJECT', 'EPyStopIteration');
25304: TOBJECT', 'EPyWarning');
25305: TOBJECT', 'EPyUserWarning');
25306: TOBJECT', 'EPyDeprecationWarning');
25307: TOBJECT', 'PendingDeprecationWarning');
25308: TOBJECT', 'FutureWarning');
25309: TOBJECT', 'EPySyntaxWarning');
25310: TOBJECT', 'EPyRuntimeWarning');
25311: TOBJECT', 'EPyReferenceError');
25312: TOBJECT', 'EPyWindowsError');
25313: CL.AddConstantN('kMaxLines', 'LongInt').SetInt( 1000);
25314: CL.AddConstantN('kMaxLineLength', 'LongInt').SetInt( 256);
25315: CL.AddTypeS('TSendDataEvent', 'Proc ( Sender : TObject; const Data : Ansistr);
25316: TReceiveDataEvent', 'Proc ( Sender : TObject; var Data : Ansistr);
25317: TSendUniDataEvent', 'Proc ( Sender : TObject; const Data : UnicodeString);
25318: TReceiveUniDataEvent', 'Proc ( Sender : TObject; var Data : UnicodeString);
25319: IOChar', 'WideChar');
25320: IOString', 'UnicodeString');
25321: TIOStringList', 'TStringList');
25322: SIRegister_TPythonInputOutput(CL);
25323: SIRegister_TDynamicDll(CL);
25324: SIRegister_TPythonInterface(CL);
25325: CL.AddTypeS('TDatetimeConversionMode', '( dcmToTuple, dcmToDatetime );
25326: //CL.AddConstantN('DEFAULT_DATETIME_CONVERSION_MODE', '').SetString( dcmToTuple);
25327: CL.AddClassN(CL.FindClass('TOBJECT'), 'TEngineClient');
25328: CL.AddTypeS('TPathInitializationEvent', 'Proc ( Sender : TObject; var Path :Str);
25329: //CL.AddTypeS('TSysPathInitEvent', 'Proc ( Sender : TObject; PathList : PPyObject);
25330: CL.AddTypeS('TPythonFlag', '(pfDebug, pfInteractive, pfNoSite, pfOptimize, pfVerbose, pfFrozenFlag
25331: , pfIgnoreEnvironmentFlag);
25332: CL.AddTypeS('TPythonFlags', 'set of TPythonFlag');
25333: SIRegister_TTracebackItem(CL);
25334: SIRegister_TPythonTraceback(CL);
25335: SIRegister_TPythonEngine(CL);
25336: SIRegister_TEngineClient(CL);
25337: CL.AddClassN(CL.FindClass('TOBJECT'), 'TMethodsContainer');
25338: CL.AddClassN(CL.FindClass('TOBJECT'), 'TEventDefs');
25339: SIRegister_TEventDef(CL);
25340: SIRegister_TEventDefs(CL);
25341: SIRegister_TMethodsContainer(CL);
25342: SIRegister_TMembersContainer(CL);
25343: SIRegister_TGetSetContainer(CL);
25344: CL.AddClassN(CL.FindClass('TOBJECT'), 'TPythonModule');
25345: CL.AddClassN(CL.FindClass('TOBJECT'), 'TErrors');

```

```

25346: CL.AddTypeS('TErrorType', '( etString, etClass );
25347: SIRegister_TParentClassError(CL);
25348: SIRegister_TError(CL);
25349: SIRegister_TErrors(CL);
25350: SIRegister_TPythonModule(CL);
25351: CL.AddClassN(CL.FindClass('TOBJECT'),'TPythonType');
25352: SIRegister_TPyObject(CL);
25353: //CL.AddTypeS('TPyObjectClass', 'class of TPyObject');
25354: CL.AddTypeS('TMappingService', '( msLength, msSubscript, msAssSubscript );
25355: CL.AddTypeS('TMappingServices', 'set of TMappingService');
25356: //CL.AddTypeS('TJclEmailReadOptions', 'set of TJclEmailReadOption');
25357: SIRegister_TTypeServices(CL);
25358: SIRegister_TPythonType(CL);
25359: CL.AddTypeS('TGetDataEvent', 'Proc ( Sender : TObject; var Data : Variant);
25360: CL.AddTypeS('TSetDataEvent', 'Proc ( Sender : TObject; Data : Variant);
25361: //CL.AddTypeS('TExtGetDataEvent', 'Proc ( Sender : TObject; var Data : PPyObject);
25362: //CL.AddTypeS('TExtSetDataEvent', 'Proc ( Sender : TObject; Data : PPyObject);
25363: SIRegister_TPythonDelphiVar(CL);
25364: SIRegister_TPyVar(CL);
25365: CL.AddTypeS('TThreadExecMode', '( emNewState, emNewInterpreter );
25366: SIRegister_TPythonThread(CL);
25367: (('Func pyio_write( self, args : PPyObject) : PPyObject');
25368: C'Func pyio_read( self, args : PPyObject) : PPyObject');
25369: C'Func pyio_SetDelayWrites( self, args : PPyObject) : PPyObject');
25370: C'Func pyio_SetMaxLines( self, args : PPyObject) : PPyObject');
25371: C'Func pyio_GetTypesStats( self, args : PPyObject) : PPyObject'); }
25372: Func GetPythonEngine : TPythonEngine);
25373: Func PythonOK :Bool);
25374: //Func PythonToDelphi( obj : PPyObject) : TPyObject);
25375: //Func IsDelphiObject( obj : PPyObject) :Bool);
25376: // Proc PyObjectDestructor( pSelf : PPyObject);
25377: //Proc FreeSubtypeInst( ob : PPyObject);
25378: //Func PyType_HasFeature( AType : PPyTypeObject; AFlag : Integer) :Bool);
25379: Func SysVersionFromDLLName( const DLLFileName :Str) :Str);
25380: Proc PythonVersionFromDLLName( LibName :Str; out MajorVersion,MinorVersion : integer);
25381: Func IsPythonVersionRegistered(PythonVersion:str; out InstallPath:string;out AllUserInstall:Bool) Bool);
25382: Proc MaskFPUEExceptions( ExceptionsMasked :Bool; MatchPythonPrecision :Bool);
25383: Func CleanStringI3( const s : Ansistr; AppendLF :Bool) : Ansistr;
25384: Func CleanStringI4( const s : UnicodeString; AppendLF :Bool) : UnicodeString;
25385: Proc setgPythonEngine(mypyengine: TPythonEngine);
25386: Func getgPythonEngine: TPythonEngine;
25387: end;
25388:
25389: Proc SIRegister_VclPythonGUIInputOutput(CL: TPSPascalCompiler);
25390: begin
25391: CL.AddConstantN('WM_WriteOutput','LongInt').SetInt( WM_USER + 1);
25392: SIRegister_TPythonGUIInputOutput(CL);
25393: CL.AddDelphiFunction('Proc Register');
25394: end;
25395:
25396: (*-----*)
25397: Proc SIRegister_TDynamicDll(CL: TPSPascalCompiler);
25398: begin
25399: //with RegClassS(CL,'TComponent', 'TDynamicDll') do
25400: with CL.AddClassN(CL.FindClass('TComponent'),'TDynamicDll') do begin
25401: Constructor Create( AOwner : TComponent);
25402: Proc Free;
25403: Proc OpenDll( const adllName :Str);
25404: Func IsHandleValid :Bool);
25405: Proc LoadDll);
25406: Proc UnloadDll);
25407: Proc Quit);
25408: RAutoLoad', 'Boolean', iptrw);
25409: RAutoUnload', 'Boolean', iptrw);
25410: RDllName', 'string', iptrw);
25411: RDllPath', 'string', iptrw);
25412: RAPIVersion', 'Integer', iptrw);
25413: RRegVersion', 'string', iptrw);
25414: RFatalAbort', 'Boolean', iptrw);
25415: RFatalMsgDlg', 'Boolean', iptrw);
25416: RUseLastKnownVersion', 'Boolean', iptrw);
25417: ROnAfterLoad', 'TNotifyEvent', iptrw);
25418: ROnBeforeLoad', 'TNotifyEvent', iptrw);
25419: ROnBeforeUnload', 'TNotifyEvent', iptrw);
25420: end;
25421: end;
25422:
25423: (*-----*)
25424: Proc SIRegister_TPythonEngine(CL: TPSPascalCompiler);
25425: begin
25426: //with RegClassS(CL,'TPythonInterface', 'TPythonEngine') do
25427: with CL.AddClassN(CL.FindClass('TPythonInterface'),'TPythonEngine') do begin
25428: Constructor Create( AOwner : TComponent);
25429: Proc Free;
25430: Proc SetPythonHome( const PythonHome : UnicodeString);
25431: Proc SetProgramName( const ProgramName : UnicodeString);
25432: Func IsType( ob : PPyObject; obt : PPyTypeObject) :Bool);
25433: Func Run_CommandAsString(const command:Ansistr; mode:Integer):str);
25434: Func Run_CommandAsObject(const command : Ansistr; mode : Integer) : PPyObject);

```



```

25435: Func Run_CommandAsObjectWithDict(const command:Ansistr;mode:Integer;locals,globals:PPyObject):
PPyObject');
25436: Func EncodeString1(const str : UnicodeString) : Ansistr;;
25437: Func EncodeString2(const str : Ansistr) : Ansistr;;
25438: Func EncodeWindowsFilePath(const str :Str) : Ansistr');
25439: Proc Initialize2;
25440: Proc DoOpenDll2(const aDllName :Str);
25441: Proc SetProgramArgs2;
25442: Proc ExecString(const command : Ansistr);
25443: Proc ExecStr(const command : Ansistr); //alias
25444: Proc ExecString3(const command : Ansistr); //alias
25445: Proc ExecStrings4(strings : TStrings);
25446: Proc ExecStrings(strings : TStrings); //alias
25447: Func EvalString5(const command : Ansistr) : PPyObject;
25448: Func EvalStringAsStr(const command : Ansistr) :Str');
25449: Func EvalStr(const command : Ansistr) :Str');
25450: Func EvalStrings6(strings : TStrings) : PPyObject;
25451: Proc ExecString7(const command:Ansistr; locals,globals:PPyObject);
25452: Proc ExecStrings8(strings : TStrings; locals, globals : PPyObject);
25453: Func EvalString9(const command:Ansistr; locals, globals : PPyObject): PPyObject;
25454: Func EvalStrings10(strings : TStrings; locals, globals : PPyObject): PPyObject;
25455: Func EvalStringsAsStr(strings : TStrings) :Str');
25456: Func EvalPyFunction(pyfunc, pyargs : PPyObject) : Variant');
25457: Func EvalFunction(pyfunc:PPyObject;const args:array of const): Variant');
25458: Func EvalFunctionNoArgs(pyfunc : PPyObject) : Variant');
25459: Func CheckEvalSyntax(const str : Ansistr) :Bool');
25460: Func CheckExecSyntax(const str : Ansistr) :Bool');
25461: Func CheckSyntax(const str : Ansistr; mode : Integer) :Bool');
25462: Proc RaiseError');
25463: Func PyObjectAsString(obj : PPyObject) :Str');
25464: Proc DoRedirectIO');
25465: Proc AddClient(client : TEngineClient);
25466: Proc RemoveClient(client : TEngineClient);
25467: Func FindClient(const aName :Str) : TEngineClient');
25468: Func TypeByName(const aTypeName : Ansistr) : PPyTypeObject');
25469: Func ModuleByName(const aModuleName : Ansistr) : PPyObject');
25470: Func MethodsByName(const aMethodsContainer :Str) : PPyMethodDef');
25471: Func VariantAsPyObject(const V : Variant) : PPyObject');
25472: Func PyObjectAsVariant(obj : PPyObject) : Variant');
25473: Func VarRecAsPyObject(const v : TVarRec) : PPyObject');
25474: Func MakePyTuple(const objects : array of PPyObject) : PPyObject');
25475: Func MakePyList(const objects : array of PPyObject) : PPyObject');
25476: Func ArrayToPyTuple(const items : array of const) : PPyObject');
25477: Func ArrayToPyList(const items : array of const) : PPyObject');
25478: Func ArrayToPyDict(const items : array of const) : PPyObject');
25479: Func StringsToPyList(strings : TStrings) : PPyObject');
25480: Func StringsToPyTuple(strings : TStrings) : PPyObject');
25481: Proc PyListToStrings(list : PPyObject; strings : TStrings);
25482: Proc PyTupleToStrings(tuple : PPyObject; strings : TStrings);
25483: Func ReturnNone : PPyObject');
25484: Func ReturnTrue : PPyObject');
25485: Func ReturnFalse : PPyObject');
25486: Func FindModule(const ModuleName : Ansistr) : PPyObject');
25487: Func FindFunction(const ModuleName, FuncName : Ansistr) : PPyObject');
25488: Func SetToList(data : Pointer; size : Integer) : PPyObject');
25489: Proc ListToSet(List : PPyObject; data : Pointer; size : Integer);
25490: Proc CheckError(ACatchStopEx :Bool);
25491: Func GetMainModule : PPyObject');
25492: Func PyTimeStruct_Check(obj : PPyObject) :Bool');
25493: Func PyDate_Check(obj : PPyObject) :Bool');
25494: Func PyDate_CheckExact(obj : PPyObject) :Bool');
25495: Func PyDateTime_Check(obj : PPyObject) :Bool');
25496: Func PyDateTime_CheckExact(obj : PPyObject) :Bool');
25497: Func PyTime_Check(obj : PPyObject) :Bool');
25498: Func PyTime_CheckExact(obj : PPyObject) :Bool');
25499: Func PyDelta_Check(obj : PPyObject) :Bool');
25500: Func PyDelta_CheckExact(obj : PPyObject) :Bool');
25501: Func PyTZInfo_Check(obj : PPyObject) :Bool');
25502: Func PyTZInfo_CheckExact(obj : PPyObject) :Bool');
25503: Func PyUnicodeFromstring11(const AString : UnicodeString) : PPyObject;
25504: Func PyUnicodeFromstring12(const AString : Ansistr) : PPyObject;
25505: Func PyUnicodeAsString(obj : PPyObject) : UnicodeString');
25506: Func PyUnicodeAsUTF8String(obj : PPyObject) : RawByteString');
25507: Func PyBytesAsAnsistr(obj : PPyObject) : Ansistr');
25508: ClientCount, 'Integer', iptr);
25509: Clients, 'TEngineClient Integer', iptr);
25510: ExecModule, 'Ansistr', iptrw);
25511: ThreadState, 'PPyThreadState', iptr);
25512: Traceback, 'TPythonTraceback', iptr);
25513: LocalVars, 'PPyObject', iptrw);
25514: GlobalVars, 'PPyObject', iptrw);
25515: IOPythonModule, 'TObject', iptr);
25516: PythonHome, 'UnicodeString', iptrw);
25517: ProgramName, 'UnicodeString', iptrw);
25518: AutoFinalize, 'Boolean', iptrw);
25519: VenvPythonExe, 'string', iptrw);
25520: DatetimeConversionMode, 'TDateTimeConversionMode', iptrw);
25521: InitScript, 'TStrings', iptrw);
25522: InitThreads, 'Boolean', iptrw);

```

```

25523:   IO', 'TPythonInputOutput', iptrw);
25524:   PyFlags', 'TPythonFlags', iptrw);
25525:   RedirectIO', 'Boolean', iptrw);
25526:   UseWindowsConsole', 'Boolean', iptrw);
25527:   OnAfterInit', 'TNotifyEvent', iptrw);
25528:   OnPathInitialization', 'TPathInitializationEvent', iptrw);
25529:   OnSysPathInit', 'TSysPathInitEvent', iptrw);
25530:   end;
25531: end;
25532:
25533: Proc SIRegister_TPythonModule(CL: TPSPascalCompiler);
25534: begin
25535:   //with RegClassS(CL, 'TMethodsContainer', 'TPythonModule') do
25536:   with CL.AddClassN(CL.FindClass('TMethodsContainer'), 'TPythonModule') do begin
25537:     Constructor Create( AOwner : TComponent);
25538:     Proc MakeModule();
25539:     Proc DefineDocString();
25540:     Proc Initialize();
25541:     Proc InitializeForNewInterpreter();
25542:     Proc AddClient( client : TEngineClient);
25543:     Func ErrorByName( const AName : Ansistr) : TError';
25544:     Proc RaiseError( const error, msg : Ansistr);
25545:     Proc RaiseErrorFmt(const error, format:Ansistr;const Args:array of const);
25546:     Proc RaiseErrorObj(const error, msg : Ansistr; obj : PPyObject);
25547:     Proc BuildErrors();
25548:     Proc SetVar( const varName : Ansistr; value : PPyObject);
25549:     Func GetVar( const varName : Ansistr) : PPyObject';
25550:     Proc DeleteVar( const varName : Ansistr);
25551:     Proc ClearVars();
25552:     Proc SetVarFromVariant(const varName:Ansistr;const value:Variant);
25553:     Func GetVarAsVariant( const varName : Ansistr) : Variant';
25554:     Module', 'PPyObject', iptr);
25555:     Clients', 'TEngineClient Integer', iptr);
25556:     ClientCount', 'Integer', iptr);
25557:     DocString', 'TStringList', iptrw);
25558:     ModuleName', 'Ansistr', iptrw);
25559:     Errors', 'TErrors', iptrw);
25560:     OnAfterInitialization', 'TNotifyEvent', iptrw);
25561:   end;
25562: end;
25563:
25564: Example:
25565: Proc initDemodll;
25566: begin //mode: eval_input, file_input
25567:   try
25568:     gEngine:= TPythonEngine.Create(nil);
25569:     gEngine.AutoFinalize:= False;
25570:     gEngine.LoadDll;
25571:     gModule:= TPythonModule.Create(nil);
25572:     gModule.Engine:= gEngine;
25573:     gModule.ModuleName:= 'pydemodll';
25574:     //gModule.AddMethod( 'add', @Add, 'add(a,b) -> a+b' );
25575:     // The engine has a property, PythonEngine.InitScript:
25576:     gEngine.InitScript.Add('import sys');
25577:     gEngine.InitScript.Add('sys.path.insert(0, "path/to/script");
25578:     //gModule.Initialize;
25579:     writeln('PythonOK: in init '+botostr(PythonOK));
25580:     //gEngine.EvalStringAsStr(pycmd);
25581:   except
25582:     gEngine.raiseError;
25583:     writeln(ExceptionToString(ExceptionType, ExceptionParam));
25584:     gEngine.Free;
25585:     gModule.Free;
25586:   end;
25587: end;
25588:
25589: const LB = CR+LF;
25590: const PyModule =
25591:   'def printData(data): '+#13#10+
25592:   '  return data+data+"/n";
25593:
25594: const PYCMD = 'print("this is box")'+LB+
25595:   'import sys'+LB+
25596:   'f=open(r"C:\maXbox\maXbox3\maXbox3\maXbox3\examples\pytest2.txt", "w")'+LB+
25597:   'f.write("Hello PyWorld_mX4, \n")'+LB+
25598:   'f.write("This data will be written on the file.")'+LB+
25599:   'f.close()';
25600:
25601: Proc PYLaz_P4D_Demo;
25602: //https://wiki.freepascal.org/Python4Delphi
25603: var eng : TPythonEngine;
25604:   Out1: TPythonGUIInputOutput;
25605: begin
25606:   eng:= TPythonEngine.Create( Nil );
25607:   Out1:= TPythonGUIInputOutput.create( nil )
25608:   Out1.output:= pyMemo; //debugout.output; //memo2;
25609:   Out1.RawOutput:= False;
25610:   Out1.UnicodeIO:= False;
25611:   Out1.maxlines:= 20;

```

```

25612:   out1.displaystring('this string')
25613:   //eng.IO:= Out1;
25614:   Out1.writeline('draw the line');
25615:   eng.LoadDll;
25616:   eng.IO:= Out1;
25617:   if eng.IsHandleValid then begin
25618:     writeln('DLLhandle: '+botostr(eng.IsHandleValid))
25619:     writeln('evens: '+ eng.EvalStringAsStr('x*2 for x in range(10)'));
25620:     writeln('gauss: '+ eng.EvalStr('sum([x for x in range(101)])')); //alias
25621:     writeln('syncheck '+
25622:       botostr(eng.CheckEvalSyntax('print("powers:",[x**2 for x in range(10)])'));
25623:     eng.ExecString('print("powers:",[x**2 for x in range(10)]);
25624:     eng.ExecStr(PYCMD);
25625:     writeln('ExecSynCheck2 '+
25626:       botostr(eng.CheckExecSyntax(filetostring(PYSCRIPT))));
25627:     eng.ExecString(filetostring(PYSCRIPT));
25628:   end
25629:   else writeln('invalid library handle! '+Getlasterrortext);
25630:   writeln('PythonOK '+botostr(PythonOK));
25631:   out1.free;
25632:   //pyImport(PyModule);
25633:   eng.free;
25634: end;
25635:
25636: //python template
25637: with TPythonEngine.Create(nil) do begin
25638:   pythonhome:= 'C:\Users\max\AppData\Local\Programs\Python\Python36-32\';
25639:   try
25640:     loadDLL;
25641:     Println('Decimal: '+
25642:       EvalStr('__import__ ("decimal").Decimal(0.1)'));
25643:   except
25644:     raiseError;
25645:   finally
25646:     free;
25647:   end;
25648:
25649:   // This is a new method in VarPyth, but I'll put it here just in case...
25650:   function VarPyToStrings(const AValue : Variant; const AStrings: TStrings): Integer;
25651:   begin
25652:     Assert(Assigned(AStrings));
25653:     if VarIsPythonList(AValue) then
25654:       GetPythonEngine.PyListToStrings(
25655:         ExtractPythonObjectFrom(AValue), AStrings)
25656:     else
25657:       raise Exception.Create('Python List expected: ' + _type(AValue));
25658:     Result := AStrings.Count;
25659:   end;
25660:
25661: procedure TForm16.Button1Click(Sender: TObject);
25662: begin
25663:   with NLTK1 do begin
25664:     nltk.download('punkt');
25665:     nltk.download('averaged_perceptron_tagger');
25666:     nltk.download('maxent_ne_chunker');
25667:     nltk.download('words');
25668:     nltk.download('treebank');
25669:
25670:     var tokens := nltk.word_tokenize(memol.lines.Text);
25671:     VarPyToStrings(tokens, lbTokens.Items);
25672:   end;
25673: Proc SIRegister_VarPyth(CL: TPSPascalCompiler);
25674: begin
25675:   CL.AddTypes('TSequenceType', '( pystTuple, pystList );
25676:   Func VarPythonCreate0( AObject : TPyObject ) : Variant;
25677:   Func VarPythonCreate1( const AValue : Variant ) : Variant;
25678:   Func VarPythonCreate2( const AValues: array of const; ASequenceType : TSequenceType ) : Variant;
25679:   Func VarPythonEval( const APythonExpression : Ansistr ) : Variant';
25680:   Func VarPython : TVarType';
25681:   Func VarIsPython( const AValue : Variant ) :Bool';
25682:   Func VarAsPython( const AValue : Variant ) : Variant';
25683:   Func ExtractPythonObjectFrom( const AValue : Variant ) : TPyObject';
25684:   Func VarIsSame( const A, B : Variant ) :Bool';
25685:   Func VarIsSameType( const A, B : Variant ) :Bool';
25686:   Func VarIsPythonSequence( const AValue : Variant ) :Bool';
25687:   Func VarIsPythonMapping( const AValue : Variant ) :Bool';
25688:   Func VarIsPythonNumber( const AValue : Variant ) :Bool';
25689:   Func VarIsPythonString( const AValue : Variant ) :Bool';
25690:   Func VarIsPythonInteger( const AValue : Variant ) :Bool';
25691:   Func VarIsPythonFloat( const AValue : Variant ) :Bool';
25692:   Func VarIsPythonTuple( const AValue : Variant ) :Bool';
25693:   Func VarIsPythonList( const AValue : Variant ) :Bool';
25694:   Func VarIsPythonDict( const AValue : Variant ) :Bool';
25695:   Func VarIsPythonClass( const AValue : Variant ) :Bool';
25696:   Func VarIsPythonMethod( const AValue : Variant ) :Bool';
25697:   Func VarIsPythonFunction( const AValue : Variant ) :Bool';
25698:   Func VarIsPythonModule( const AValue : Variant ) :Bool';
25699:   Func VarIsPythonCallable( const AValue : Variant ) :Bool';
25700:   Func VarIsPythonIterator( const AValue : Variant ) :Bool';

```

```

25701: Func VarIsPythonUnicode( const AValue : Variant) :Bool');
25702: Func VarIsPythonDateTime( const AValue : Variant) :Bool');
25703: Func VarIsPythonDate( const AValue : Variant) :Bool');
25704: Func VarIsPythonTime( const AValue : Variant) :Bool');
25705: Func VarIsPythonDateTimeDelta( const AValue : Variant) :Bool');
25706: Func VarIsPythonTZInfo( const AValue : Variant) :Bool');
25707: Func VarIsBool( const AValue : Variant) :Bool');
25708: Func VarIsEnum( const AValue : Variant) :Bool');
25709: Func VarIsInstanceOf( const AInstance, AClass : Variant) :Bool');
25710: Func VarIsSubclassOf( const ADerived, AClass : Variant) :Bool');
25711: Func VarIsSubtypeOf( const ADerived, AType : Variant) :Bool');
25712: Func VarIsNone( const AValue : Variant) :Bool');
25713: Func VarIsTrue( const AValue : Variant) :Bool');
25714: Func VarModuleHasObject( const AModule : Variant; aObj: Ansistr):Bool');
25715: Func NewPythonList( const ASize : Integer) : Variant');
25716: Func NewPythonTuple( const ASize : Integer) : Variant');
25717: Func NewPythonDict : Variant');
25718: Func VarPythonAsString( AValue : Variant) :Str');
25719: Func VarPythonToVariant( AValue : Variant) : Variant');
25720: Func PyNone : Variant');
25721: Func pyEllipsis : Variant');
25722: Func pyMainModule : Variant');
25723: Func pyBuiltinModule : Variant');
25724: Func pySysModule : Variant');
25725: Func ypDatetimeModule : Variant');
25726: Func PyImport( const AModule : Ansistr) : Variant');
25727: Func pylon( const AValue : Variant) : NativeInt');
25728: Func py_type( const AValue : Variant) : Variant');
25729: Func pyiter( const AValue : Variant) : Variant');
25730: CL.AddTypeS(TVarPyEnumerator, 'record FIterator : Variant; FCurrent : Variant; end');
25731: CL.AddTypeS(TVarPyEnumerateHelper, 'record FIterable : Variant; end');
25732: Func VarPyIterate( const AValue : Variant) : TVarPyEnumerateHelper');
25733: end;
25734:
25735: Proc SIRegister_WDCCMisc(CL: TPSPascalCompiler); //WMI Delphi Code Creator
25736: begin //https://github.com/RRUZ/wmi-delphi-code-creator/tree/master/Units/Misc
25737: TProcLog, 'Proc ( const Log :Str);
25738: Proc wdc_CaptureConsoleOutput( const lpCommandLine :Str; OutPutList : TStrings);
25739: Proc wdc_MsgWarning( const Msg :Str);
25740: Proc wdc_MsgInformation( const Msg :Str);
25741: Func wdc_MsgQuestion( const Msg :Str) :Bool');
25742: Func wdc_GetFileVersion( const FileName :Str) :Str');
25743: Func wdc_GetFileDescription( const FileName :Str) :Str');
25744: Func wdc_GetTempDirectory :Str');
25745: Func wdc_GetWindowsDirectory :Str');
25746: Func wdc_GetSpecialFolder( const CSIDL : integer) :Str');
25747: Func wdc_IsWow64 :Bool');
25748: Func wdc_CopyDir( const fromDir, toDir :Str) :Bool');
25749: Proc wdc_SetGridColumnWidths( DbGrid : TDBGrid);
25750: Func wdc_Ping( const Address :Str; Retries, BufferSize : Word; Log : TStrings) :Bool');
25751: Proc wdc_ScaleImage32( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double);
25752: Proc wdc_ExtractIconFile( Icon : TIcon; const FileName :Str; IconType :Card);
25753: Proc wdc_ExtractBitmapFile( Bmp : TBitmap; const FileName :Str; IconType :Card);
25754: Proc wdc_ExtractBitmapFile32( Bmp : TBitmap; const FileName :Str; IconType :Card);
25755: Proc wdc_CheckForUpdates( Silent :Bool);
25756: end;
25757:
25758: (*-----*)
25759: Proc SIRegister_WDCCOLEVariantEnum(CL: TPSPascalCompiler);
25760: begin
25761: SIRegister_IoleVariantEnum(CL);
25762: SIRegister_IGetOleVariantEnum(CL);
25763: SIRegister_ToleVariantEnum(CL);
25764: SIRegister_ToleVariantArrayEnum(CL);
25765: Func GetOleVariantEnum( Collection : OLEVariant) : IGetOleVariantEnum');
25766: Func GetOleVariantArrEnum( Collection : OLEVariant) : IGetOleVariantEnum');
25767: end;
25768:
25769: (*-----*)
25770: Proc SIRegister_WDCCWinInet(CL: TPSPascalCompiler);
25771: begin
25772: CL.AddConstantN('WM_UWININET_THREAD_FINISHED', 'LongInt').SetInt( WM_USER + 669);
25773: CL.AddConstantN('WM_UWININET_THREAD_CANCELLED', 'LongInt').SetInt( WM_USER + 670);
25774: CL.AddTypeS('TuWinInetProcCallBack', 'Proc ( BytesRead : Integer);
25775: SIRegister_TWinInetGetThread(CL);
25776: Func wdc_GetRemoteFileSize( const Url :Str) : Integer');
25777: Proc wdc_WinInet_HttpGet( const Url :Str; Stream : TStream; CallBack : TuWinInetProcCallBack);
25778: Func wdc_WinInet_HttpGet2( const Url :Str; CallBack : TuWinInetProcCallBack) :Str;
25779: Func wdc_GetWinInetError( ErrorCode :Card) :Str');
25780: end;
25781:
25782: Proc SIRegister_PythonVersions(CL: TPSPascalCompiler);
25783: begin
25784: SIRegister_TPythonVersion(CL);
25785: CL.AddTypeS('TPythonVersions', 'array of TPythonVersion');
25786: Func CompareVersions( A, B :Str) : Integer');
25787: Func IsEXEX64( const EXEName :Str) :Bool');
25788: Func Isx64( const FileName :Str) :Bool');
25789: Func GetRegisteredPythonVersion( SysVersion :Str; out PythonVersion : TPythonVersion) :Bool');

```

```

25790: Func GetRegisteredPythonVersions : TPythonVersions');
25791: Func GetLatestRegisteredPythonVersion( out PythonVersion : TPythonVersion) :Bool';
25792: Func PythonVersionFromPath(const Path:str;out PythonVersion:TPythonVersion;AcceptVirtualEnvs:Bool):Bool);
25793: end;
25794:
25795: Example: var pyv: TPythonVersion; pyvs:TPythonVersions;
25796:   pyvs:= GetRegisteredPythonVersions;
25797:   writeln(objtostr(pyv));
25798:   for it:= 1 to length(pyvs)-1 do begin
25799:     writeln(pyvs[it].dllname)
25800:     writeln(pyvs[it].installpath)
25801:     writeln(pyvs[it].PythonExecutable )
25802:     writeln(pyvs[it].Version )
25803:   end;
25804:
25805: Proc SIRegister_TPythonAction(CL: TPSPascalCompiler);
25806: begin
25807:   //with RegClassS(CL,'TAction', 'TPythonAction') do
25808:   with CL.AddClassN(CL.FindClass('TAction'),'TPythonAction') do begin
25809:     'fRegisteredMethods', 'TList', iptrw);
25810:     'fPythonModule', 'TPythonModule', iptrw);
25811:     'fClearname', 'string', iptrw);
25812:     'fRegistername', 'string', iptrw);
25813:     'fUnregistername', 'string', iptrw);
25814:     Proc ClearMethods');
25815:     Constructor Create( AOwner : TComponent);
25816:     Func HandlesTarget( Target : TObject) :Bool';
25817:     Func Execute :Bool';
25818:     Proc UpdateTarget( Target : TObject);
25819:     Proc InitializeAction');
25820:     RegisterProperty('RegisteredMethods', 'TList', iptr);
25821:     RegisterProperty('PythonModule', 'TPythonModule', iptrw);
25822:   end;
25823: end;
25824:
25825: //V4.7.5.90
25826: procedure SIRegister_TSingleListClass(CL: TPSPascalCompiler);
25827: begin
25828:   TSingleListSortCompare
25829:   = function (Item1, Item2: Single): Integer;
25830:   TSingleDescriptor
25831:   = function (Index:Integer;Item : Single) : string;
25832:
25833:   //with RegClassS(CL,'TObject', 'TSingleListClass') do
25834:   with CL.AddClassN(CL.FindClass('TObject'),'TSingleListClass') do begin
25835:     Constructor Create');
25836:     Procedure Free;
25837:     Function Add( Item : Single) : Integer');
25838:     Procedure Clear');
25839:     Procedure SaveToStream( const S : TStream);
25840:     Procedure LoadFromStream( const S : TStream; const KeepCurrentSortType : Boolean);
25841:     Procedure SaveToFile( FileName : string);
25842:     Procedure LoadFromFile( FileName : string; const KeepCurrentSortType : Boolean);
25843:     Procedure Delete( Index : Integer);
25844:     Function ErrMsg( const Msg : string; Data : Integer) : string');
25845:     Procedure Exchange( Index1, Index2 : Integer);
25846:     Function Expand : TSingleListClass');
25847:     Function First : Single');
25848:     Function IndexOf( Value : Single) : Integer');
25849:     Procedure Insert( Index : Integer; Item : Single);
25850:     Function Last : Single');
25851:     Procedure Move( CurIndex, NewIndex : Integer);
25852:     Function Remove( Item : Single) : Integer');
25853:     Procedure Pack( NilValue : Single);
25854:     Procedure Sort( Compare : TSingleListSortCompare);
25855:     Procedure SortUp');
25856:     Procedure SortDown');
25857:     Procedure ShowList( StringList: TStringList; Descriptor: TSingleDescriptor; ClearIt:Boolean);
25858:     Function Minimum : Single');
25859:     Function Maximum : Single');
25860:     Function Range : Single');
25861:     Function Sum : Extended');
25862:     Function SumSqr : Extended');
25863:     Function Average : Extended');
25864:     Procedure CopyFrom( List : TSingleListClass; const KeepCurrentSortType : Boolean);
25865:     Procedure CopyTo( List : TSingleListClass; const KeepDestSortType : Boolean);
25866:     Procedure Push( Value : Single);
25867:     Function LifoPop( DefValue : Single) : Single');
25868:     Function FifoPop( DefValue : Single) : Single');
25869:     RegisterProperty('List', 'PSinglePtrList', iptr);
25870:     RegisterProperty('Capacity', 'Integer', iptrw);
25871:     RegisterProperty('Count', 'Integer', iptrw);
25872:     RegisterProperty('Items', 'Single Integer', iptrw);
25873:     SetDefaultProperty('Items');
25874:     RegisterProperty('SortType', 'TSingleSortOption', iptrw);
25875:   end;
25876: end;
25877:
25878: procedure SIRegister_DoubleList4(CL: TPSPascalCompiler);

```



```

25879: begin
25880: CL.AddConstantN('SDoubleListVoidError','String').SetString( 'Invalid method call (empty list)!');
25881: CL.AddConstantN('SDoubleListSortError','String').SetString( 'Invalid method call (sorted list)!');
25882: // = array[0..MaxDoubleListSize - 1] of Double;
25883: // CL.AddTypeS('TDoublePtrList', 'array[0..2147417850] of Single');
25884: CL.AddTypeS('TDoublePtrList', 'array of Single');
25885: { TDoubleListSortCompare
25886: = function (Item1, Item2: Double): Integer;
25887: TDoubleDescriptor = function (Index:Integer;Item : Double) : string; }
25888: CL.AddTypeS('TDoubleListSortCompare', 'function (Item1,Item2: Double):Integer;
25889: //TSingleDescriptor // = function (Index:Integer;Item : Single) : string;
25890: CL.AddTypeS('TDoubleDescriptor', 'function (Index:Integer;Item : Double) : string;
25891: CL.AddTypeS('TDoubleSortOption', '( DoubleSortNone, DoubleSortUpWithDup, Doub'
25892: + 'leSortUpNoDup, DoubleSortDownWithDup, DoubleSortDownNoDup );
25893: SIRegister_TDoubleList(CL);
25894: CL.AddDelphiFunction('Function DefDescDouble( Index : Integer; Item : Double) : string');
25895: end;
25896:
25897: procedure SIRegister_ByteListClass(CL: TPSPascalCompiler);
25898: begin
25899: CL.AddConstantN('SByteListVoidError','String').SetString( 'Invalid method call (empty list)!');
25900: CL.AddConstantN('SByteListSortError','String').SetString( 'Invalid method call (sorted list)!');
25901: CL.AddTypeS('TByteSortOption', '( ByteSortNone, ByteSortUpWithDup, ByteSortUp'
25902: + 'NoDup, ByteSortDownWithDup, ByteSortDownNoDup );
25903: CL.AddTypeS('TBytePtrList', 'array of Byte');
25904: TByteListSortCompare', 'function (Item1, Item2: Byte): Integer;
25905: TByteDescriptor', 'function (Index:Integer;Item : Byte) : string;
25906: SIRegister_TByteListClass(CL);
25907: CL.AddDelphiFunction('Function DefDescByte( Index : Integer; Item : Byte): string');
25908: end;
25909:
25910: procedure SIRegister_neuralplanbuilder(CL: TPSPascalCompiler);
25911: begin
25912: 'NeuralMaxStates2','LongInt').SetInt( 400);
25913: 'NeuralMaxPlans2','LongInt').SetInt( 100);
25914: CL.AddTypeS('TNeuralState2', 'array of byte');
25915: CL.AddTypeS('TProcPred2', Function ( var ST : array of byte; Action : byte): boolean);
25916: SIRegister_TActionStateList(CL);
25917: Procedure TVisitedStatesCopy( var A, B : TActionStateList);
25918: SIRegister_TPlan(CL);
25919: SIRegister_TCompositePlan(CL);
25920: end;
25921:
25922: procedure SIRegister_neuralvolume(CL: TPSPascalCompiler);
25923: begin
25924: 'csMinAvxSize','LongInt').SetInt( 16);
25925: 'csEncoderRGB','LongInt').SetInt( 0);
25926: 'csEncodeHSV','LongInt').SetInt( 1);
25927: 'csEncodeHSL','LongInt').SetInt( 2);
25928: 'csEncodeLAB','LongInt').SetInt( 3);
25929: 'csEncodeGray','LongInt').SetInt( 4);
25930: CL.AddTypeS('TNeuralFloat', 'Single');
25931: //CL.AddTypeS('TNeuralFloatPtr', '^TNeuralFloat // will not work');
25932: //CL.AddTypeS('TNeuralFloatArrPtr', '^TNeuralFloatArr // will not work');
25933: CL.AddTypeS('TNeuralIntegerArray', 'array of integer');
25934: //CL.AddTypeS('T', 'TNeuralFloat');
25935: CL.AddTypeS('PtrInt', 'Integer');
25936: SIRegister_TNNetList(CL);
25937: SIRegister_TVVolume(CL);
25938: SIRegister_TNNetVolume(CL);
25939: SIRegister_TNNetVolumePair(CL);
25940: SIRegister_TMObject(CL);
25941: SIRegister_TNNetVolumeList(CL);
25942: SIRegister_TNNetVolumePairList(CL);
25943: SIRegister_TNNetKMeans(CL);
25944: SIRegister_TNNetStringList(CL);
25945: SIRegister_TStringListInt(CL);
25946: SIRegister_TNNetDictionary(CL);
25947: Function CreateTokenizedStringList86( str : string; c : char) : TStringList;
25948: Function CreateTokenizedStringList87( c : char) : TStringList;
25949: Function HyperbolicTangent( x : TNeuralFloat) : TNeuralFloat;
25950: Function HyperbolicTangentDerivative( x : TNeuralFloat) : TNeuralFloat;
25951: Function RectifiedLinearUnit( x : TNeuralFloat) : TNeuralFloat;
25952: Function RectifiedLinearUnitDerivative( x : TNeuralFloat) : TNeuralFloat;
25953: Function RectifiedLinearUnitLeaky( x : TNeuralFloat) : TNeuralFloat;
25954: Function RectifiedLinearUnitLeakyDerivative( x : TNeuralFloat) : TNeuralFloat;
25955: Function ReLULeakyBound( x : TNeuralFloat) : TNeuralFloat;
25956: Function ReLULeakyBoundDerivative( x : TNeuralFloat) : TNeuralFloat;
25957: Function Sigmoid( x : TNeuralFloat) : TNeuralFloat;
25958: Function SigmoidDerivative( x : TNeuralFloat) : TNeuralFloat;
25959: Function Identity( x : TNeuralFloat) : TNeuralFloat;
25960: Function IdentityDerivative( x : TNeuralFloat) : TNeuralFloat;
25961: Function SoftmaxDerivative( x : TNeuralFloat) : TNeuralFloat;
25962: Function DiffAct( x : TNeuralFloat) : TNeuralFloat;
25963: Function DiffActDerivative( x : TNeuralFloat) : TNeuralFloat;
25964: Function NeuronForceMinMax88( x, pMin, pMax : TNeuralFloat) : TNeuralFloat;
25965: Function NeuronForceMinMax89( x, pMin, pMax : integer) : integer;
25966: Function NeuronForceRange( x, range : TNeuralFloat) : TNeuralFloat;
25967: Function NeuronForceMinRange( x, range : TNeuralFloat) : TNeuralFloat;

```

```

25968: Procedure rgb2hsv( r, g, b : TNeuralFloat; var h, s, v : TNeuralFloat);
25969: Procedure hsv2rgb( h, s, v : TNeuralFloat; var r, g, b : TNeuralFloat);
25970: Function hue2rgb( p, q, t : TNeuralFloat) : TNeuralFloat';
25971: Procedure rgb2hsl( r, g, b : TNeuralFloat; var h, s, l : TNeuralFloat);
25972: Procedure hsl2rgb( h, s, l : TNeuralFloat; var r, g, b : TNeuralFloat);
25973: Procedure lab2rgb( l, a, b : TNeuralFloat; var r, g, bb : TNeuralFloat);
25974: Procedure rgb2lab( r, g, b : TNeuralFloat; var l, a, bb : TNeuralFloat);
25975: Function RoundAsByte( x : TNeuralFloat) : byte';
25976: Function CompareStringListIntegerAsc( List : TStringList; Index1, Index2 : Integer) : Integer';
25977: Function CompareStringListIntegerDesc( List : TStringList; Index1, Index2 : Integer) : Integer';
25978: Function CompareNNetVolumeListAsc( const Item1, Item2 : TNNNetVolume) : Integer';
25979: Function CompareNNetVolumeListDesc( const Item1, Item2 : TNNNetVolume) : Integer';
25980: Function NeuralFloatToStr( V : TNeuralFloat) : string';
25981: Function NeuralStrToFloat( V : String) : TNeuralFloat';
25982: Procedure TestTNNNetVolume( ); //messagedialog!
25983: Procedure TestKMeans2( ); //unattendend
25984: end;
25985:
25986: (*-----*)
25987: procedure SIRegister_neuralvolumeev(CL: TPSPascalCompiler);
25988: begin
25989: Procedure SaveHandleToBitmap( OutputFileName : string; hWnd : HWND);
25990: Procedure LoadVolumeIntoTImage( V : TNNNetVolume; Image : TImage; color_encoding : integer);
25991: Procedure LoadRGBVolumeIntoTImage( V : TNNNetVolume; Image : TImage);
25992: Procedure LoadPictureIntoVolume( LocalPicture : TPicture; Vol : TNNNetVolume);
25993: Procedure LoadBitmapIntoVolume( LocalBitmap : TBitmap; Vol : TNNNetVolume);
25994: Procedure LoadImageFromFileIntoVolume( ImageFileName : string; V : TNNNetVolume);
25995: end;
25996:
25997: function GetSmallestIdxInRange(StartPos, Len: integer): integer;
25998: function GetValueCount(Value: T): integer;
25999: GetParentProcessName2
26000: writeln('GetParentProcessName2: '+GetProcessName(GetParentProcessID(getprocessID)));
26001: Function DefDescByte( Index : Integer; Item : Byte) : string';
26002: function NumStringParts(SourceStr,Delimiter:String):Integer';
26003: function GetStringPart(SourceStr,Delimiter:String;Num:Integer):string';
26004: Function InverseCompareResult( const C : TCompareResult) : TCompareResult';
26005:
26006: function GetWindowProcessID(const Wnd: Windows.HWND): Windows.DWORD;
26007: var
26008: GetWindowThreadProcessId: function(Wnd: Windows.HWND; // API fn prototype
26009: lpdwProcessId: Windows.PDWORD): Windows.DWORD; stdcall;
26010: begin
26011: // We load the API function explicitly to make the routine compatible with as
26012: // many compilers as possible: the definition of GetWindowThreadProcessId in
26013: // Delphi and FreePascal Windows units varies across compilers
26014: GetWindowThreadProcessId := GetProcAddress(
26015: GetModuleHandle('user32.dll'), 'GetWindowThreadProcessId' );
26016: if Assigned(GetWindowThreadProcessId) and Windows.IsWindow(Wnd) then
26017: GetWindowThreadProcessId(Wnd, @Result)
26018: else
26019: Result := Windows.DWORD(-1);
26020: end;
26021:
26022:
26023: Procedure TestVectorClassExtended - type TInt64Array = TInt64ArrayClass;
26024:
26025: procedure SIRegister_TVVectorClass(CL: TPSPascalCompiler);
26026: begin
26027: //with RegClassS(CL,'TVectorBaseArray', 'TVectorClass') do
26028: //with CL.AddClassN(CL.FindClass('TVectorBaseArray'),'TVectorClass') do
26029: with CL.AddClassN(CL.FindClass('TExtendedArray'),'TVectorClass') do begin
26030: Function CreateInstance : AType';
26031: Procedure Add( const V : MFloat);
26032: Procedure Add1( const V : PMFloat; const Count : Integer);
26033: Procedure Add2( const V : PInt64; const Count : Integer);
26034: Procedure Add3( const V : MFloatArray);
26035: Procedure Add4( const V : Int64Array);
26036: Procedure Add5( const V : TVectorBaseArray);
26037: Procedure Add6( const V : TInt64Array);
26038: Procedure Add7( const V : TObject);
26039: Procedure Subtract8( const V : MFloat);
26040: Procedure Subtract9( const V : PMFloat; const Count : Integer);
26041: Procedure Subtract10( const V : PInt64; const Count : Integer);
26042: Procedure Subtract11( const V : MFloatArray);
26043: Procedure Subtract12( const V : Int64Array);
26044: Procedure Subtract13( const V : TVectorBaseArray);
26045: Procedure Subtract14( const V : TInt64Array);
26046: Procedure Subtract15( const V : TObject);
26047: Procedure Multiply16( const V : MFloat);
26048: Procedure Multiply17( const V : PMFloat; const Count : Integer);
26049: Procedure Multiply18( const V : PInt64; const Count : Integer);
26050: Procedure Multiply19( const V : MFloatArray);
26051: Procedure Multiply20( const V : Int64Array);
26052: Procedure Multiply21( const V : TVectorBaseArray);
26053: Procedure Multiply22( const V : TInt64Array);
26054: Procedure Multiply23( const V : TObject);
26055: Function DotProduct24( const V : PMFloat; const Count : Integer): MFloat;
26056: Function DotProduct25( const V : PInt64; const Count : Integer) : MFloat;

```

```

26057:   Function DotProduct26( const V : MFloatArray) : MFloat;
26058:   Function DotProduct27( const V : Int64Array) : MFloat;
26059:   Function DotProduct28( const V : TVectorBaseArray) : MFloat;
26060:   Function DotProduct29( const V : TInt64Array) : MFloat;
26061:   Function DotProduct30( const V : TObject) : MFloat;
26062:   Function Norm : MFloat';
26063:   Function Min : MFloat';
26064:   Function Max : MFloat';
26065:   Function Range( var Min, Max : MFloat) : MFloat';
26066:   Function IsZero( const CompareDelta : MFloat) : Boolean';
26067:   Function HasZero( const CompareDelta : MFloat) : Boolean';
26068:   Function HasNegative : Boolean';
26069:   Procedure Normalize';
26070:   Procedure Negate';
26071:   Procedure ValuesInvert';
26072:   Procedure ValuesSqr';
26073:   Procedure ValuesSqrt';
26074:   Function Sum : MFloat';
26075:   Function SumOfSquares : MFloat';
26076:   Procedure SumAndSquares( out Sum, SumOfSquares : MFloat);
26077:   Procedure SumAndCubes( out Sum, SumOfSquares, SumOfCubes : MFloat);
26078:   Procedure SumAndQuads( out Sum, SumOfSquares, SumOfCubes, SumOfQuads : MFloat);
26079:   Function WeightedSum( const Weights : TVector) : MFloat';
26080:   Function Mean : MFloat';
26081:   Function HarmonicMean : MFloat';
26082:   Function GeometricMean : MFloat';
26083:   Function Median : MFloat';
26084:   Function Mode : MFloat';
26085:   Function Variance : MFloat';
26086:   Function StdDev( var Mean : MFloat) : MFloat';
26087:   Function PopulationVariance : MFloat';
26088:   Function PopulationStdDev : MFloat';
26089:   Function M1 : MFloat';
26090:   Function M2 : MFloat';
26091:   Function M3 : MFloat';
26092:   Function M4 : MFloat';
26093:   Function Skew : MFloat';
26094:   Function Kurtosis : MFloat';
26095:   Function Product : MFloat';
26096:   Function Angle( const V : TVector) : MFloat';
26097: end;
26098: end;
26099:
26100: (*-----*)
26101: procedure SIRegister_TInt64ArrayClass(CL: TPSPascalCompiler);   BigDataClass
26102: begin
26103:   //with RegClassS(CL, 'AInt64Array', 'TInt64ArrayClass') do
26104:   with CL.AddClassN(CL.FindClass('AInt64Array'), 'TInt64ArrayClass') do begin
26105:     Constructor Create( const V : Int64Array);
26106:     Procedure ExchangeItems( const Idx1, Idx2 : Integer);
26107:     Function DuplicateRange( const LoIdx, HiIdx : Integer) : AArray';
26108:     Procedure Delete( const Idx : Integer; const ACount : Integer);
26109:     Procedure Insert( const Idx : Integer; const ACount : Integer);
26110:     Procedure Assign1( const V : Int64Array);
26111:     Procedure Assign2( const V : array of Int64);
26112:     Function AppendItem( const Value : Int64) : Integer';
26113:     ('Data', 'Int64Array', iptrw);
26114:     RegisterProperty('Count', 'Integer', iptrw);
26115:   end;
26116: end;
26117:
26118: procedure SIRegister_TExtendedArray(CL: TPSPascalCompiler);
26119: begin
26120:   //with RegClassS(CL, 'AExtendedArray', 'TExtendedArray') do
26121:   with CL.AddClassN(CL.FindClass('AExtendedArray'), 'TExtendedArray') do begin
26122:     Constructor Create( const V : ExtendedArray);
26123:     //RegisterMethod('Procedure Free;
26124:     Procedure Assign( const Source : TObject);
26125:     Procedure ExchangeItems( const Idx1, Idx2 : Integer);
26126:     Function DuplicateRange( const LoIdx, HiIdx : Integer) : AArray';
26127:     Procedure Delete( const Idx : Integer; const ACount : Integer);
26128:     Procedure Insert( const Idx : Integer; const ACount : Integer);
26129:     Procedure Assign2( const V : ExtendedArray);
26130:     //RegisterMethod('Procedure Assign( const V : array of Extended);
26131:     Function AppendItem( const Value : Extended) : Integer';
26132:     RegisterProperty('Data', 'ExtendedArray', iptrw);
26133:     RegisterProperty('Count', 'Integer', iptrw);
26134:   end;
26135: end;
26136:
26137: (*-----*)
26138: procedure SIRegister_neuraldatasets(CL: TPSPascalCompiler);
26139: begin
26140:   TTinyImageChannel, 'array [0..31] of array[0..31] of byte; ');
26141:   TTinyImageChannel1D, 'array [0..32 * 32 - 1] of byte; ');
26142:   TMNistImage, 'array [0..27] of array[0..27] of byte; ');
26143:   //TTinyImageChannel1D = packed array [0..32 * 32 - 1] of byte;
26144:   //TMNistImage = packed array [0..27, 0..27] of byte;
26145:   TTinyImage, 'record bLabel : byte; R : TTinyImageChannel; G : T'

```

```

26146:      + 'TinyImageChannel; B : TTinyImageChannel; end');
26147:      TCifar100Image, 'record bCoarseLabel : byte; bFineLabel : byte;
26148:      + ' R : TTinyImageChannel; G : TTinyImageChannel; B : TTinyImageChannel; end');
26149:      TTinySingleChannelImage, 'record bLabel : byte; Grey : TTinyImageChannel; end');
26150:      TTinySingleChannelImageID, 'record bLabel : byte; Grey : TTinyImageChannelID; end');
26151:      //CL.AddTypes('TTinySingleChannelImagePtr', '^TTinySingleChannelImage // will not work');
26152:      //CL.AddTypes('TTinySingleChannelImageIDPtr', '^TTinySingleChannelImageID // will not work');
26153:      SIRegister_TFileNameList(CL);
26154:      SIRegister_TClassesAndElements(CL);
26155:      Procedure CreateVolumesFromImagesFromFolder(out ImgTrainingVolumes,ImgValidationVolumes,
      ImgTestVolumes:TNNNetVolumeList;FolderName,pImageSubFolder:string;color_encoding:integer;TrainingProp,
      ValidationProp,TestProp:single;+'NewSizeX:integer;NewSizeY:integer);
26156:      Procedure CreateFileNamesListsFromImagesFromFolder( out TrainingFileNames, ValidationFileNames,
      TestFileNames: TFileNameList; FolderName, pImageSubFolder : string; TrainingProp, ValidationProp, TestProp
      : single);
26157:      //CL.AddDelphiFunction('Procedure LoadImageIntoVolume( M : TFPMemoryImage; Vol : TNNNetVolume);
26158:      //CL.AddDelphiFunction('Procedure LoadVolumeIntoImage( Vol : TNNNetVolume; M : TFPMemoryImage);
26159:      Function LoadImageFromFileIntoVolume( ImageFileName : string; V : TNNNetVolume) : boolean);
26160:      Function SaveImageFromVolumeToFile( V : TNNNetVolume; ImageFileName : string) : boolean);
26161:      Procedure ConfusionWriteCSVHeader( var CSVConfusion : TextFile; Labels : array of string);
26162:      Procedure ConfusionWriteCSV( var CSVConfusion : TextFile; Vol : TNNNetVolume; Digits : integer);
26163:      Procedure LoadTinyImageIntoNNetVolume1( var TI : TTinyImage; Vol : TNNNetVolume);
26164:      Procedure LoadTinyImageIntoNNetVolume2( var TI : TCifar100Image; Vol : TNNNetVolume);
26165:      Procedure LoadTinyImageIntoNNetVolume3( var TI : TMNISTImage; Vol : TNNNetVolume);
26166:      Procedure LoadNNetVolumeIntoTinyImage4( Vol : TNNNetVolume; var TI : TTinyImage);
26167:      Procedure LoadNNetVolumeIntoTinyImage5( Vol : TNNNetVolume; var TI : TCifar100Image);
26168:      Procedure LoadTinySingleChannelIntoNNetVolume( var SC : TTinySingleChannelImage; Vol : TNNNetVolume);
26169:      Procedure TinyImageCreateGrey( var TI : TTinyImage; var TIGrey : TTinySingleChannelImage);
26170:      Procedure TinyImageHE( var TI, TIHE : TTinySingleChannelImage);
26171:      Procedure TinyImageVE( var TI, TIVE : TTinySingleChannelImage);
26172:      Procedure TinyImageRemoveZeroGradient( var TI : TTinySingleChannelImage; distance : byte);
26173:      Procedure TinyImageHVE( var TI, TIHE : TTinySingleChannelImage);
26174:      Function TinyImageToID( var TI : TTinySingleChannelImage) : TTinySingleChannelImageID);
26175:      Procedure CreateCifar10Volumes(out ImgTrainingVolumes,ImgValidationVolumes,
      ImgTestVolumes:TNNNetVolumeList;color_encoding:byte);
26176:      Procedure CreateCifar100Volumes(out ImgTrainingVolumes,ImgValidationVolumes,
      ImgTestVolumes:TNNNetVolumeList;color_encoding:byte;Verbose:bool;
26177:      Procedure CreateMNISTVolumes( out ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes :
      TNNNetVolumeList; TrainFileName, TestFileName : string; Verbose : boolean; IsFashion : boolean);
26178:      Procedure loadCifar10Dataset6(ImgVolumes:TNNNetVolumeList;idx:integer;base_pos:
      integer;color_encoding:byte););
26179:      Procedure
      loadCifar10Dataset7(ImgVolumes:TNNNetVolumeList;fileName:string;base_pos:integer;color_encoding:byte);
26180:      Procedure loadCifar100Dataset(ImgVolumes TNNNetVolumeList;fileName: string;
      color_encoding:byte;Verbose:bool);
26181:      Procedure
      loadMNISTDataset(ImgVolumes:TNNNetVolumeList;fileName:str;Verbose:bool;IsFashion:bool;MaxLabel:int);
26182:      Function CheckCIFARFile( ) : boolean);
26183:      Function CheckCIFAR100File( ) : boolean);
26184:      Function CheckMNISTFile( fileName : string; IsFasion : boolean) : boolean);
26185:      Procedure TestBatch(NN TNNNet;ImgVolumes:TNNNetVolumeList;SampleSize:integer;out Rate,Loss,
      ErrorSum:TNeuralFloat)
26186:      Procedure TranslateCifar10VolumesToMachineAnimal( VolumeList : TNNNetVolumeList);
26187:      Function SwapEndian( I : integer) : integer);
26188:      end;
26189:
26190:      (*-----*)
26191:      procedure SIRegister_neuralnetworkCAI(CL: TPSPascalCompiler);
26192:      begin
26193:      CL.AddConstantN('csMaxInterleavedSize','integer').SetInt( 95);
26194:      CL.AddTypes('TNeuralFloatDynArr', 'array of TNeuralFloat');
26195:      CL.AddClassN(CL.FindClass('TOBJECT'),'TNNNet');
26196:      SIRegister_TNNNetNeuron(CL);
26197:      SIRegister_TNNNetNeuronList(CL);
26198:      CL.AddConstantN('csNNetMaxParameterIdx','LongInt').SetInt( 7);
26199:      SIRegister_TNNNetLayer(CL);
26200:      //CL.AddTypes('TNNNetLayerClass', 'class of TNNNetLayer');
26201:      SIRegister_TNNNetLayerConcatWeights(CL);
26202:      SIRegister_TNNNetLayerList(CL);
26203:      SIRegister_TNNNetInputBase(CL);
26204:      SIRegister_TNNNetInput(CL);
26205:      SIRegister_TNNNetIdentity(CL);
26206:      SIRegister_TNNNetPad(CL);
26207:      SIRegister_TNNNetIdentityWithoutL2(CL);
26208:      SIRegister_TNNNetIdentityWithoutBackprop(CL);
26209:      //CL.AddTypes('TNNNetActivationFunctionClass', 'class of TNNNetIdentity');
26210:      SIRegister_TNNNetReLUBase(CL);
26211:      SIRegister_TNNNetDigital(CL);
26212:      SIRegister_TNNNetReLU(CL);
26213:      SIRegister_TNNNetReLU(CL);
26214:      SIRegister_TNNNetSELU(CL);
26215:      SIRegister_TNNNetSwish(CL);
26216:      SIRegister_TNNNetReLUsqrt(CL);
26217:      SIRegister_TNNNetPower(CL);
26218:      SIRegister_TNNNetLeakyReLU(CL);
26219:      SIRegister_TNNNetVeryLeakyReLU(CL);
26220:      SIRegister_TNNNetSigmoid(CL);
26221:      SIRegister_TNNNetHyperbolicTangent(CL);
26222:      SIRegister_TNNNetMulLearning(CL);

```

```
26223: SIRegister_TNNetMulByConstant (CL);
26224: SIRegister_TNNetNegate (CL);
26225: SIRegister_TNNetAddAndDiv (CL);
26226: SIRegister_TNNetAddNoiseBase (CL);
26227: SIRegister_TNNetDropout (CL);
26228: SIRegister_TNNetRandomMulAdd (CL);
26229: SIRegister_TNNetChannelRandomMulAdd (CL);
26230: SIRegister_TNNetLayerMaxNormalization (CL);
26231: SIRegister_TNNetLayerStdNormalization (CL);
26232: SIRegister_TNNetMovingStdNormalization (CL);
26233: SIRegister_TNNetChannelTransformBase (CL);
26234: SIRegister_TNNetChannelShiftBase (CL);
26235: SIRegister_TNNetChannelBias (CL);
26236: SIRegister_TNNetChannelMul (CL);
26237: SIRegister_TNNetChannelMulByLayer (CL);
26238: SIRegister_TNNetCellMulByCell (CL);
26239: SIRegister_TNNetCellBias (CL);
26240: SIRegister_TNNetCellMul (CL);
26241: SIRegister_TNNetChannelZeroCenter (CL);
26242: SIRegister_TNNetChannelStdNormalization (CL);
26243: SIRegister_TNNetLocalResponseNorm2D (CL);
26244: SIRegister_TNNetInterleaveChannels (CL);
26245: SIRegister_TNNetLocalResponseNormDepth (CL);
26246: SIRegister_TNNetReshape (CL);
26247: SIRegister_TNNetConcatBase (CL);
26248: SIRegister_TNNetConcat (CL);
26249: SIRegister_TNNetDeepConcat (CL);
26250: SIRegister_TNNetSum (CL);
26251: SIRegister_TNNetSplitChannels (CL);
26252: SIRegister_TNNetSplitChannelEvery (CL);
26253: SIRegister_TNNetFullConnect (CL);
26254: //CL.AddTypes('TNNetFullConnectClass', 'class of TNNetFullConnect');
26255: SIRegister_TNNetFullConnectLinear (CL);
26256: SIRegister_TNNetFullConnectSigmoid (CL);
26257: SIRegister_TNNetFullConnectReLU (CL);
26258: SIRegister_TNNetFullConnectDiff (CL);
26259: SIRegister_TNNetSoftMax (CL);
26260: CL.AddClassN (CL.FindClass('TOBJECT'), 'TNNetLayerFullConnect');
26261: CL.AddClassN (CL.FindClass('TOBJECT'), 'TNNetLayerFullConnectReLU');
26262: CL.AddClassN (CL.FindClass('TOBJECT'), 'TNNetLayerSoftMax');
26263: CL.AddClassN (CL.FindClass('TOBJECT'), 'TNNetDense');
26264: CL.AddClassN (CL.FindClass('TOBJECT'), 'TNNetDenseReLU');
26265: SIRegister_TNNetConvolutionAbstract (CL);
26266: SIRegister_TNNetDepthwiseConv (CL);
26267: SIRegister_TNNetDepthwiseConvLinear (CL);
26268: SIRegister_TNNetDepthwiseConvReLU (CL);
26269: SIRegister_TNNetConvolutionBase (CL);
26270: //CL.AddTypes('TNNetConvolutionClass', 'class of TNNetConvolutionBase');
26271: SIRegister_TNNetGroupedConvolutionLinear (CL);
26272: SIRegister_TNNetGroupedConvolutionReLU (CL);
26273: SIRegister_TNNetGroupedPointwiseConvLinear (CL);
26274: //CL.AddTypes('TNNetGroupedPointwiseConvClass', 'class of TNNetGroupedPointwiseConvLinear');
26275: SIRegister_TNNetGroupedPointwiseConvReLU (CL);
26276: SIRegister_TNNetConvolution (CL);
26277: SIRegister_TNNetConvolutionSharedWeights (CL);
26278: SIRegister_TNNetConvolutionLinear (CL);
26279: SIRegister_TNNetConvolutionReLU (CL);
26280: SIRegister_TNNetPointwiseConv (CL);
26281: SIRegister_TNNetPointwiseConvLinear (CL);
26282: SIRegister_TNNetPointwiseConvReLU (CL);
26283: SIRegister_TNNetDeconvolution (CL);
26284: SIRegister_TNNetDeconvolutionReLU (CL);
26285: SIRegister_TNNetLocalConnect (CL);
26286: SIRegister_TNNetLocalProduct (CL);
26287: SIRegister_TNNetDeLocalConnect (CL);
26288: SIRegister_TNNetLocalConnectReLU (CL);
26289: SIRegister_TNNetDeLocalConnectReLU (CL);
26290: SIRegister_TNNetPoolBase (CL);
26291: SIRegister_TNNetMaxPool (CL);
26292: SIRegister_TNNetMaxPoolPortable (CL);
26293: SIRegister_TNNetMinPool (CL);
26294: SIRegister_TNNetMaxChannel (CL);
26295: SIRegister_TNNetMinChannel (CL);
26296: SIRegister_TNNetAvgPool (CL);
26297: SIRegister_TNNetAvgChannel (CL);
26298: SIRegister_TNNetDeMaxPool (CL);
26299: SIRegister_TNNetUpsample (CL);
26300: CL.AddClassN (CL.FindClass('TOBJECT'), 'TNNetDeAvgPool');
26301: SIRegister_TNNet (CL);
26302: SIRegister_THistoricalNets (CL);
26303: SIRegister_TNNetDataParallelism (CL);
26304: SIRegister_TNNetByteProcessing (CL);
26305: SIRegister_TNNetForByteProcessing (CL);
26306: SIRegister_TBytePredictionViaNNet (CL);
26307: SIRegister_TEasyBytePredictionViaNNet (CL);
26308: Procedure CompareComputing ( NN1, NN2 : TNNet);
26309: Procedure CompareNNStructure ( NN, NN2 : TNNet);
26310: Procedure TestConvolutionAPI ( );
26311: Procedure TestDataParallelism ( NN : TNNet);
```



```

26312: //CL.AddDelphiFunction('Procedure
TestConvolutionOpenCL(platform_id:cl_platform_id;device_id:cl_device_id);
26313: //CL.AddDelphiFunction('Procedure
TestFullConnectOpenCL(platform_id:cl_platform_id;device_id:cl_device_id);
26314: Procedure RebuildPatternOnPreviousPatterns( Calculated : TNNetVolume; LocalWeight : TNNetVolume;
PrevLayer : TNNetNeuronList; PrevStride : integer; ReLU : boolean; Threshold : TNeuralFloat);
26315: Procedure RebuildNeuronListOnPreviousPatterns( CalculatedLayer : TNNetNeuronList; CurrentLayer, PrevLayer
: TNNetNeuronList; PrevStride:integer;ReLU:boolean;Threshold:TNeuralFloat); InputLayer :=
26316: function NN.AddDenseNetBlockCAI70 (InnerConvNum,iConvNeuronCount,0, netconv,ChkUseSeparableConv.Checked,
ChkMovingNorm.Checked,nil,nil, 0,1,1.0,1,1,1);
26317: end;
26318:
26319: (*-----*)
26320: procedure SIRegister_neuralfit(CL: TPSPascalCompiler);
26321: begin
26322: TCustomLearningRateScheduleObjFn', 'Function ( Epoch : integer) : single');
26323: TMultiThreadProcItem', 'TObject');
26324: SIRegister_TNeuralFitBase(CL);
26325: SIRegister_TNeuralFitWithImageBase(CL);
26326: TNNetDataAugmentationFn', 'Procedure ( pInput : TNNetVolume; ThreadId : integer);
26327: TNNetLossFn', 'Function ( ExpectedOutput, FoundOutput : TNNetVolume; ThreadId : integer) :
TNeuralFloat');
26328: TNNetGetPairFn', 'Function ( Idx : integer; ThreadId : integer) : TNNetVolumePair');
26329: TNNetGet2VolumesProc', 'Procedure ( Idx : integer; ThreadId : integer; pInput, pOutput : TNNetVolume);
26330: SIRegister_TNeuralDataLoadingFit(CL);
26331: SIRegister_TNeuralFit(CL);
26332: SIRegister_TNeuralImageLoadingFit(CL);
26333: SIRegister_TNeuralImageFit(CL);
26334: Function MonopolarCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26335: Function BipolarCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26336: Function BipolarCompare99( A, B : TNNetVolume; ThreadId : integer) : boolean');
26337: Function ClassCompare( A, B : TNNetVolume; ThreadId : integer) : boolean');
26338: end;
26339: V 47620
26340: FFit.FitLoading2(FDiscriminator, 64*10, 500, 500, 64, 35000,
26341: @GetDiscriminatorTrainingPair, nil, nil); // This
26342: FFit.FitLoading22(FDiscriminator, 64*10, 500, 500, 64, 35000,
26343: @GetDiscriminatorTrainingProc, nil, nil); //}
26344:
26345: procedure SIRegister_neuralgeneric(CL: TPSPascalCompiler);
26346: begin
26347: SIRegister_TIncDec(CL);
26348: SIRegister_TRandom(CL);
26349: Function MaxSingle2( x, y : single) : single');
26350: Function GetMaxDivisor( x, acceptableMax : integer) : integer');
26351: Function GetMaxAcceptableCommonDivisor( a, b : integer; max_acceptable : integer) : integer');
26352: end;
26353:
26354: (*-----*)
26355: procedure SIRegister_neuraldatasetsv(CL: TPSPascalCompiler);
26356: begin
26357: TImageDynArr', 'array of TImage');
26358: TLabelDynArr', 'array of TLabel');
26359: Procedure LoadTinyImageIntoTImage( var TI : TTinyImage; var Image : TImage);
26360: Procedure LoadTISingleChannelIntoImage( var TI : TTinySingleChannelImage; var Image : TImage);
26361: Procedure ShowNeurons(pNeuronList:TNNetNeuronList;var pImage:TImageDynArr;startImage,filterSize,
color_encoding:int;ScalePerImage:bool;
26362: Procedure CreateAscentImages(GrBoxNeurons: TGroupBox; var pImage: TImageDynArr; var pLabelX, pLabelY:
TLabelDynArr; ImagesNum : integer; inputSize, displaySize, imagesPerRow : integer);
26363: Procedure CreateNeuronImages(GrBoxNeurons: TGroupBox; var pImage: TImageDynArr; var pLabelX, pLabelY:
TLabelDynArr; pNeuronList : TNNetNeuronList; filterSize, imagesPerRow, NeuronNum : integer);
26364: Procedure FreeNeuronImages( var pImage : TImageDynArr; var pLabelX, pLabelY : TLabelDynArr);
26365: Procedure LoadNNLayersIntoCombo( NN : TNNet; Combo : TComboBox);
26366: end;
26367:
26368: (*-----*)
26369: procedure SIRegister_flcFloats(CL: TPSPascalCompiler);
26370: begin
26371: Function flcDoubleMin( const A, B : Double) : Double');
26372: Function flcDoubleMax( const A, B : Double) : Double');
26373: Function flcExtendedMin( const A, B : Extended) : Extended');
26374: Function flcExtendedMax( const A, B : Extended) : Extended');
26375: Function flcFloatMin( const A, B : Float) : Float');
26376: Function flcFloatMax( const A, B : Float) : Float');
26377: Function flcFloatClip( const Value : Float; const Low, High : Float) : Float');
26378: Function flcInSingleRange( const A : Float) : Boolean');
26379: Function flcInDoubleRange( const A : Float) : Boolean');
26380: Function flcInCurrencyRange0( const A : Float) : Boolean;
26381: Function flcInCurrencyRange1( const A : Int64) : Boolean;
26382: Function flcExtendedExponentBase2( const A : Extended; var Exponent : Integer) : Boolean');
26383: Function flcExtendedExponentBase10( const A : Extended; var Exponent : Integer) : Boolean');
26384: Function flcExtendedIsInfinity( const A : Extended) : Boolean');
26385: Function flcExtendedIsNaN( const A : Extended) : Boolean');
26386: CL.AddConstantN('flcSingleCompareDelta','Extended').setExtended( 1.0E-34);
26387: flcDoubleCompareDelta','Extended').setExtended( 1.0E-280);
26388: ExtendedCompareDelta','').SetString( DoubleCompareDelta);
26389: flcExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
26390: DefaultCompareDelta','').SetString( SingleCompareDelta);
26391: Function flcFloatZero( const A : Float; const CompareDelta : Float) : Boolean');

```

```

26392: Function flcFloatOne( const A : Float; const CompareDelta : Float) : Boolean'';
26393: Function flcFloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean'';
26394: Function flcFloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult'';
26395: flcSingleCompareEpsilon', 'Extended').setExtended( 1.0E-5);
26396: flcDoubleCompareEpsilon', 'Extended').setExtended( 1.0E-13);
26397: flcExtendedCompareEpsilon', 'Extended').setExtended( 1.0E-17);
26398: flcDefaultCompareEpsilon', 'Extended').setExtended( 1.0E-10);
26399: Function flcExtendedApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean'';
26400: Function flcExtendedApproxCompare( const A, B : Extended; const CompareEpsilon : Double) : TCompareResult'';
26401: Function flcDoubleApproxEqual( const A, B : Double; const CompareEpsilon : Double) : Boolean'';
26402: Function flcDoubleApproxCompare( const A, B : Double; const CompareEpsilon : Double) : TCompareResult'';
26403: Function flcFloatApproxEqual( const A, B : Float; const CompareEpsilon : Float) : Boolean'';
26404: Function flcFloatApproxCompare( const A, B : Float; const CompareEpsilon : Float) : TCompareResult'';
26405: Function flcFloatToStringA( const A : Float) : AnsiString'';
26406: Function flcFloatToStringB( const A : Float) : RawByteString'';
26407: Function flcFloatToStringU( const A : Float) : UnicodeString'';
26408: Function flcFloatToString( const A : Float) : String'';
26409: Function flcTryStringToFloatB( const A : RawByteString; out B : Float) : Boolean'';
26410: Function flcTryStringToFloatU( const A : UnicodeString; out B : Float) : Boolean'';
26411: Function flcTryStringToFloat( const A : String; out B : Float) : Boolean'';
26412: Function flcStringToFloatB( const A : RawByteString) : Float'';
26413: Function flcStringToFloatU( const A : UnicodeString) : Float'';
26414: Function flcStringToFloat( const A : String) : Float'';
26415: Function flcStringToFloatDefB( const A : RawByteString; const Default : Float) : Float'';
26416: Function flcStringToFloatDefU( const A : UnicodeString; const Default : Float) : Float'';
26417: Function flcStringToFloatDef( const A : String; const Default : Float) : Float'';
26418: Procedure FLCFloatStringTest'';
26419: Procedure FLCFloatTest'';
26420: end;
26421:
26422: procedure SIRegister_neuralthread(CL: TPSPascalCompiler);
26423: begin
26424:   CL.AddTypeS('TNeuralProc', 'Procedure ( index, threadnum : integer);
26425:   SIRegister_TNeuralThread(CL);
26426:   SIRegister_TNeuralThreadList(CL);
26427:   Procedure NeuralThreadListCreate( pSize : integer);
26428:   Procedure NeuralThreadListFree( );
26429:   Function NNTL : TNeuralThreadList'';
26430:   Procedure CreateNeuralThreadListIfRequired( );
26431:   Function NeuralDefaultThreadCount : integer'';
26432:   Procedure NeuralInitCriticalSection( var pCritSec : TRTLCriticalSection);
26433:   Procedure NeuralDoneCriticalSection( var pCritSec : TRTLCriticalSection);
26434:   Procedure InitCriticalSection( var pCritSec : TRTLCriticalSection);
26435:   Procedure DoneCriticalSection( var pCritSec : TRTLCriticalSection);
26436:   Function neuralGetProcessId( ) : integer'';
26437: end;
26438:
26439: procedure SIRegister_uSysTools(CL: TPSPascalCompiler);
26440: begin
26441:   Function
26442:   usGetRegExFileList (APath:String;APattern:String;ASubDirs:Boolean;AResultList:TStrings;AAttributes:Integer;AWithPath:Boolean;
26443:   Procedure usFindFileInPaths( const AFileName:String; const ASearchPaths:String; AMinimumFileSize:Int64; const
26444:   AResultList:TStrings);
26445:   Procedure usPrintWindow( Wnd : HWND; ATo : TBitmap);
26446:   Function usGetWindowByClassTree( const AClassTree: TStringList; AParent:HWND;
26447:   AWithWildCards:Boolean):HWND'';
26448:   Function usGetTempFile( AExtension : String) : String'';
26449:   Function usDeleteFileToWasteBin( AFileName : string) : boolean'';
26450:   Function usExpandEnvVars( AInputString : String) : String'';
26451:   Function usFileSize( AFileName : String) : Int64'';
26452:   Function usIsMultiprocessor : Boolean'';
26453:   Function usProcessorCount : Cardinal'';
26454:   Function usIsAdmin: Boolean'';
26455:   Function usMultipleStringReplace (AString:Str;AOldPatterns,ANewPatterns:array of
26456:   Str;AFlags:TReplaceFlags):Str;
26457:   Function usMakeFileName( ADesiredFName : String) : String'';
26458:   Function usIsLike( AString, APattern : String) : Boolean'';
26459:   Function usVarRecToVariant( AValue : TVarRec) : Variant'';
26460:   Function usVariantToTypedVarRec( const Item : Variant; VarType : TVarType) : TVarRec'';
26461:   Function usVariantToVarRec( const Item : Variant) : TVarRec'';
26462:   Procedure usFinalizeVarRec( var Item : TVarRec);
26463:   Function usGetWindowText( wnd : HWND) : WideString'';
26464:   Procedure usSetWindowText( wnd : HWND; txt : WideString);
26465:   Function usMultipleWideStringReplace (AString:WideString;AOldPatterns,ANewPatterns:array of WideString;
26466:   AFlags: TReplaceFlags):WideString'';
26467:   Procedure usAcceptNumericOnly( var Key : Char; Comma : Boolean);
26468:   Function usSameGUID( AGUID1, AGUID2 : TGUID) : Boolean'';
26469:   Function usIsBitSet( AValue : Integer; ABitIndex : Byte) : Boolean'';
26470:   Function usClientToScreen( AWindow : HWND; var APoint : TPoint) : Boolean'';
26471:   Function usGetDataObjectFromFileList( const Directory : string; Files : TStrings) : IDataObject'';
26472:   Function usGetDataObjectFromFile( const Directory : string; AFile : String) : IDataObject'';
26473:   Function usGetFileListFromDataObject( const DataObject : IDataObject; Files : TStrings) : Integer'';
26474:   Function usKeyPressed( AKey : Smallint) : Boolean;
26475:   Function usKeyPressed1( AKey : Byte) : Boolean;
26476:   Function usKeyToogled( AKey : Smallint) : Boolean;
26477:   Function usKeyToogled1( AKey : Byte) : Boolean;
26478:   Function usGetShellFolder(CSIDL : integer) : string'';
26479: end;
26480:

```

```

26477: function SerializeStream(const AStream : TStream) : String;
26478: procedure DeserializeToStream(const Data : String; const AStream : TStream);
26479: procedure ReplaceImageInImageList(FromImageList: TImageList; FromIndex: Integer;
26480: ToImageList: TImageList; ToIndex: Integer);
26481: Function GetSIDStringFromUser( AUsername : AnsiString; AServerName : AnsiString) : String'';
26482:
26483: TestConvolutionAPI(); procedure TestBackProp(); <!-- 1.5 GByte MemoryTest
26484: https://github.com/ariaghora/noe
26485: https://github.com/Fr0sT-Brutal/awesome-pascal#machine-learning
26486: https://github.com/TheUnknownOnes/theunknownones/tree/master/Libraries/PascalScriptAddons
26487:
26488: procedure SIRegister_URungeKutta4(CL: TPSPascalCompiler);
26489: begin
26490: CL.AddConstantN('maxfuncs2','LongInt').SetInt( 10);
26491: //CL.AddTypeS('float', 'extended');
26492: CL.AddTypeS('TUserFunction2', 'Function ( T, X, XPrime : double) : double');
26493: TUserCallbackFunction2', 'Function ( T, X, XPrime : double) : Boolean');
26494: TNDData2', 'record X : double; xPrime : double; end');
26495: TNVector2', 'array[0..maxfuncs2] of TNDData2;
26496: // TFuncVect = array[1..MaxFuncs] of TUserFunctionV;
26497: // TNVector = array[0..maxfuncs] of TNDData;
26498: TUserFunctionV2', 'Function ( V : TNVector2) : double');
26499: TFuncVect2', 'array[1..MaxFuncs2] of TUserFunctionV2;
26500: TUserCallbackFunctionV2', 'Function (V:TNVector2) : boolean');
26501: Procedure
RungeKutta2ndOrderIC2(LowerLimit:double;UpperLimit:double;InitialValue:double;InitialDeriv:double;
ReturnInterval:double;CalcInterval:double;var Error:byte;UserFunc:TUserFunctionV2;UserCallBack:
TUserCallbackFunctionV2);
26502: Procedure RungeKutta2ndOrderIC_System2(LowerLimit:double; UpperLimit: double; InitialValues : TNVector2;
ReturnInterval: double; CalcInterval: double; var Error: byte; NumEquations: Integer; Vector: TFuncVect2;
UserCallBack: TUserCallbackFunctionV2);
26503: end;
26504:
26505: procedure SIRegister_OverbyteIcsUtils(CL: TPSPascalCompiler);
26506: begin
26507: CL.AddTypeS('TicsDbcsLeadBytes', 'TSysCharset');
26508: 'MB_ERR_INVALID_CHARS','LongWord').SetUInt( $00000008);
26509: 'WC_ERR_INVALID_CHARS','LongWord').SetUInt( $80);
26510: CP_UTF16','LongInt').SetInt( 1200);
26511: CP_UTF16Be','LongInt').SetInt( 1201);
26512: CP_UTF32','LongInt').SetInt( 12000);
26513: CP_UTF32Be','LongInt').SetInt( 12001);
26514: TOBJECT','EICSStringConvertError');
26515: TCharsetDetectResult', '(cdrAscii,cdrUtf8,cdrUnknown );
26516: 'TicsNormForm','(icsNormalizationOther,icsNormalizationC,icsNormalizationD,icsNormalizationKD );
26517: TicsSearchRecW','record Time:Integer;Size:Integer;Attr: '
Integer;Name:String;ExcludeAttr:Integer;FindHandle:THandle;FindData:TWin32FindData;end');
26519: TUnicode_String','record Length:Word;MaximumLength:Word;Buffer:WideChar;end');
26520: TThreadId', 'LongWord;
26521: // CL.AddTypeS('PUnicode_String', '^TUnicode_String // will not work');
26522: SIRegister_TicsFileStreamW(CL);
26523: CL.AddConstantN('ICONV_UNICODE','String').SetString( 'UTF-16LE');
26524: Function IcsIconvNameFromCodePage( CodePage : LongWord) : AnsiString';
26525: Function IcsIsValidAnsiCodePage( const CP : LongWord) : Boolean';
26526: Procedure IcsCharLowerA( var ACh : AnsiChar);
26527: Function IcsGetCurrentThreadId : TThreadId');
26528: Function IcsGetFreeDiskSpace( const APath : String) : Int64');
26529: Function IcsGetLocalTimeZoneBias : LongInt');
26530: Function IcsDateTimeToUTC( dtDT : TDateTime): TDateTime');
26531: Function IcsUTCToDateTime( dtDT : TDateTime): TDateTime');
26532: Function IcsGetTickCount : LongWord');
26533: //AddDelphiFunction('Function IcsWcToMb(CodePage:LongWord; Flags:Cardinal;
WStr:PWideChar;WStrLen:Integer; MbStr:PAnsiChar;MbStrLen:Integer;DefaultChar:PAnsiChar;UsedDefaultChar:PLon
WStr:PWideChar;WStrLen:Int):Int');
26535: Function IcsGetDefaultWindowsUnicodeChar( CodePage : LongWord) : WideChar');
26536: Function IcsGetDefaultWindowsAnsiChar( CodePage : LongWord) : AnsiChar');
26537: Procedure IcsGetAcp( var CodePage : LongWord);
26538: Function IcsIsDBCSCodePage( CodePage : LongWord) : Boolean');
26539: Function IcsIsDBCSLeadByte( Ch : AnsiChar; CodePage : LongWord) : Boolean');
26540: Function IcsIsMBCSCodePage( CodePage : LongWord) : Boolean');
26541: Function IcsIsSBCSCodePage( CodePage : LongWord) : Boolean');
26542: Function IcsGetLeadBytes( CodePage : LongWord) : TICSDbcsLeadBytes');
26543: Function icsUnicodeToUsAscii4( const Str: UnicodeString; FailCh : AnsiChar) : AnsiString;
26544: Function icsUnicodeToUsAscii5( const Str: UnicodeString) : AnsiString;
26545: Function icsUsAsciiToUnicode6( const Str: RawByteString; FailCh : AnsiChar) : UnicodeString;
26546: Function icsUsAsciiToUnicode7( const Str: RawByteString) : UnicodeString;
26547: Function icsUnicodeToAnsi8(const Str:WideChar;ACodePage LongWord; SetCodePage : Boolean) : RawByteString;
26548: Function icsUnicodeToAnsi9(const Str:UnicodeString;ACodePage:LongWord;SetCodePage:Boolean):RawByteString;
26549: Function icsUnicodeToAnsi10( const Str: UnicodeString) : RawByteString;
26550: Function icsAnsiToUnicode11( const Str: AnsiChar; ACodePage : LongWord) : UnicodeString;
26551: Function icsAnsiToUnicode12( const Str: RawByteString; ACodePage : LongWord) : UnicodeString;
26552: CL.AddDelphiFunction('Function icsAnsiToUnicode13( const Str : RawByteString) : UnicodeString;
26553: Function IcsBufferToUnicode14(const Buffer,BufferSize:Integer;BufferCodePage:LongWord;out FailedByteCount
: Integer)UnicodeString;
26554: Function IcsBufferToUnicode15(const Buffer,
BufferSize:Integer;BufferCodePage:LongWord;RaiseFailedBytes:Bool): UnicodeString;
26555: Function IcsGetWideCharCount(const Buffer,BufferSize:Integer;BufferCodePage:LongWord;out
InvalidEndByteCount: Integer):Integer;

```

```

26556: Function IcsGetWideChars(const Buffer,
BufferSize:Integer;BufferCodePage:LongWord;Chars:WideChar;CharCount: Integer):Integer;
26557: Function
icsStreamWriteString16(AStream:TStream;Str:WideChar;cLen:Int;ACodePage:LongWord;WriteBOM:Bool):Int;
26558: Function icsStreamWriteString17(AStream:TStream;Str:WideChar;cLen:Integer;ACodePage:LongWord):
Integer;
26559: Function icsStreamWriteString18(AStream:TStream;const Str:UnicodeString;ACodePage
LongWord;WriteBOM:Bool):Int;
26560: Function icsStreamWriteString19(AStream:TStream;const Str:UnicodeString;ACodePage:LongWord):Integer;
26561: Function icsStreamWriteString20(AStream:TStream;const Str:UnicodeString):Integer;
26562: Function icsIsUsAscii21(const Str:RawByteString):Boolean;
26563: Function icsIsUsAscii22(const Str:UnicodeString):Boolean;
26564: Procedure IcsAppendStr(var Dest:RawByteString;const Src:RawByteString);
26565: Function icsatoi23(const Str:RawByteString):Integer;
26566: Function icsatoi24(const Str:UnicodeString):Integer;
26567: Function icsatoi6425(const Str:RawByteString):Int64;
26568: Function icsatoi6426(const Str:UnicodeString):Int64;
26569: Function IcsCalcTickDiff(const StartTick, EndTick:LongWord):LongWord;
26570: Function icsStringToUtf827(const Str:UnicodeString):RawByteString;
26571: Function icsStringToUtf828(const Str:RawByteString;ACodePage:LongWord):RawByteString;
26572: Function icsUtf8ToStringW(const Str:RawByteString):UnicodeString;
26573: Function icsUtf8ToStringA(const Str:RawByteString;ACodePage:LongWord):AnsiString;
26574: Function icsCheckUnicodeToAnsi(const Str:UnicodeString;ACodePage:LongWord):Boolean;
26575: Function icsIsUtf8TrailByte(const B:Byte):Boolean;
26576: Function icsIsUtf8LeadByte(const B:Byte):Boolean;
26577: Function IcsUtf8Size(const LeadByte:Byte):Integer;
26578: Function icsIsLeadChar(ch:WideChar):Boolean;
26579: Function icsIsUtf8Valid29(const Str:RawByteString):Boolean;
26580: //Function IsUtf8Valid30(const Buf:Pointer;Len:Integer):Boolean;
26581: //Function CharSetDetect31(const Buf:Pointer;Len:Integer):TCharSetDetectResult;
26582: Function icsCharSetDetect32(const Str:RawByteString):TCharSetDetectResult;
26583: Function IcsCharNextUtf8(const Str:AnsiChar):AnsiChar;
26584: Function IcsCharPrevUtf8(const Start, Current:AnsiChar):AnsiChar;
26585: Function icsConvertCodepage(const
Str:RawByteString;SrcCodePage:LongWord;DstCodePage:LongWord):RawByteString;
26586: Function icshtoin33(Value:WideChar;Len:Integer):Integer;
26587: Function icshtoin34(Value:AnsiChar;Len:Integer):Integer;
26588: Function icshtoi235(value:WideChar):Integer;
26589: Function icshtoi236(value:AnsiChar):Integer;
26590: Function IcsBufferToHex37(const Buf, Size:Integer):String;
26591: Function IcsBufferToHex38(const Buf, Size:Integer;Separator:Char):String;
26592: Function icsIsXDigit39(Ch:WideChar):Boolean;
26593: Function icsIsXDigit40(Ch:AnsiChar):Boolean;
26594: Function icsXDigit41(Ch:WideChar):Integer;
26595: Function icsXDigit42(Ch:AnsiChar):Integer;
26596: Function icsIsCharInSysCharSet43(Ch:WideChar;const ASet:TSysCharSet):Boolean;
26597: Function icsIsCharInSysCharSet44(Ch:AnsiChar;const ASet:TSysCharSet):Boolean;
26598: Function icsIsDigit45(Ch:WideChar):Boolean;
26599: Function icsIsDigit46(Ch:AnsiChar):Boolean;
26600: Function icsIsSpace47(Ch:WideChar):Boolean;
26601: Function icsIsSpace48(Ch:AnsiChar):Boolean;
26602: Function icsIsCRLF49(Ch:WideChar):Boolean;
26603: Function icsIsCRLF50(Ch:AnsiChar):Boolean;
26604: Function icsIsSpaceOrCRLF51(Ch:WideChar):Boolean;
26605: Function icsIsSpaceOrCRLF52(Ch:AnsiChar):Boolean;
26606: Function icsXDigit2(S:PChar):Integer;
26607: Function icsstpbk53(PValue:WideChar):WideChar;
26608: Function icsstpbk54(PValue:AnsiChar):AnsiChar;
26609: Function icsStrNextChar(const Str:AnsiChar;ACodePage:LongWord):AnsiChar;
26610: Function icsStrPrevChar(const Start, Current:AnsiChar;ACodePage:LongWord):AnsiChar;
26611: Function icsStrCharLength55(const Str:AnsiChar;ACodePage:LongWord):Integer;
26612: Function icsNextCharIndex56(const S:RawByteString;Index:Integer;ACodePage:LongWord):Integer;
26613: Function IcsGetBomBytes(ACodePage:LongWord):TBytes;
26614: Function IcsGetBufferCodepage57(Buf:AnsiChar;ByteCount:Integer):LongWord;
26615: Function IcsGetBufferCodepage58(Buf:AnsiChar;ByteCount:Integer;out BOMSize:Integer):LongWord;
26616: Function IcsSwap16(Value:Word):Word;
26617: Procedure IcsSwap16Buf(Src, Dst:Word;WordCount:Integer);
26618: Function IcsSwap32(Value:LongWord):LongWord;
26619: Procedure IcsSwap32Buf(Src, Dst:LongWord;LongWordCount:Integer);
26620: Function IcsSwap64(Value:Int64):Int64;
26621: Procedure IcsSwap64Buf(Src, Dst:Int64;QuadWordCount:Integer);
26622: Procedure IcsNameThreadForDebugging(AThreadName:AnsiString;AThreadID:TThreadID);
26623: Function IcsNormalizeString(const S:UnicodeString;NormForm:TIcsNormForm):UnicodeString;
26624: Function IcsCryptGenRandom(var Buf, BufSize:Integer):Boolean;
26625: Function IcsRandomInt(const ARange:Integer):Integer;
26626: Function IcsFileUtcModified(const FileName:String):TDateTime;
26627: //Function IcsInterlockedCompareExchange(var Destination:Pointer;Exchange:Pointer;Comperand
Pointer):Pointer;
26628: Function IcsExtractFilePathW(const FileName:UnicodeString):UnicodeString;
26629: Function IcsExtractFileDirW(const FileName:UnicodeString):UnicodeString;
26630: Function IcsExtractFileDriveW(const FileName:UnicodeString):UnicodeString;
26631: Function IcsExtractFileNameW(const FileName:UnicodeString):UnicodeString;
26632: Function IcsExtractFileExtW(const FileName:UnicodeString):UnicodeString;
26633: Function IcsExpandFileNameW(const FileName:UnicodeString):UnicodeString;
26634: Function IcsExtractNameOnlyW(FileName:UnicodeString):UnicodeString;
26635: Function IcsChangeFileExtW(const FileName, Extension:UnicodeString):UnicodeString;
26636: Function IcsStrAllocW(Len:Cardinal):WideChar;
26637: Function IcsStrLenW(Str:WideChar):Cardinal;
26638: Function IcsAnsiCompareFileNameW59(const S1, S2:UnicodeString):Integer;

```



```

26639: Function IcsAnsiCompareFileNameW60( const Utf8S1, Utf8S2 : UTF8String) : Integer;
26640: Function IcsDirExistsW61( const FileName : WideChar) : Boolean;
26641: Function IcsDirExistsW62( const FileName : UnicodeString) : Boolean;
26642: Function IcsDirExistsW63( const Utf8FileName : UTF8String) : Boolean;
26643: Function IcsFindFirstW64( const Path : UnicodeString; Attr : Integer; var F: TicsSearchRecW):Integer;
26644: Function IcsFindFirstW65( const Utf8Path : UTF8String; Attr : Integer;var F: TicsSearchRecW):Integer;
26645: Procedure IcsFindCloseW( var F : TicsSearchRecW);
26646: Function IcsFindNextW( var F : TicsSearchRecW) : Integer;
26647: Function IcsIncludeTrailingPathDelimiterW( const S : UnicodeString) : UnicodeString;
26648: Function IcsExcludeTrailingPathDelimiterW( const S : UnicodeString) : UnicodeString;
26649: Function IcsExtractLastDir66( const Path : RawByteString) : RawByteString;
26650: Function IcsExtractLastDir67( const Path : UnicodeString) : UnicodeString;
26651: Function IcsFileGetAttrW68( const FileName : UnicodeString) : Integer;
26652: Function IcsFileGetAttrW69( const Utf8FileName : UTF8String) : Integer;
26653: Function IcsFileSetAttrW70( const FileName : UnicodeString; Attr : Integer) : Integer;
26654: Function IcsFileSetAttrW71( const Utf8FileName : UTF8String; Attr : Integer) : Integer;
26655: Function IcsDeleteFileW72( const FileName : UnicodeString) : Boolean;
26656: Function IcsDeleteFileW73( const Utf8FileName : UTF8String) : Boolean;
26657: Function IcsRenameFileW74( const OldName, NewName : UnicodeString) : Boolean;
26658: Function IcsRenameFileW75( const Utf8OldName, Utf8NewName : UTF8String) : Boolean;
26659: Function IcsForceDirectoriesW76( Dir : UnicodeString) : Boolean;
26660: Function IcsForceDirectoriesW77( Utf8Dir : UTF8String) : Boolean;
26661: Function IcsCreateDirW78( const Dir : UnicodeString) : Boolean;
26662: Function IcsCreateDirW79( const Utf8Dir : UTF8String) : Boolean;
26663: Function IcsRemoveDirW80( const Dir : UnicodeString) : Boolean;
26664: Function IcsRemoveDirW81( const Utf8Dir : UTF8String) : Boolean;
26665: Function IcsFileAgeW82( const FileName : UnicodeString) : Integer;
26666: Function IcsFileAgeW83( const Utf8FileName : UTF8String) : Integer;
26667: Function IcsFileExistsW84( const FileName : UnicodeString) : Boolean;
26668: Function IcsFileExistsW85( const Utf8FileName : UTF8String) : Boolean;
26669: Function IcsAnsiLowerCaseW( const S : UnicodeString) : UnicodeString;
26670: Function IcsAnsiUpperCaseW( const S : UnicodeString) : UnicodeString;
26671: Function IcsMakeWord( L, H : Byte) : Word;
26672: Function IcsMakeLong( L, H : Word) : Longint;
26673: Function IcsHiWord( LW : LongWord) : Word;
26674: Function IcsHiByte( W : Word) : Byte;
26675: Function IcsLoByte( W : Word) : Byte;
26676: Function IcsLoWord( LW : LongWord) : Word;
26677: Procedure IcsCheckOSError( ALastError : Integer);
26678: Function IcsIntToStrA( N : Integer) : AnsiString;
26679: Function IcsIntToHexA( N : Integer; Digits : Byte) : AnsiString;
26680: Function IcsTrim86( const Str : AnsiString) : AnsiString;
26681: Function IcsTrim87( const Str : UnicodeString) : UnicodeString;
26682: Function IcsLowerCase88( const S : AnsiString) : AnsiString;
26683: Function IcsLowerCase89( const S : UnicodeString) : UnicodeString;
26684: Function IcsUpperCase90( const S : AnsiString) : AnsiString;
26685: Function IcsUpperCase91( const S : UnicodeString) : UnicodeString;
26686: Function IcsUpperCaseA( const S : AnsiString) : AnsiString;
26687: Function IcsLowerCaseA( const S : AnsiString) : AnsiString;
26688: Function IcsCompareTextA( const S1, S2 : AnsiString) : Integer;
26689: Function IcsTrimA( const Str : AnsiString) : AnsiString;
26690: Function IcsSameTextA( const S1, S2 : AnsiString) : Boolean;
26691: Function IcsCompareStr92( const S1, S2 : AnsiString) : Integer;
26692: Function IcsCompareStr93( const S1, S2 : UnicodeString) : Integer;
26693: Function IcsCompareText94( const S1, S2 : AnsiString) : Integer;
26694: Function IcsCompareText95( const S1, S2 : UnicodeString) : Integer;
26695: Function IcsStrLen96( const Str : AnsiChar) : Cardinal;
26696: Function IcsStrLen97( const Str : WideChar) : Cardinal;
26697: Function IcsStrPas98( const Str : AnsiChar) : AnsiString;
26698: Function IcsStrPas99( const Str : WideChar) : string;
26699: Function IcsStrCopy100( Dest : AnsiChar; const Source : AnsiChar) : AnsiChar;
26700: Function IcsStrCopy101( Dest : WideChar; const Source : WideChar) : WideChar;
26701: Function IcsStrPCopy102( Dest : PChar; const Source : string) : PChar;
26702: Function IcsStrPCopy103( Dest : AnsiChar; const Source : AnsiString) : AnsiChar;
26703: Function IcsStrPLCopy104( Dest : PChar; const Source : string; MaxLen : Cardinal) : PChar;
26704: Function IcsStrPLCopy105( Dest : AnsiChar; const Source : AnsiString; MaxLen : Cardinal) : AnsiChar;
26705: Function
IcsStrCompOrdinalW(Str1:WideChar;Str1Length:Int;Str2:WideChar;Str2Length:Int;IgnoreCase:Bool):Int;
26706: //Func
RtlCompareUnicodeString(String1:PUNICODE_STRING;String2:PUNICODE_STRING;CaseInsensitive:BOOL):LongInt');
26707: Function icsIsDebuggerPresent : BOOL;
26708: SIRegister _TicsIntegerList(CL);
26709: SIRegister _TicsCriticalSection(CL);
26710: end;
26711:
26712: procedure SIRegister_TCustomApplication(CL: TPSPascalCompiler);
26713: begin
26714: //with RegClassS(CL,'TComponent', 'TCustomApplication') do
26715: with CL.AddClassN(CL.FindClass('TComponent'),'TCustomApplication') do begin
26716: Constructor Create( AOwner : TComponent);
26717: Procedure HandleException( Sender : TObject);
26718: Procedure Initialize';
26719: Procedure Run';
26720: Procedure ShowException( E : Exception);
26721: Procedure Terminate';
26722: Procedure Terminate1( AExitCode : Integer);
26723: Function FindOptionIndex( const S : string; var Longopt : Boolean; StartAt : Integer) : Integer;
26724: Function GetOptionValue( const S : string) : string;
26725: Function GetOptionValue1( const C : Char; const S : string) : string;

```



```

26726:   Function GetOptionValues( const C : Char; const S : String) : TStringDynArray');
26727:   Function HasOption( const S : String) : Boolean');
26728:   Function HasOption1( const C : Char; const S : String) : Boolean');
26729:   Function CheckOptions(const ShortOptions:String;const Longopts:TStrings;Opts,
NonOpts:TStrings;AllErrors: Bool):String;
26730:   Function CheckOptions1(const ShortOptions:String;const Longopts:array of string;Opts,NonOpts:TStrings;
AllErrors:Bool):String;
26731:   Function CheckOptions2(const ShortOptions:String;const Longopts:TStrings; AllErrors:Boolean):String');
26732:   Function CheckOptions3(const ShortOptions:String;const LongOpts:array of
string;AllErrors:Boolean):String;
26733:   Function CheckOptions4( const ShortOptions:String;const LongOpts:String; AllErrors:Boolean):String');
26734:   Function GetNonOptions(const ShortOptions: String; const Longopts:array of string) : TStringDynArray');
26735:   Procedure GetNonOptions1(const ShortOptions:String;const Longopts:array of string;NonOptions:TStrings);
26736:   Procedure GetEnvironmentList( List : TStrings; NamesOnly : Boolean);
26737:   Procedure GetEnvironmentList1( List : TStrings);
26738:   Procedure Log(EventType : TEventType; const Msg : String);
26739:   Procedure Log1(EventType : TEventType; const Fmt : String; const Args : array of string);
26740:   ExeName', 'string', iptr);
26741:   Terminated', 'Boolean', iptr);
26742:   Title', 'string', iptrw);
26743:   OnException', 'TExceptionEvent', iptrw);
26744:   ConsoleApplication', 'Boolean', iptr);
26745:   Location', 'String', iptr);
26746:   Params', 'String integer', iptr);
26747:   ParamCount', 'Integer', iptr);
26748:   EnvironmentVariable', 'String String', iptr);
26749:   OptionChar', 'Char', iptrw);
26750:   CaseSensitiveOptions', 'Boolean', iptrw);
26751:   StopOnException', 'Boolean', iptrw);
26752:   ExceptionExitCode', 'Longint', iptrw);
26753:   ExceptObject', 'Exception', iptrw);
26754:   ExceptObjectJS', 'JSValue', iptrw);
26755:   EventLogFilter', 'TEventLogTypes', iptrw);
26756: end;
26757: end;
26758:
26759:
26760: ADD 4.7.6.10 III
26761: CL.AddDelphiFunction('function list_modules(aexename: string): string;
26762: CL.AddDelphiFunction('function list_units(aexename: string): string;
26763: check //ListView_SetItemState(H, I, Data, LVIS_SELECTED); commctrl
26764:
26765: procedure GetResourceHeader(var ResourceHeader: String; const FormName: String;
26766: const FileSize: Integer);
26767: GetResourceFormFile(List: TStrings; const FormName, FileName: String) : Boolean;
26768: function GetImageBase(const FileName: TFileName): DWORD;
26769: procedure ListViewSelectAll(ListView: TListView; Deselect: Boolean);
26770: function ListView_SetItemState(hwndLV: HWND; i: Integer; data, mask: UINT): Bool;
26771: function ListView_GetItemState(hwndLV: HWND; i, mask: Integer): Integer;
26772: function CreateOrGetOleObject(const ClassName: string): IDispatch;
26773: function FmtStrToInt(S: string): Integer;
26774: function MakeCellStrLV(const Text: String; Index: Integer; listview: TListview): String;
26775: https://docwiki.embarcadero.com/Libraries/Sydney/en/System.SysUtils.EmptyStr
26776: const emptystring= '';
26777:
26778: procedure SIRegister_TIdHashMessageDigest5(CL: TPSPascalCompiler);
26779: begin
26780:   //with RegClassS(CL,'TIdHashMessageDigest4', 'TIdHashMessageDigest5') do
26781:   with CL.AddClassN(CL.FindClass('TIdHashMessageDigest4'),'TIdHashMessageDigest5') do begin
26782:     Constructor Create;
26783:     Procedure Free;
26784:     Function HashValue(const ASrc: string): T4x4LongWordRecord);
26785:     Function HashValue1(AStream : TStream): T4x4LongWordRecord);
26786:     Function HashValueString(const ASrc: string): T4x4LongWordRecord);
26787:     Function HashValueStream(AStream : TStream): T4x4LongWordRecord);
26788:     Function AsHex( const AValue : T4x4LongWordRecord ) : string);
26789:     function hashStreamAsHex(s: TStream): string;
26790:   end;
26791: end;
26792:
26793: procedure SIRegister_SeSHA256(CL: TPSPascalCompiler);
26794: begin
26795:   CL.AddConstantN('seHEADERSIZE','LongInt').SetInt( 80);
26796:   CL.AddConstantN('emptystring','string').setstring('');
26797:   CL.AddTypeS('T32se', 'array [0 .. 31] of byte;
26798:   CL.AddTypeS('THeaderse', 'array [0 .. seHEADERSIZE - 1] of byte;
26799:   //T32 = array [0 .. 31] of byte;
26800:   // To store the block header
26801:   // THeader = array [0 .. HEADERSIZE - 1] of byte;
26802:   // CL.AddTypeS('PSHA256HASH', '^TSHA256HASH // will not work');
26803:   Function CalcSHA256String( Msg : AnsiString) : TSHA256HASH;
26804:   Function CalcSHA256Stream( Stream : TStream) : TSHA256HASH;
26805:   Function seSHA256ToStr( Hash : TSHA256HASH) : String);
26806:   Function SHA256ToBinaryStr( Hash : TSHA256HASH) : AnsiString);
26807:   Function ReverseHash( Hash : string) : string);
26808:   Function GetMemoryStream : TMemoryStream);
26809:   Function T32ToString( const at32 : T32se) : string);
26810:   Function CalcHeaderSHA256( aHeader : THeaderse) : TSHA256HASH;
26811: end;

```

```

26812:
26813: procedure SIRegister_BlocksUnit(CL: TPSPascalCompiler);
26814: begin
26815:   CL.AddTypeS('TCrypto', '( tcBitcoin, tcPascalCoin );
26816:   CL.AddTypeS('TNetbc', '( tnMainNet, tnTestNet );
26817:   CL.AddTypeS('PByte', 'Byte');
26818:   SIRegister_TBBlockFile(CL);
26819:   CL.AddTypeS('TBBlockHeader', 'record versionNumber : UInt32; aPreviousBlockHas'
26820:   +h: T32se; aMerkleRoot: T32se; time: UInt32; DifficultyTarget : UInt32; nonce: UInt32; end');
26821:   SIRegister_TInputbc(CL);
26822:   SIRegister_TOutputbc(CL);
26823:   SIRegister_TInputsbc(CL);
26824:   SIRegister_TOutputsbc(CL);
26825:   SIRegister_TTransaction(CL);
26826:   SIRegister_TBBlockTransactions(CL);
26827:   SIRegister_TBBlockRecord(CL);
26828:   CL.AddTypeS('TStartFileBlockFoundNotify', 'Procedure ( const aBlockFiles: tstringlist);
26829:   TFoundFileBlockNotify', 'Procedure ( const aBlockFile : TBBlockFi'
26830:   +le; const actualFileBlockNumber, totalBlockFiles : integer; var next : boolean);
26831:   TEndFilesBlockFoundNotify', 'Procedure ( const aBlockFiles : tstringlist);
26832:   TFoundBlockNotify', 'Procedure(const aBlock: TBBlockRecord; var findnext: boolean);
26833:   TBBlockProcessStepNotify', 'Procedure ( const aPos, asize : int64);
26834:   TEndProcessBlockFile', 'Procedure ( const aBlockFile : TBBlockFile);
26835:   SIRegister_TBBlocks(CL);
26836: end;
26837:
26838: procedure SIRegister_DelticsCommandLine(CL: TPSPascalCompiler);
26839: begin
26840:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TCommandLine');
26841:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TCommandLineSwitch');
26842:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EInvalidCommandLine');
26843:   //CL.AddTypeS('PSwitchDef', '^TSwitchDef // will not work');
26844:   CL.AddTypeS('TSwitchDef', 'record Name : String; Switch : String; Default : S'
26845:   +tring; MinValues : Integer; MaxValues : Integer; Separator : Char; end');
26846:   SIRegister_TCommandLine(CL);
26847:   SIRegister_TCommandLineSwitch(CL);
26848:   Procedure CommandLineToArgs( const aString : String; const aArgs : TStrings);
26849: end;
26850:
26851: (*-----*)
26852: procedure SIRegister_DelticsStrUtils(CL: TPSPascalCompiler);
26853: begin
26854:   CL.AddConstantN('LIKENESS_MATCH', 'Char').SetString( #255);
26855:   CL.AddTypeS('TANSIStringArray', 'array of ANSIString');
26856:   //CL.AddTypeS('TStringArray', 'array of String');
26857:   SIRegister_IStringList(CL);
26858:   SIRegister_IStringTool(CL);
26859:   SIRegister_TStringTool(CL);
26860:   Procedure dArrayInit( var aArray : TANSIStringArray; const aSize : Integer);
26861:   Procedure ArrayAdd1( var aArray : TANSIStringArray; const aValue : ANSIString);
26862:   Procedure ArrayAdd2( var aArray : TANSIStringArray; const aValue : array of ANSIString);
26863:   Procedure ArrayExtend( var aArray : TANSIStringArray; const aCount : Integer);
26864:   Function Likeness( const A : ANSIString; const B : ANSIString) : Integer;
26865:   Function LikenessEx(const A:ANSIString;const B:ANSIString; var strA:ANSIString;var
strB:ANSIString):Integer;
26866:   Function LikenessEx2(const A:ANSIString;const B:ANSIString;var strA:ANSIString;var strB:ANSIString;var
iSame :
Int;var iTotal:Int):Int;
26867:   Function CamelCapsToWords( const aString : String) : String;
26868:   Function ReverseStr( const aString : String) : String;
26869:   Function ReverseStr2( const aString : ANSIString) : ANSIString;
26870:   Function dSplit( const aString : String; const aDelim : Char; const aSlices : TStrings) : Integer;
26871:   Function dSplit1( const aString : String; const aDelim : Char; var aSlices : TStringArray) : Integer;
26872:   Function dSplit2( const aString : ANSIString; const aDelim : ANSIChar; var aSlices : TANSIStringArray);
26873:   Procedure dSplit2( const aString : ANSIString; const aDelim : ANSIChar; var aSlices : TANSIStringArray);
26874:   Function StrBeginsWith( const aString : String; const aBeginning : String) : Boolean;
26875:   Function StrBeginsWithText( const aString : String; const aBeginning : String) : Boolean;
26876:   Function dStrContains( const aString : String; const aSubString : String) : Boolean;
26877:   Function StrContainsText( const aString : String; const aSubString : String) : Boolean;
26878:   Function StrEndsWith( const aString : String; const aEnd : String) : Boolean;
26879:   Function StrEndsWithText( const aString : String; const aEnd : String) : Boolean;
26880:   Function StrLPad( const aString : String; const aWidth : Integer; const aTrunc : Boolean) : String;
26881:   Function StrRPad( const aString : String; const aWidth : Integer; const aTrunc : Boolean) : String;
26882:   Function StrDequote( const aString : String) : String;
26883:   Function StrPop( var aString : String; const aDelim : Char) : String;
26884:   Function StrPopQuoted( var aString : String; const aDelim : Char) : String;
26885:   Function dIsAlpha( const aChar : Char) : Boolean;
26886:   Function dIsDigit( const aChar : Char) : Boolean;
26887:   Function dIsLower( const aChar : Char) : Boolean;
26888:   Function dIsUpper( const aChar : Char) : Boolean;
26889:   //SIRegister_TManagedStringList(CL);
26890:   function delticsExplode(const Separator, S: string; Limit: Integer = 0): TStringArray;
26891:   function delticsImplode(const Glue: string; const Pieces: array of string): string;
26892: end;
26893:
26894: (*-----*)
26895: procedure SIRegister_DelticsSysUtils(CL: TPSPascalCompiler);
26896: begin
26897:   CL.AddConstantN('delticsEmptyStr', 'String').SetString( '');
26898:   //CL.AddConstantN('NullGUID', 'TGUID').SetString( '{00000000-0000-0000-0000-000000000000}');

```

```

26899: SIRegister_IAutoFree(CL);
26900: //CL.AddTypeS('Exception', 'Exception');
26901: SIRegister_ENotImplemented(CL);
26902: CL.AddClassN(CL.FindClass('TOBJECT'),'EAccessDenied');
26903: CL.AddTypeS('TGUIDFormat', '( gfDefault, gfNoBraces, gfNoHyphens, gfDigitsOnly );
26904: CL.AddTypeS('TRoundingStrategy', '( rsDefault, rsAwayFromZero, rsTowardsZero );
26905: CL.AddTypeS('TriBoolean', '( tbUnknown, tbTRUE, tbFALSE );
26906: Procedure deltCloneList( const aSource : TStringList; const aDest : TStringList);
26907: //CL.AddDelphiFunction('Function AutoFree( const aReference : PObject ) : IUnknown;
26908: //('Function AutoFree5( const aReferences : array of PObject ) : IUnknown;
26909: Procedure deltFreeAndNIL( var aObject: Tstrings);
26910: //Procedure NILRefs( const aObjects : array of PObject);
26911: Function deltIfThen7( aValue : Boolean; aTrue, aFalse : Boolean) : Boolean;
26912: Function deltIfThen8( aValue : Boolean; aTrue, aFalse : TObject) : TObject;
26913: Function deltIfThen9( aValue : Boolean; aTrue, aFalse : Integer) : Integer;
26914: Function deltIfThen10( aValue : Boolean; aTrue, aFalse : ANSIString) : ANSIString;
26915: Function deltIfThen11( aValue : Boolean; aTrue, aFalse : String) : String;
26916: Function deltIfThenInt( aValue : Boolean; aTrue, aFalse : Integer) : Integer;
26917: Function deltStringIndex( const aString : String; const aCases : array of String : Integer);
26918: Function deltTextIndex( const aString : String; const aCases : array of String : Integer);
26919: Function deltMin64( ValueA, ValueB : Int64) : Int64;
26920: Function deltMin12( ValueA, ValueB : Cardinal) : Cardinal;
26921: Function deltMin13( ValueA, ValueB : Integer) : Integer;
26922: Function deltMax64( ValueA, ValueB : Int64) : Int64;
26923: Function deltMax14( ValueA, ValueB : Cardinal) : Cardinal;
26924: Function deltMax15( ValueA, ValueB : Integer) : Integer;
26925: Procedure deltExchange( var A, B, aSize : LongWord);
26926: Function deltGUIDToString( const aGUID : TGUID; const aFormat : TGUIDFormat) : String;
26927: Function deltStringToGUID( const aString : String) : TGUID;
26928: Function deltTryStringToGUID( const aString : String; var aGUID : TGUID) : Boolean;
26929: Function deltIsNull( const aValue : TGUID) : Boolean;
26930: Function deltNewGUID : TGUID;
26931: Function deltSameGUID( const GUIDA, GUIDB : TGUID) : Boolean;
26932: Procedure deltAddTrailingBackslash( var aString : String);
26933: Procedure deltRemoveTrailingBackslash( var aString : String);
26934: Function deltBinToHex( const aBuf : string; const aSize : Integer) : String;
26935: Function deltReverseBytes16( const aValue : Word) : Word;
26936: Function deltReverseBytes17( const aValue : LongWord) : LongWord;
26937: Function deltReverseBytes18( const aValue : Int64) : Int64;
26938: Function deltRound( const aValue : Extended; const aStrategy : TRoundingStrategy) : Integer;
26939: //Procedure ForEachComponent(const aComponent:TComponent; const aProc:TComponentProc;const
aRecursive:Bool; const aClass:TComponentClass);
26940: Function deltExec( const aEXE : String; const aCommandLine : String) : Cardinal;
26941: Procedure deltExecAndWait( const aEXE : String; const aCommandLine : String);
26942: Function deltFindProcess( const aEXEName : String; var aProcess : TProcessEntry32) : Boolean;
26943: Function deltProcessExists( const aEXEName : String) : Boolean;
26944: Function deltRegisterDLL( const aFileName : String; const aRegistrationProc:String):Boolean;
26945: Function deltIsTRUE( const aTri : TriBoolean) : Boolean;
26946: Function deltIsFALSE( const aTri : TriBoolean) : Boolean;
26947: Function deltIsKnown( const aTri : TriBoolean) : Boolean;
26948: end;
26949:
26950: procedure SIRegister_U_Splines(CL: TPSPascalCompiler);
26951: begin
26952: CL.AddConstantN('MaxNoVertices','LongInt').SetInt( 100);
26953: CL.AddTypeS('TSplineVertex', 'record X : Single; Y : single; end');
26954: CL.AddTypeS('TSplineRow', 'array of TSplineVertex');
26955: SIRegister_TBSPline(CL);
26956: SIRegister_TSplines(CL);
26957: //CL.AddDelphiFunction('Procedure Register');
26958: SIRegister_U_SpringMass2(CL);
26959: CL.AddDelphiFunction('function getPython: string;
26960: CL.AddDelphiFunction('function getPython2: string;
26961: CL.AddDelphiFunction('function getPython3: string;
26962: end;
26963:
26964: procedure SIRegister_MARSCoreUtils(CL: TPSPascalCompiler);
26965: begin
26966: Function CreateCompactGuidStr : string;
26967: //CL.AddDelphiFunction('Function BooleanToTJSON( AValue : Boolean) : TJSONValue');
26968: Function SmartConcat(const AArgs:array of string;const ADelimiter:string;const AAvoidDuplicateDelimiter:
Boolean;const ATrim:Boolean; const ACaseInsensitive:Boolean):string;
26969: Function EnsurePrefix2( const AString, APrefix : string; const AIgnoreCase : Boolean) : string;
26970: Function EnsureSuffix( const AString, ASuffix : string; const AIgnoreCase : Boolean) : string;
26971: Function StringArrayToString( const AArray : array of string; const ADelimiter : string) : string;
26972: Procedure CopyStream2(ASourceStream,
ADestStream:TStream;AOverWriteDest:Bool;AThenResetDestPosition:Boolean);
26973: Function DateToISO8601( const ADate : TDateTime; AInputIsUTC : Boolean) : string;
26974: Function ISO8601ToDate( const AISODate : string; AReturnUTC : Boolean) : TDateTime;
26975: Function IsMask( const AString : string) : Boolean;
26976: Function MatchesMask2( const AString, AMask : string) : Boolean;
26977: Function StreamToBase64( const AStream : TStream) : string;
26978: Procedure Base64ToStream( const ABase64 : string; const ADestStream : TStream);
26979: Function StreamToBytes( const ASource : TStream) : TBytes;
26980: //Function GetEncodingName( const AEncoding : TEncoding) : string;
26981: function StringFallback(const AStrings: array of string; const ADefault: string) : string;
26982: function StripPrefix(const APrefix, AString: string) : string;
26983: function StripSuffix(const ASuffix, AString: string) : string;
26984: end;

```

```

26985:
26986: procedure SIRegister_U_CoasterB(CL: TPSPascalCompiler);
26987: begin
26988:   //CL.AddTypeS('float', 'single');
26989:   CL.AddTypeS(tcoasterRealPoint, 'record x : single; y : single; end');
26990:   TTrackPoint, 'record x : single; y : single; angleright : single; '
26991:   +sinangleright : single; cosangleright : single; quadrantright : integer; len'
26992:   +gthright : single; radiusright : single; end');
26993:   TTrackPointArray, 'array of TTrackPoint'; //carts, 'array of TCartLocRec'
26994:   TTrackRec, 'record x : single; y : single; index : integer; distT'
26995:   +pend : single; quadrant : integer; radius: single; angle: single; sinangle '
26996:   +: single; cosangle : single; end');
26997:   TCartLocRec, 'record p1 : tpoint; p2: tpoint; p3: tpoint; p4: tpoint; wr: integer; end');
26998:   TCartLocRecArray, 'array of TCartLocRec';
26999:   TLoopType, '( Insideloop, Outsideloop, none );
27000:   TTemplateMode, '( normal, makeline1, makeline2, makecircle1, makecircle2, movecircle );
27001:   CL.AddTypeS(ARRsoundstr, 'array [0..5] of string');
27002:   //type ARRstr = array [0..5] of string;
27003:   SIRegister_TcoasterLine(CL);
27004:   SIRegister_TcoasterCircle(CL);
27005:   SIRegister_TCoaster(CL);
27006: end;
27007:
27008: procedure SIRegister_TSpring(CL: TPSPascalCompiler);
27009: begin
27010:   //with RegClassS(CL, 'tPanel', 'TSpring') do
27011:   with CL.AddClassN(CL.FindClass('tPanel'), 'TSpring') do begin
27012:     K, 'single', iptrw); M, 'single', iptrw);
27013:     G, 'single', iptrw); A, 'single', iptrw);
27014:     x, 'single', iptrw); v, 'single', iptrw);
27015:     C, 'single', iptrw); X0, 'single', iptrw);
27016:     V0, 'single', iptrw); XEnd, 'single', iptrw);
27017:     InitiallyConstrained, 'Boolean', iptrw);
27018:     Timeinc, 'single', iptrw);
27019:     Scale, 'single', iptrw);
27020:     Stype, 'integer', iptrw);
27021:     delay, 'integer', iptrw);
27022:     pts, 'Tspringpointarray', iptrw);
27023:     step, 'integer', iptrw);
27024:     wraps, 'integer', iptrw);
27025:     drawColor, 'TColor', iptrw);
27026:     Constructor create( Aowner : TPanel; r : Trect);
27027:     Procedure drawspring( Len : single);
27028:     Procedure erasespring);
27029:     Procedure redrawinitial);
27030:     Function Setup(newV0, newX0, newC, newK, newM, newG, newTimeInc: single; newconstraint: bool;
27031:                   Nbrloops, newStep, newdelay: integer; newcolor: TColor): bool);
27032:     Procedure animate(form1: TForm; memo2: Tmemo);
27033:     Procedure animate2(form1: TForm; memo2: Tmemo; detail: boolean);
27034:     //procedure animate2(form1: TForm; memo2: Tmemo; detail: boolean);
27035:     Procedure sizechanged( sender : TObject);
27036:     Function GetMaxAmp : single);
27037:   end;
27038: end;
27039:
27040: procedure SIRegister_clJsonParser(CL: TPSPascalCompiler);
27041: begin
27042:   SIRegister_EclJSONError(CL);
27043:   CL.AddClassN(CL.FindClass('TObject'), 'TclJSONString');
27044:   CL.AddClassN(CL.FindClass('TObject'), 'TclJSONPair');
27045:   CL.AddClassN(CL.FindClass('TObject'), 'TclJSONObject');
27046:   CL.AddClassN(CL.FindClass('TObject'), 'TclJSONArray');
27047:   SIRegister_TclJSONBase(CL);
27048:   SIRegister_TclJSONPair(CL);
27049:   SIRegister_TclJSONValue(CL);
27050:   SIRegister_TclJSONString(CL);
27051:   SIRegister_TclJSONBoolean(CL);
27052:   SIRegister_TclJSONArray(CL);
27053:   SIRegister_TclJSONObject(CL);
27054:   CL.AddConstantN('cUnexpectedDataEndCode', 'LongInt').SetInt( - 100);
27055:   'cUnexpectedDataSymbolCode', 'LongInt').SetInt( - 101);
27056:   'cInvalidControlSymbolCode', 'LongInt').SetInt( - 102);
27057:   'cInvalidUnicodeEscSequenceCode', 'LongInt').SetInt( - 103);
27058:   'cUnrecognizedEscSequenceCode', 'LongInt').SetInt( - 104);
27059:   'cUnexpectedDataTypeCode', 'LongInt').SetInt( - 106);
27060:   function GetConnectionKind2(var strKind: string): Boolean;
27061: end;
27062:
27063: procedure SIRegister_SynHighlighterPython(CL: TPSPascalCompiler);
27064: begin
27065:   CL.AddTypeS(TtkTokenKindpy, '( tkComment, tkIdentifier, tkKey, tkNull, tkNumb'
27066:   +er, tkSpace, tkString, tkSymbol, tkNonKeyword, tkTrippleQuotedString, tkSy'
27067:   +stemDefined, tkHex, tkOct, tkFloat, tkUnknown );
27068:   CL.AddTypeS('TRangeStatepy', '( rsANil, rsComment, rsUnKnown, rsMultilineString'
27069:   +', rsMultilineString2, rsMultilineString3 );
27070:   SIRegister_TSynPythonSyn(CL);
27071: end;
27072:
27073: procedure SIRegister_DudsCommonDelphi(CL: TPSPascalCompiler);

```

```

27074: begin
27075:   Function DelphiIDERunning : Boolean'';
27076:   Function GetDelphiVersionRegistryKey( Version : Integer) : String'';
27077:   Function GetDelphiRootDirectory( Version : Integer) : String'';
27078:   Function IsDelphiVersionInstalled( Version : Integer) : Boolean'';
27079:   Function GetDelphiEnvironmentVariables(Version:Int;const EnvironmentVariables:TStrings):Bool;
27080:   Function GetDelphiLibraryPaths (Version:Int;const
LibraryPaths:TStrings;EvaluateMacros:Bool;CheckPaths:Bool):Bool;
27081:   function StrConcat (ArrS:array of string;Space:string):string;
27082:   function StrDelSpc (s:string):string;
27083:   function StrSplit2 (S:string;Ch:char):TStringDynArray;
27084:   procedure StrReplaceFirst (var S:string;olds,news:string);
27085:   procedure StrReplaceLast (var S:string;OldS,NewS:string);
27086:   procedure ConvertV3ToS (X,Y,Z:extended;var r,a,b:extended);
27087:   procedure ConvertSToV3 (R,A,B:extended;var X,Y,Z:extended);
27088:   procedure InterpolateColor (Color0,Color1:TColor;Count:integer;var TempColor:array of TColor);
27089:   function TryStrToextended (const S: string; out Value: extended): Boolean;
27090:   function isAVXSupported: Boolean;
27091:   function GetEXEVersionData (const FileName: string): TEXEVersionData;
27092:   function txtsearch(pat: string; text: string): integer;
27093:   procedure GetRegisteredAntiviruses (ProgIDs: TStrings);
27094:   type
27095:     TEXEVersionData = record
27096:       CompanyName,
27097:       FileDescription, FileVersion,
27098:       InternalName,
27099:       LegalCopyright, LegalTrademarks,
27100:       OriginalFileName, ProductName,
27101:       ProductVersion, Comments,
27102:       PrivateBuild, SpecialBuild: string;
27103:     end;
27104:   type
27105:     TNeuronClass = class (TObject)
27106:     private
27107:       FThreshold: Single;
27108:       FInputCount: Integer;
27109:       FInputs:TSingleDynArray;
27110:       FWeights:TSingleDynArray;
27111:       FAnalogOutput: Boolean;
27112:       function GetInput (Index: Integer): Single;
27113:       function GetOutput: Single;
27114:       function GetWeight (Index: Integer): Single;
27115:       procedure SetInput (Index: Integer; const Value: Single);
27116:       procedure SetWeight (Index: Integer; const Value: Single);
27117:       procedure SetInputCount (const Value: Integer);
27118:       function GetLoad: Single;
27119:     public
27120:       Constructor Create (aInputCount:Integer);
27121:       procedure RandomInitialize;
27122:       Property Load:Single read GetLoad;
27123:       property Output:Single read GetOutput;
27124:       property Threshold:Single read FThreshold write FThreshold;
27125:       property Inputs[Index: Integer]: Single read GetInput write SetInput;
27126:       property Weights[Index: Integer]: Single read GetWeight write SetWeight;
27127:       property InputCount:Integer read FInputCount write SetInputCount;
27128:       property AnalogOutput:Boolean read FAnalogOutput write FAnalogOutput;
27129:   end;
27130:   TPJConsoleApp:= class (TPJCustomConsoleApp) Console Application Runner CAR
27131:   public
27132:     // Make all inherited protected properties public
27133:     property StdIn;
27134:     property StdOut;
27135:     property StdErr;
27136:     property CommandLine;
27137:     property CurrentDir;
27138:     property Visible;
27139:     property MaxExecTime;
27140:     property TimeSlice;
27141:     property KillTimedOutProcess;
27142:     property ProcessAttrs;
27143:     property ThreadAttrs;
27144:     property UseNewConsole;
27145:     property ConsoleTitle;
27146:     property ConsoleColors;
27147:     property ScreenBufferSize;
27148:     property WindowPosition;
27149:     property WindowSize;
27150:     property Environment;
27151:     property UnicodeEnvironment;
27152:     property Priority;
27153:     property TimeToLive;
27154:     property ElapsedTime;
27155:     property ProcessInfo;
27156:     property ExitCode;
27157:     property ErrorCode;
27158:     property ErrorMessage;
27159:     property OnStart;
27160:     property OnWork;
27161:     property OnComplete;

```



```

27162:   end;
27163:
27164: procedure SIRegister_uHTMLBuilder(CL: TPSPascalCompiler);
27165: begin
27166:   SIRegister_THTMLBase(CL);
27167:   SIRegister_THTMLCell(CL);
27168:   CL.AddTypeS('TAfterAddRow', 'Procedure (Table:THTMLBase; DataSet: TDataSet);
27169:   SIRegister_THTMLItem(CL);
27170:   SIRegister_THTMLParagraph(CL);
27171:   SIRegister_THTMLRow(CL);
27172:   SIRegister_THTMLTable(CL);
27173:   SIRegister_THTMLReport(CL);
27174:   SIRegister_THTMLBuild(CL);
27175: end;
27176: https://github.com/guitorres/htmlbuilder/blob/master/src/uHTMLBuilder.pas
27177:
27178: procedure SIRegister_RestJsonUtils(CL: TPSPascalCompiler);
27179: begin
27180:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidValue');
27181:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidValueForField');
27182:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJsonInvalidSyntax');
27183:   SIRegister_ENoSerializableClass(CL);
27184:   SIRegister_TJsonUtil(CL);
27185:   Function JavaToDelphiDateTime2( const dt : int64) : TDateTime);
27186:   Function DelphiToJavaDateTime2( const dt : TDateTime) : int64);
27187:   Function ISO8601DateToJavaDateTime( const str : String; var ms : Int64) : Boolean);
27188:   Function ISO8601DateToDelphiDateTime( const str : string; var dt : TDateTime) : Boolean);
27189:   Function DelphiDateTimeToISO8601Date2( dt : TDateTime) : string');
27190: end;
27191:
27192: procedure SIRegister_pxQRcode(CL: TPSPascalCompiler);
27193: begin
27194:   CL.AddConstantN('QR_ECLEVEL_L','LongInt').SetInt( 0);
27195:   ('QR_ECLEVEL_M','LongInt').SetInt( 1);
27196:   ('QR_ECLEVEL_Q','LongInt').SetInt( 2);
27197:   ('QR_ECLEVEL_H','LongInt').SetInt( 3);
27198:   Procedure CreateQRCodeBMP( const AText:string;const ABitmapStream:TStream;const ALevel:Byte;const
ASize:Int);
27199: end;
27200:
27201: procedure SIRegister_DelphiZXingQRCode(CL: TPSPascalCompiler);
27202: begin
27203:   TQRCodeEncoding','( qrcAuto,qrcNumeric,qrcAlphanumeric,qrcISO88591,qrcUTF8NoBOM,qrcUTF8BOM );
27204:   CL.AddTypeS('T2DBooleanArray', 'array of array of Boolean');
27205:   SIRegister_TDelphiZXingQRCode(CL);
27206: end;
27207:
27208: function DownloadArchive( sURL, sArchivoLocal: String ): boolean;
27209: function GetUrlContent(const Url: string): UTF8String;
27210: procedure FindFiles2List(StartDir,FileMask: string; recursively:boolean; var FilesList: TStringList);
27211: unit UtilsTimeCode;
27212: function TCtoFrames(S:String):Integer;
27213: function FramestoTC(Fi:Integer):String;
27214: function DFtoFrames(S:String):Integer;
27215: function FramestoDF(Fi:Integer):String;
27216: function ValidDropFrame(S:String):Boolean;
27217:
27218: unit uPSI_KLibUtils;
27219: procedure SIRegister_KLibUtils(CL: TPSPascalCompiler);
27220: begin
27221:   CL.AddTypeS('TResourceklib', 'record name : string; _type : string; end');
27222:   CL.AddTypeS('TAnonymousMethodklib',{reference to} 'procedure');
27223:   CL.AddTypeS('TCallbackklib', 'procedure(msg: string);
27224:   Procedure deleteFilesInDirklib( pathDir : string; const filesToKeep : array of string);
27225:   Procedure deleteFilesInDirWithStartingFileNameklib(dirName:string;
startingFileName:string;fileType:string);
27226:   Function checkIfFileExistsAndEmptyklib( fileName : string) : boolean);
27227:   Procedure deleteFileIfExistsklib( fileName : string);
27228:   Function getTextFromFileklib( fileName : string) : string);
27229:   Function checkIfThereIsSpaceAvailableOnDriveklib( drive : char; requiredSpaceInBytes:int64): boolean);
27230:   Function getFreeSpaceOnDriveklib( drive : char) : int64);
27231:   Function getIndexOfDriveklib( drive : char) : integer);
27232:   Function getDriveExeklib : char);
27233:   Function getDirSizeklib( path : string) : int64);
27234:   Function getCombinedPathWithCurrentDirklib( pathToCombine : string) : string);
27235:   Function getDirExeklib : string);
27236:   Procedure createDirIfNotExistsklib( dirName : string);
27237:   Function checkIfIsLinuxSubDirklib( subDir : string; mainDir : string) : boolean);
27238:   Function getPathInLinuxStyleklib( path : string) : string);
27239:   Function checkIfIsSubDirklib(subDir : string; mainDir: string; trailingPathDelimiter:char):boolean);
27240:   Function getValidFullPathklib( fileName : string) : string);
27241:   Function checkMD5Fileklib( fileName : string; MD5 : string) : boolean);
27242:   Procedure unzipResourceklib( nameResource : string; destinationDir : string);
27243:   //Function getPNGResource( nameResource : string) : TPngImage);
27244:   Procedure getResourceAsEXEFileklib( nameResource : string; destinationFileName: string);
27245:   Procedure getResourceAsZIPFileklib( nameResource : string; destinationFileName: string);
27246:   Procedure getResourceAsFileklib( resource : TResourceklib; destinationFileName: string);
27247:   Function getResourceAsString( resource : TResourceklib) : string);
27248:   Function getResourceAsStringklib( resource : TResourceklib) : string);

```

```

27249: Function getResourceAsStreamklib( resource : TResourceklib ) : TResourceStream';
27250: Procedure unzipklib( zipFileName : string; destinationDir : string; deleteZipAfterUnzip : boolean);
27251: //Function checkRequiredFTPProperties( FTPCredentials : TFTPCredentials ) : boolean';
27252: Function getValidItalianTelephoneNumbersklib( number : string) : string';
27253: Function getValidTelephoneNumbersklib( number : string) : string';
27254: Function getRandStringklib( size : integer) : string';
27255: Function getFirstFileNameInDirklib( dirName : string; fileType: string; fullPath: boolean): string';
27256: Function getFileNameListInDirklib( dirName : string; fileType: string; fullPath: boolean): TStringList';
27257: Procedure saveToFileklib( source : string; fileName : string);
27258: Function getCombinedPathklib( path1 : string; path2 : string) : string';
27259: Function getCurrentDayOfWeekAsStringklib : string';
27260: Function getDayOfWeekAsStringklib( date : TDateTime) : string';
27261: Function getCurrentDateTimeAsStringklib : string';
27262: Function getDateTimeAsStringklib( date : TDateTime) : string';
27263: Function getCurrentDateAsStringklib : string';
27264: Function getDateAsStringklib( date : TDateTime) : string';
27265: Function getCurrentTimestampklib : string';
27266: Function getCurrentDateTimeAsStringWithFormattingklib( formatting : string) : string';
27267: Function getDateTimeAsStringWithFormattingklib( value : TDateTime; formatting: string): string';
27268: Function getCurrentDateTimeklib : TDateTime';
27269: Function getParsedXMLStringklib( mainString : string) : string';
27270: Function getDoubleQuotedStringklib( mainString : string): string';
27271: Function getSingleQuotedStringklib( mainString : string): string';
27272: Function
getMainStringWithSubStringInsertedklib(mainString:string;insertedString:string;index:integer):string;
27273: Function getStringWithoutLineBreaksklib( mainString : string; substituteString : string) : string';
27274: Function getCSVFieldFromStringAsDateklib( mainString:string; index : integer; delimiter : Char): TDate;
27275: Function getCSVFieldFromStringAsDateklib(mainString:string;index:integer;formatSettings:TFormatSettings;
delimiter:Char):TDate;
27276: Function getCSVFieldFromStringAsDoubleklib( mainString:string; index:integer;delimiter:Char):Double;
27277: Function
getCSVFieldFromStringAsDouble2klib(mainString:string;index:integer;formatSettings:TFormatSettings;
delimiter:Char):Double;
27278: Function getCSVFieldFromStringAsIntegerklib(mainString:string;index:integer;delimiter:Char):integer';
27279: Function getCSVFieldFromStringklib( mainString : string; index : integer; delimiter : Char) : string';
27280: Function getNumberOfLinesInStrFixedWordWrapklib( source : string) : integer';
27281: Function stringToStrFixedWordWrapklib( source : string; fixedLen : integer) : string';
27282: Function stringToStringListWithFixedLenklib( source : string; fixedLen : integer) : TStringList';
27283: Function stringToStringListWithDelimiterklib( value : string; delimiter : Char) : TStringList';
27284: Function stringToTStringListklib( source : string) : TStringList';
27285: Function arrayOfStringToTStringListklib( arrayOfStrings : array of string) : TStringList';
27286: Procedure splitStringsklib(source:str;delimiter:str;var destFirstString:str;var destSecondString: string);
27287: Procedure splitStrings2klib(source:str;delimiterPosition:int;var destFirstString:str;var
destSecondString:str);
27288: Function getMergedStringsklib( firstString : string; secondString : string; delimiter: string) : string';
27289: Function checkIfEmailIsValidklib( email : string) : boolean';
27290: Function
checkIfMainStringContainsSubStringNoCaseSensitiveklib(mainString:string;subString:string):boolean;
27291: Function
checkIfMainStringContainsSubStringklib(mainString:str;subString:str;caseSensitiveSearch:bool):bool;
27292: Function getDoubleAsStringklib( value : Double; decimalSeparator : char) : string';
27293: Function getFloatToStrDecimalSeparatorklib : char';
27294: Procedure tryToExecuteProcedureklib( myProcedure : TAnonymousMethodklib; raiseExceptionEnabled: boolean);
27295: Procedure tryToExecuteProcedure2klib( myProcedure : TCallbackklib; raiseExceptionEnabled : boolean);
27296: //Procedure tryToExecuteProcedure3( myProcedure : TProcedure; raiseExceptionEnabled : boolean);
27297: Procedure executeProcedureklib( myProcedure : TAnonymousMethodklib);
27298: Procedure executeProcedure2klib( myProcedure : TCallbackklib);
27299: function setResourceHInstanceklib(aresource: longword): longword;
27300: function setResourceHInstanceklib(aresource: longword): longword;
27301: function executeAndWaitExe(fileName:string; params:string; exceptionIfReturnCodeIsNot0:boolean):LongInt;
27302: end;
27303:
27304: procedure SIRegister_KLibWindows(CL: TPSPascalCompiler);
27305: begin
27306: //CL.AddDelphiFunction('Procedure downloadFile( info : TDownloadInfo; forceOverwrite : boolean);
27307: Function getFirstPortAvaliableklib( defaultPort : integer; host : string) : integer';
27308: Function checkIfPortIsAvaliableklib( host : string; port : Word) : boolean';
27309: Function checkIfAddressIsLocalhostklib( address : string) : boolean';
27310: Function getIPFromHostNameklib( hostName : string) : string';
27311: Function getIPklib : string';
27312: Function checkIfRunUnderWineklib : boolean';
27313: Function checkIfWindowsArchitectureIsX64klib : boolean';
27314: CL.AddTypeS('TPIDCredentials', 'record ownerUserName : string; domain : string; end');
27315: CL.AddTypeS('TWindowsArchitectureklib', '( WindowsX86, WindowsX64 );
27316: Function getWindowsArchitectureklib : TWindowsArchitectureklib');
27317: Function checkIfUserIsAdminklib : boolean';
27318: Function IsUserAnAdminklib : boolean';
27319: CL.AddTypeS('TShowWindowType', '( _SW_HIDE, _SW_SHOWNORMAL, _SW_NORMAL, _SW_S'
27320: +HOWMINIMIZED, _SW_SHOWMAXIMIZED, _SW_MAXIMIZE, _SW_SHOWNOACTIVATE, _SW_SHO'
27321: +W, _SW_MINIMIZE, _SW_SHOWMINNOACTIVE, _SW_SHOWNA, _SW_RESTORE, _SW_SHOWDEFAULT, _SW_FORCEMINIMIZE, _SW_MAX );
27322: Procedure openWebPageWithDefaultBrowserklib( url : string);
27323: Function
shellExecuteOpenklib(fileName:string;params:string;directory:string;showWindowType:TShowWindowType;
exceptionIfFunctionFails:boolean):integer');
27324: Function shellExecuteExeAsAdminklib(fileName:string;params:string;showWindowType : TShowWindowType;
exceptionIfFunctionFails:boolean):integer');
27325: Function shellExecuteExeklib(fileName:string;params:string;showWindowType:TShowWindowType;
exceptionIfFunctionFails:boolean;operation: string):integer');
27326: Function
myShellExecuteklib(handle:integer;operation:string;fileName:string;params:string;directory:string;
showWindowType:TShowWindowType;exceptionIfFunctionFails:boolean):integer');

```

```

27327: Function shellExecuteExCMDAndWaitklib(params:string;runAsAdmin:boolean;showWindowType:TShowWindowType;
exceptionIfReturnCodeIsNot0:boolean):LongInt');
27328: Function shellExecuteExAndWaitklib(fileName:string; params:string;runAsAdmin:boolean;showWindowType:
TShowWindowType;exceptionIfReturnCodeIsNot0:boolean) : LongInt');
27329: Function executeAndWaitExeklib(fileName:string;params:string;exceptionIfReturnCodeIsNot0:bool):LongInt');
27330: Function netShareklib(targetDir : string; netName : string; netPassw : string;
grantAllPermissionToEveryoneGroup : boolean) : string');
27331: Procedure addTCP_IN_FirewallExceptionklib(ruleName:string;port:Word;description:string;grouping:string;
executable:string)
27332: Procedure deleteFirewallExceptionklib( ruleName : string);
27333: // CL.AddTypes('TExplicitAccess', 'EXPLICIT_ACCESS_A');
27334: Procedure grantAllPermissionsNetToTheObjectForTheEveryoneGroupklib( myObject : string);
27335: Procedure grantAllPermissionsNetToTheObjectForTheUsersGroupklib( myObject : string);
27336: Procedure grantAllPermissionNetToTheObjectklib( windowsGroupOrUser : string; myObject : string);
27337: Procedure grantAllPermissionsToTheObjectForTheEveryoneGroupklib( myObject : string);
27338: Procedure grantAllPermissionsToTheObjectForTheUsersGroupklib( myObject : string);
27339: Procedure grantAllPermissionsToTheObjectklib( windowsGroupOrUser : string; myObject : string);
27340: Procedure grantAllPermissionsToTheObjectForTheEveryoneGroup2klib( myObject : string);
27341: Procedure grantAllPermissionsToTheObjectForTheUsersGroup2klib( myObject : string);
27342: Procedure grantAllPermissionsToTheObject2klib( windowsGroupOrUser : string; myObject : string);
27343: Function checkIfWindowsGroupOrUserExistsklib( windowsGroupOrUser : string) : boolean');
27344: Procedure createDesktopLinkklib( fileName : string; nameDesktopLink : string; description: string);
27345: Function getDesktopDirPathklib : string');
27346: Procedure copyDirIntoTargetDirklib( sourceDir : string; targetDir : string; forceOverwrite : boolean);
27347: Procedure copyDirklib( sourceDir : string; destinationDir : string; silent : boolean);
27348: Procedure createHideDirklib( dirName : string; forceDelete : boolean);
27349: Procedure deleteDirectoryIfExistsklib( dirName : string; silent : boolean);
27350: Procedure myMoveFileklib( sourceFileName : string; targetFileName : string);
27351: Procedure createEmptyFileIfNotExistsklib( filename : string);
27352: Procedure createEmptyFileklib( filename : string);
27353: Function checkIfIsWindowsSubDirklib( subDir : string; mainDir : string) : boolean');
27354: Function getParentDirklib( source : string) : string');
27355: Function getValidFullPathInWindowsStyleklib( path : string) : string');
27356: Function getPathInWindowsStyleklib( path : string) : string');
27357: Function getStringWithEnvVariablesReadedklib( source : string) : string');
27358: Function setProcessWindowToForegroundklib( processName : string) : boolean');
27359: Function getPIDOfCurrentUserByProcessNameklib( nameProcess : string) : DWORD');
27360: Function getWindowsUsernameklib : string');
27361: Function checkUserOfProcessklib( userName : String; PID : DWORD) : boolean');
27362: Function getPIDCredentialsklib( PID : DWORD) : TPIDCredentials');
27363: Function getPIDByProcessNameklib( nameProcess : string) : DWORD');
27364: Function getMainWindowHandleByPIDklib( PID : DWORD) : DWORD');
27365: Procedure closeApplicationklib( handle : THandle);
27366: Function sendMemoryStreamUsing_WM_COPYDATAklib( handle : THandle; data : TMemoryStream) : integer');
27367: Function sendStringUsing_WM_COPYDATAklib(handle:THandle; data:string; msgIdentifier: integer):integer');
27368: Procedure mySetForegroundWindowklib( handle : THandle);
27369: Function checkIfWindowExistsklib( className : string; captionForm : string) : boolean');
27370: Function myFindWindowklib( className : string; captionForm : string) : THandle');
27371: Function checkIfExistsKeyIn_HKEY_LOCAL_MACHINEklib( key : string) : boolean');
27372: Procedure waitForMultipleklib( processHandle : THandle; timeout: DWORD; modalMode : boolean);
27373: Procedure waitForklib( processHandle : THandle; timeout: DWORD; modalMode : boolean);
27374: Procedure raiseLastSysErrorMessageklib');
27375: Function getLastSysErrorMessageklib : string');
27376: Function getLocaleDecimalSeparatorklib : char');
27377: Procedure terminateCurrentProcessklib( exitCode : Cardinal; raiseExceptionEnabled : boolean);
27378: Procedure myTerminateProcessklib(processHandle:THandle;exitCode:Cardinal;raiseExceptionEnabled: boolean);
27379: //Function fixedGetNamedSecurityInfo( pObjectName : LPWSTR; ObjectType : SE_OBJECT_TYPE; SecurityInfo :
SECURITY_INFORMATION; ppsidOwner, ppsidGroup : PPSID; ppDacl, ppSacl : PPACL; var ppSecurityDescriptor :
PSECURITY_DESCRIPTOR) : DWORD');
27380: end;
27381:
27382: https://github.com/lonewolfonline/delphi-utilities/blob/master/AzuliasUtils.pas
27383:
27384: procedure SIRegister_TAzuliaHTML(CL: TPSPascalCompiler);
27385: begin
27386: //with RegClassS(CL,'TComponent', 'TAzuliasHTML') do
27387: with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliasHTML') do begin
27388: RegisterMethod('Function HeadTop( Title : string) : string');
27389: Function HeadClose : string');
27390: Function BodyTop(LeftMargin,RightMargin:Integer;TextColor,LinkColor,VisitedLinkColor,ActiveLinkColor,
BackgroundImage:string;FixedBackground:bool;Extra:string):string');
27391: Function BodyClose : string');
27392: Function ConvertDOSPathToHTML( WindowsPath : string) : string');
27393: Function ConvertHTMLPathToDOS( WindowsPath : string) : string');
27394: end;
27395: end;
27396: (*-----*)
27397: procedure SIRegister_TAzuliaDisk(CL: TPSPascalCompiler);
27398: begin
27399: //with RegClassS(CL,'TComponent', 'TAzuliasDisk') do
27400: with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliasDisk') do begin
27401: Function GetDiskLabel( Drive : char) : string');
27402: Function GetDiskSerial( Drive : char) : string');
27403: Function GetDiskFileSystem( Drive : char) : string');
27404: Function GetDiskType( Drive : char) : integer');
27405: Function GetDiskFlags( Drive : char) : integer');
27406: Procedure ChangeDiskLabel( Drive : char; NewLabel : string);
27407: end;
27408: end;

```

```

27409: (*-----*)
27410: procedure SIRegister_TAzuliaSpeaker(CL: TPSPascalCompiler);
27411: begin
27412:   //with RegClassS(CL,'TComponent', 'TAzuliaSpeaker') do
27413:   with CL.AddClassN(CL.FindClass('TComponent'),'TAzuliaSpeaker') do begin
27414:     Procedure Delay( MSecs : Integer);
27415:     Procedure Play( Freq : Word; MSecs : Integer);
27416:     Procedure Stop';
27417:   end;
27418: end;
27419: (*-----*)
27420: procedure SIRegister_TAzuliaFiles(CL: TPSPascalCompiler);
27421: begin
27422:   //with RegClassS(CL,'TOBJECT', 'TAzuliaFiles') do
27423:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TAzuliaFiles') do begin
27424:     Procedure FindRecursive(const path: string;const mask:string;LogFunction:TLogFuncnt;Re_curse:Boolean);
27425:     Procedure SearchFileExt( const Dir, Ext : string; Files : TStrings);
27426:     Function GetFileSize( const FileName : string) : LongInt';
27427:     Function GetKBFileSize( FileSize : integer) : string';
27428:     Function LongToShortFilename( LongFilename : string) : string';
27429:     Function ShortToLongFilename( LongFilename : string) : string';
27430:     Procedure AppendDataToFilename( Filename, Data : string);
27431:     Procedure SaveDataToFilename( Filename, Data : string);
27432:     Procedure CopyFile( const FileName, DestName : string);
27433:     Procedure MoveFile( const FileName, DestName : string);
27434:     Function HasAttr( const FileName : string; Attr : Word) : Boolean';
27435:     Function ExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle';
27436:     Function GetWindowsDir : string';
27437:     Function GetTempDir : string';
27438:     Function GetINIStrng( Section, Ident, DefaultString, Filename : string) : string';
27439:   end;
27440: end;
27441: (*-----*)
27442: procedure SIRegister_TAzuliaStrings(CL: TPSPascalCompiler);
27443: begin
27444:   //with RegClassS(CL,'TOBJECT', 'TAzuliaStrings') do
27445:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TAzuliaStrings') do begin
27446:     Function NPos( C : Char; S : string; N : Byte) : Byte';
27447:     Function IntToRoman( Value : Longint) : string';
27448:     Function RomanToInt( const S : string) : Longint';
27449:     Function GetDirFromStr( DirectoryString : string; IndexFromRight : integer) : string';
27450:     Function RemoveChars( InputString : string; CharToRemove : Char) : string';
27451:     Function GetStringFromGarbage( InputString : string; Index : integer) : string';
27452:     Function SplitupWithGarbage( InputString : string; CommonSplitter : char) : string';
27453:     Function GetCount( Character : char; InputString : string) : integer';
27454:     Function RemoveTrailingChars( StringToRemoveFrom : string; Trailer : char) : string';
27455:   end;
27456: end;
27457: (*-----*)
27458: procedure SIRegister_TSystemRegistry(CL: TPSPascalCompiler);
27459: begin
27460:   //with RegClassS(CL,'TOBJECT', 'TSystemRegistry') do
27461:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TSystemRegistry') do begin
27462:     Function GetStringFromRegistry( Root : integer; Key, Value : string) : string';
27463:     Function SaveStringToRegistry( Root : integer; Key, Value, Name : string) : bool';
27464:     Procedure ToggleBinaryValue( RootKey : integer; Path, Key : string);
27465:     Procedure DeleteValueInRegistry( RootKey : integer; Path, Name : string);
27466:     Procedure RenameValueInRegistry( RootKey : integer; Path, OldName, NewName : string);
27467:   end;
27468: end;
27469: (*-----*)
27470: procedure SIRegister_TBrowseForFolderDialog(CL: TPSPascalCompiler);
27471: begin
27472:   //with RegClassS(CL,'TOBJECT', 'TBrowseForFolderDialog') do
27473:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TBrowseForFolderDialog') do begin
27474:     RegisterProperty('Title', 'string', iptrw);
27475:     Function Execute : Boolean';
27476:     RegisterProperty('Path', 'string', iptr);
27477:   end;
27478: end;
27479: (*-----*)
27480: procedure SIRegister_IShellLink(CL: TPSPascalCompiler);
27481: begin
27482:   //with RegClassS(CL,'TOBJECT', 'IShellLink') do
27483:   with CL.AddClassN(CL.FindClass('TOBJECT'),'IShellLink') do begin
27484:     Function GetPath(pszFile:LPSTR;cchMaxPath:integer; var pfd:TWin32FindData;fFlags:DWORD):HResult';
27485:     Function GetIDList( var ppidl : PITEMIDLIST) : HResult';
27486:     Function SetIDList( pidl : PITEMIDLIST) : HResult';
27487:     Function GetDescription( pszName : LPSTR; cchMaxName : integer) : HResult';
27488:     Function SetDescription( pszName : LPSTR) : HResult';
27489:     Function GetWorkingDirectory( pszDir : LPSTR; cchMaxPath : integer) : HResult';
27490:     Function SetWorkingDirectory( pszDir : LPSTR) : HResult';
27491:     Function GetArguments( pszArgs : LPSTR; cchMaxPath : integer) : HResult';
27492:     Function SetArguments( pszArgs : LPSTR) : HResult';
27493:     Function GetHotkey( var pwHotkey : word) : HResult';
27494:     Function SetHotkey( wHotkey : word) : HResult';
27495:     Function GetShowCmd( var piShowCmd : integer) : HResult';
27496:     Function SetShowCmd( iShowCmd : integer) : HResult';
27497:     Function GetIconLocation( pszIconPath LPSTR;cchIconPath:integer;var piIcon: integer): HResult';

```



```

27498:     Function SetIconLocation( pszIconPath : LPSTR; iIcon : integer) : HRESULT');
27499:     Function SetRelativePath( pszPathRel : LPSTR; dwReserved : DWORD) : HRESULT');
27500:     Function Resolve( Wnd : HWND; fFlags : DWORD) : HRESULT');
27501:     Function SetPath( pszFile : LPSTR) : HRESULT');
27502: end;
27503: end;
27504: (*-----*)
27505: procedure SIRegister_AzulialUtils(CL: TPSPascalCompiler);
27506: begin
27507:     CL.AddTypeS('azucharSet', 'set of char');
27508:     SIRegister_IShellLink(CL);
27509:     SIRegister_TBrowseForFolderDialog(CL);
27510:     SIRegister_TSystemRegistry(CL);
27511:     SIRegister_TAzuliaStrings(CL);
27512:     CL.AddTypeS('TLogFunc', 'Function ( const path : string; const SRec : TSearchRec) : Boolean');
27513:     SIRegister_TAzuliaFiles(CL);
27514:     SIRegister_TAzuliaSpeaker(CL);
27515:     SIRegister_TAzuliaDisk(CL);
27516:     SIRegister_TAzuliaHTML(CL);
27517:     CL.AddTypeS('GrabType', '( GTSCREEN, GTWINDOW, GTCLIENT )');
27518:     CL.AddClassN(CL.FindClass('TOBJECT'), 'EInvalidDest');
27519:     CL.AddClassN(CL.FindClass('TOBJECT'), 'EFCantMove');
27520:     Function azuBoolToStr( value : boolean) : string';
27521:     Procedure azuChangeWallpaper( FName : string; IsTiled, IsStretch : Boolean);
27522:     Function azuDoIExist( WndTitle : string) : Boolean';
27523:     Procedure azuAlert( Text : string);
27524:     Procedure azuTileImage( SourceImage, DestImage : TImage);
27525:     Function azuIncChar( iX : Char) : Char';
27526:     Function azuDecChar( iX : Char) : Char';
27527:     Procedure azuSaveForm( F : TForm; Filename : string);
27528:     Procedure azuLoadForm( F : TForm; Filename : string);
27529:     Function azuToggleBool( totoggle : bool) : bool';
27530:     Function azuremoveTrailingChars( const s : string; chars : azucharSet) : string';
27531:     Function azuremoveLeadChars( const s : string; chars : azucharSet) : string';
27532:     Function azuremoveChars( const s : string; chars : char) : string';
27533:     Procedure azuDelay( MSecs : Integer);
27534:     Function azuSHBrowseDialog( title : string; Handle : integer) : string';
27535:     Procedure azureregister filetype( Extention, REGDesc, UserDesc, Icon, Appl : string);
27536:     Procedure azuRemoveFileType( Extention, RegDesc : string);
27537:     Procedure azuGetFunctionNamesFromDLL( DLLName : string; List : TStrings);
27538:     Procedure azuSaveListViewToFile( AListView : TListView; sFileName : string);
27539:     Procedure azuLoadListViewToFile( AListView : TListView; sFileName : string);
27540:     Procedure azuLoadCSV( Filename : string; sg : TStringGrid);
27541:     Function azuJPEGDimensions( Filename : string; var X, Y : Word) : boolean';
27542:     Procedure azuResizeImage( FileName : string; MaxWidth : Integer);
27543:     Function azuSaveJPEGPictureFile( Bitmap: TBitmap; FilePath, FileName: string; Quality: Integer) : Boolean';
27544:     Function azuLoadJPEGPictureFile( Bitmap : TBitmap; FilePath, FileName : string) : Boolean';
27545:     Procedure azuSmoothResize( Src, Dst : TBitmap);
27546:     Function azuMsecToStr( Milli : Cardinal) : string';
27547:     Procedure azuDirToStrings( APath : string; AStrings : TStrings; WithExt, WithPath : boolean);
27548:     Procedure azuSetWallPaper( ImageFileName : string; IsTiled : boolean);
27549:     Function azuGetFileSize( FileName : string) : int64';
27550:     Function azuGetExeByExtension( sExt : string) : string';
27551:     Function azuRefreshScreenIcons : Boolean';
27552:     Function azuIndexToColor( AIndex : integer) : TColor';
27553:     Function azuColorToIndex( AColor : TColor) : integer';
27554:     Function azuTitleCase( Text2 : string) : string';
27555:     Function azuToggleCase( Text2 : string) : string';
27556:     Function azuSentenceCase( Text2 : string) : string';
27557:     Procedure azuGrabScreen( bm : TBitmap; gt : GrabType);
27558:     Procedure azuSaveStringGrid( StringGrid : TStringGrid; FileName : String);
27559:     Procedure azuLoadStringGrid( StringGrid : TStringGrid; FileName : String);
27560:     Function azuGetInetFile( const fileURL, FileName : String) : boolean';
27561:     Procedure azuRunOnStartup( sProgTitle, sCmdLine : string; bRunOnce : boolean);
27562:     Procedure azuRemoveOnStartup( sProgTitle : string);
27563:     CL.AddConstantN('SInvalidDest', 'String').SetString( 'Destination %s does not exist');
27564:     SFCantMove, 'String').SetString( 'Cannot move file %s');
27565:     AzuMsg1, 'String').SetString( 'File "%s" does not exist!');
27566:     AzuMsg2, 'String').SetString( '"%s" is not a ListView file!');
27567:     sr_WindowMetrics, 'String').SetString( 'Control Panel\Desktop\WindowMetrics\');
27568:     sr_ShellIconSize, 'String').SetString( 'Shell Icon Size');
27569: end;
27570:
27571: procedure SIRegister_TALWinInetHTTPClient2(CL: TPSPascalCompiler);
27572: begin
27573:     //with RegClassS(CL, 'TALHTTPClient2', 'TALWinInetHTTPClient2') do
27574:     with CL.AddClassN(CL.FindClass('TALHTTPClient2'), 'TALWinInetHTTPClient2') do begin
27575:         RegisterMethod('Constructor Create');
27576:         Procedure Free';
27577:         Procedure Connect';
27578:         Procedure Disconnect';
27579:         AccessType, 'TALWinInetHttpInternetOpenAccessType', iptrw);
27580:         InternetOptions, 'TALWininetHTTPClientInternetOptionSet', iptrw);
27581:         OnStatus, 'TALWinInetHTTPClientStatusEvent', iptrw);
27582:         IgnoreSecurityErrors, 'Boolean', iptrw);
27583:     end;
27584: end;
27585:
27586: procedure SIRegister_TALHTTPClient2(CL: TPSPascalCompiler);

```



```

27587: begin
27588:   //with RegClassS(CL,'TObject', 'TALHTTPClient') do
27589:   with CL.AddClassN(CL.FindClass('TObject'),'TALHTTPClient2') do begin
27590:     Constructor Create();
27591:     Procedure Free();
27592:     Procedure Get(const aUrl:AnsiString;const aRequestFields:TALStrings;const
aResponseContent:TStream;const aResponseHeader:TALHTTPResponseHeader2;const
ARequestHeaderValues:TALNameValueArray;const aEncodeRequestFields:Bool);
27593:     Procedure Get1(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);
27594:     Function Get2(const aUrl:AnsiString; const ARequestHeaderValues : TALNameValueArray) : AnsiString;
27595:     Function Get3(const aUrl:AnsiString;const aRequestFields:TALStrings;const ARequestHeaderValues:
TALNameValueArray;const aEncodeRequestFields:Boolean) : AnsiString;
27596:     Procedure Post(const aUrl:AnsiString; const aResponseContent:TStream;const aResponseHeader:
TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);
27597:     Procedure Post1(const aUrl:AnsiString;const aPostDataStream:TStream;const
aResponseContent:TStream;const aResponseHeader:TALHTTPResponseHeader2;const
ARequestHeaderValues:TALNameValueArray);
27598:     Function Post2(const aUrl: AnsiString; const ARequestHeaderValues: TALNameValueArray):AnsiString;
27599:     Function Post3(const aUrl:AnsiString;const aPostDataStream:TStream;const ARequestHeaderValues:
TALNameValueArray) :AnsiString;
27600:     Procedure PostUrlEncoded(const aUrl:AnsiString;const aRequestFields:TALStrings;const aResponseContent :
TStream; const aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray;const
aEncodeRequestFields:Bool);
27601:     Function PostUrlEncoded1(const aUrl:AnsiString;const aRequestFields:TALStrings;const
ARequestHeaderValues : TALNameValueArray;const aEncodeRequestFields:Boolean):AnsiString;
27602:     Procedure PostMultipartFormData(const aUrl:AnsiString;const aRequestFields:TALStrings;const
aRequestFiles:TALMultiPartFormDataContents;const aResponseContent:TStream;const aResponseHeader +
27603:     ':TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);
27604:     Function PostMultipartFormData1(const aUrl: AnsiString; const aRequestFields: TALStrings; const
aRequestFiles:TALMultiPartFormDataContents;const ARequestHeaderValues: TALNameValueArray) : AnsiString;
27605:     Procedure Head(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);
27606:     Function Head1(const aUrl:AnsiString; const ARequestHeaderValues: TALNameValueArray) : AnsiString;
27607:     Procedure Trace(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);
27608:     Function trace1(const aUrl:AnsiString; const ARequestHeaderValues:TALNameValueArray) : AnsiString;
27609:     Procedure Put(const aUrl:AnsiString;const aPutDataStream:TStream;const aResponseContent:TStream;const
aResponseHeader:TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray;
27610:     Function Put1(const aURL:AnsiString;const aPutDataStream:TStream;const ARequestHeaderValues:
TALNameValueArray):AnsiString;
27611:     Procedure Delete(const aUrl: AnsiString;const aResponseContent:TStream;const aResponseHeader:
TALHTTPResponseHeader2;const ARequestHeaderValues : TALNameValueArray);
27612:     Function Delete1(const aURL:AnsiString;const ARequestHeaderValues: TALNameValueArray): AnsiString;
27613:     Procedure Options(const aUrl:AnsiString;const aResponseContent:TStream;const aResponseHeader:
TALHTTPResponseHeader2;const ARequestHeaderValues:TALNameValueArray);
27614:     Function Options1(const aURL:AnsiString; const ARequestHeaderValues:TALNameValueArray):AnsiString;
27615:     RegisterProperty('ConnectTimeout', 'Integer', iptrw);
27616:     SendTimeout, 'Integer', iptrw);
27617:     ReceiveTimeout, 'Integer', iptrw);
27618:     UploadBufferSize, 'cardinal', iptrw);
27619:     ProxyParams, 'TALHTTPClientProxyParams', iptr);
27620:     RequestHeader, 'TALHTTPRequestHeader2', iptr);
27621:     ProtocolVersion, 'TALHTTPProtocolVersion', iptrw);
27622:     UserName, 'AnsiString', iptrw);
27623:     Password, 'AnsiString', iptrw);
27624:     OnUploadProgress, 'TALHTTPClientUploadProgressEvent', iptrw);
27625:     OnDownloadProgress, 'TALHTTPClientDownloadProgressEvent', iptrw);
27626:     OnRedirect, 'TALHTTPClientRedirectEvent', iptrw);
27627:   end;
27628: end;
27629:
27630: procedure SIRegister_ALHttpclient2(CL: TPSPascalCompiler);
27631: begin
27632:   TALHTTPPropertyChangeEvent', 'Procedure ( sender: TObject; const PropertyIndex: Integer)';
27633:   TALHTTPProtocolVersion', '( HTTPPv_1_0, HTTPPv_1_1 )';
27634:   TALHTTPMethod2', '( HTTPmt_Get, HTTPmt_Post, HTTPmt_Head, HTTPmt_
27635:   '+Trace, HTTPmt_Put, HTTPmt_Delete, HTTPmt_Options )';
27636:   SIRegister_TALHTTPRequestHeader2(CL);
27637:   TALNameValuePair', 'record Name : ansistring; Value : ansistring; end';
27638:   TALNameValueArray', 'array of TALNameValuePair';
27639:   SIRegister_TALHTTPCookie2(CL);
27640:   SIRegister_TALHTTPCookieCollection2(CL);
27641:   SIRegister_TALHTTPResponseHeader2(CL);
27642:   SIRegister_EALHTTPClientException2(CL);
27643:   SIRegister_TALHTTPClientProxyParams2(CL);
27644:   TALHTTPClientRedirectEvent', 'Procedure ( sender : TObject; const NewURL : AnsiString)';
27645:   TALHTTPClientUploadProgressEvent', 'Procedure ( sender : TObject; Sent: Integer; Total: Integer)';
27646:   TALHTTPClientDownloadProgressEvent', 'Procedure ( sender: TObject; Read: Integer; Total: Integer)';
27647:   SIRegister_TALHTTPClient2(CL);
27648:   Procedure ALHTTPEncodeParamNameValues2( const ParamValues : TALStrings);
27649:   //Procedure ALExtractHTTPFields( Separators,WhiteSpace,Quotes: TSysCharSet; Content:PAnsiChar; Strings :
TALStrings; StripQuotes : Boolean);
27650:   Function ALRemoveSchemeFromUrl2( const aUrl : AnsiString) : ansistring;
27651:   Function ALExtractSchemeFromUrl2( const aUrl : AnsiString) : TInternetScheme;
27652:   Function ALExtractHostNameFromUrl2( const aUrl : AnsiString) : AnsiString;
27653:   Function ALExtractDomainNameFromUrl2( const aUrl : AnsiString) : AnsiString;
27654:   Function ALExtractUrlPathFromUrl2( const aUrl : AnsiString) : AnsiString;
27655:   Function ALInternetCrackUrl2(const aUrl:AnsiString;var SchemeName,HostName,UserName,Password,UrlPath,
ExtraInfo : AnsiString;var PortNumber:integer):Bool;

```

```

27656: Function AlInternetCrackUrl12(const aUrl:AnsiString;var SchemeName,HostName,UserName,Password,UrlPath,
Anchor: AnsiString;const Query:TALStrings;var PortNumber:integer): Boolean;
27657: Function AlInternetCrackUrl22( var Url:AnsiString;var Anchor:AnsiString;const Query:TALStrings):
Boolean;');
27658: Function AlRemoveAnchorFromUrl2( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;');
27659: Function AlRemoveAnchorFromUrl12( const aUrl : AnsiString) : AnsiString;');
27660: Function AlCombineUrl2( const RelativeUrl, BaseUrl : AnsiString) : AnsiString;');
27661: Function AlCombineUrl12( const RelativeUrl,BaseUrl,Anchor AnsiString;const Query:TALStrings):
AnsiString;');
27662: Function ALGmtDateTimeToRfc822Str2( const aValue : TDateTime) : AnsiString;');
27663: Function ALDateTimeToRfc822Str2( const aValue : TDateTime) : AnsiString;');
27664: Function ALTryRfc822StrToGMTDateTime2( const S : AnsiString; out Value : TDateTime) : Boolean;');
27665: Function ALRfc822StrToGMTDateTime2( const s : AnsiString) : TDateTime;');
27666: Function ALTryIPv4StrToNumeric2( const aIPv4Str : AnsiString; var aIPv4Num : Cardinal) : Boolean;');
27667: Function ALIPv4StrToNumeric2( const aIPv4 : AnsiString) : Cardinal;');
27668: Function ALNumericToIPv4Str2( const aIPv4 : Cardinal) : AnsiString;');
27669: Function ALIPv4EndOfRange2( const aStartIPv4 : Cardinal; aMaskLength : integer) : Cardinal;');
27670: Function ALZeroIPv62 : TALIPv6Binary;');
27671: Function ALIsValidIPv6BinaryStr2( const aIPv6BinaryStr : AnsiString) : boolean;');
27672: Function ALTryIPv6StrToBinary2( aIPv6Str : AnsiString; var aIPv6Bin : TALIPv6Binary) : Boolean;');
27673: Function ALIPv6StrToBinary2( const aIPv6 : AnsiString) : TALIPv6Binary;');
27674: Function ALBinaryToIPv6Str2( const aIPv6 : TALIPv6Binary) : AnsiString;');
27675: Function ALBinaryStrToIPv6Binary2( const aIPv6BinaryStr : AnsiString) : TALIPv6Binary;');
27676: Function ALBinaryStrToIPv6Str2( const aIPv6BinaryStr : AnsiString) : AnsiString;');
27677: Function ALIPv6EndOfRange2( const aStartIPv6 : TALIPv6Binary; aMaskLength : integer) : TALIPv6Binary;');
27678: Procedure ALIPv6SplitParts2(const aIPv6: TALIPv6Binary;var aLowestPart:UInt64; var aHigestPart: UInt64);');
27679: CL.AddConstantN('cALHTTIClient_MsgInvalidURL2','String').SetString('Invalid url '%s' - only supports
'http' and 'https' schemes');
27680: CL.AddConstantN('cALHTTIClient_MsgInvalidHTTPRequest2','String').SetString('Invalid HTTP Request:Length
is 0');
27681: CL.AddConstantN('cALHTTIClient_MsgEmptyURL2','String').SetString('Empty URL');
27682: function ALHTTPEncode(const AStr: AnsiString): AnsiString;');
27683: function ALHTTDecode(const AStr: AnsiString): AnsiString;');
27684: end;
27685:
27686: CL.AddDelphiFunction('procedure RunDosInMemo(DosApp: string; AMemo:TMemo);');
27687:
27688: {type TTextHandler =} procedure TTextHandlerQ(const aText: string);
27689: begin
27690:     memo2.lines.add(aText);
27691: end;
27692: writeln(itoa(JExecute('cmd /C dir *.*',@TTextHandlerQ, true, false)););
27693: writeln(itoa(JExecute3('cmd /C dir *.*',Nil, true, false)););
27694:
27695: memo2.lines.add(getIPs.text);
27696: srlist:= TStringlist.create;
27697: GetDNSServers(srlist)
27698: writeln(srlist.text)
27699: srlist.Free;
27700:
27701: https://presse.tuv.com/en/photovoltaics-tuv-rheinland-investigates-potential-for-applications-on-rail-
infrastructure/
27702:
27703: procedure SIRegister_TRestUtils(CL: TPSPascalCompiler);
27704: begin
27705:     //with RegClassS(CL,'OBJECT', 'TRestUtils') do
27706:     with CL.AddClassN(CL.FindClass('OBJECT'),'TRestUtils') do begin
27707:         Function Base64Encode( const AValue : String) : String;');
27708:         Function Base64Decode( const AValue : String) : String;');
27709:     end;
27710: end;
27711:
27712: (*-----*)
27713: procedure SIRegister_TStatusCode(CL: TPSPascalCompiler);
27714: begin
27715:     //with RegClassS(CL,'OBJECT', 'TStatusCode') do
27716:     with CL.AddClassN(CL.FindClass('OBJECT'),'TStatusCode') do begin
27717:         Function CONTINUE : TReponseCode;');
27718:         Function SWITCHING_PROTOCOLS : TReponseCode;');
27719:         Function OK : TReponseCode;');
27720:         Function CREATED : TReponseCode;');
27721:         Function ACCEPTED : TReponseCode;');
27722:         Function NON_AUTHORITATIVE_INFORMATION : TReponseCode;');
27723:         Function NO_CONTENT : TReponseCode;');
27724:         Function RESET_CONTENT : TReponseCode;');
27725:         Function PARTIAL_CONTENT : TReponseCode;');
27726:         Function MULTIPLE_CHOICES : TReponseCode;');
27727:         Function MOVED_PERMANENTLY : TReponseCode;');
27728:         Function FOUND : TReponseCode;');
27729:         Function SEE_OTHER : TReponseCode;');
27730:         Function NOT_MODIFIED : TReponseCode;');
27731:         Function USE_PROXY : TReponseCode;');
27732:         Function TEMPORARY_REDIRECT : TReponseCode;');
27733:         Function BAD_REQUEST : TReponseCode;');
27734:         Function UNAUTHORIZED : TReponseCode;');
27735:         Function PAYMENT_REQUIRED : TReponseCode;');
27736:         Function FORBIDDEN : TReponseCode;');
27737:         Function NOT_FOUND : TReponseCode;');
27738:         Function METHOD_NOT_ALLOWED : TReponseCode;');

```

```

27739:   Function NOT_ACCEPTABLE : TReponseCode');
27740:   Function PROXY_AUTHENTICATION_REQUIRED : TReponseCode');
27741:   Function REQUEST_TIMEOUT : TReponseCode');
27742:   Function CONFLICT : TReponseCode');
27743:   Function GONE : TReponseCode');
27744:   Function LENGTH_REQUIRED : TReponseCode');
27745:   Function PRECONDITION_FAILED : TReponseCode');
27746:   Function REQUEST_ENTITY_TOO_LARGE : TReponseCode');
27747:   Function REQUEST_URI_TOO_LONG : TReponseCode');
27748:   Function UNSUPPORTED_MEDIA_TYPE : TReponseCode');
27749:   Function REQUESTED_RANGE_NOT_SATISFIABLE : TReponseCode');
27750:   Function EXPECTATION_FAILED : TReponseCode');
27751:   Function UNPROCESSABLE_ENTITY : TReponseCode');
27752:   Function INTERNAL_SERVER_ERROR : TReponseCode');
27753:   Function NOT_IMPLEMENTED : TReponseCode');
27754:   Function BAD_GATEWAY : TReponseCode');
27755:   Function SERVICE_UNAVAILABLE : TReponseCode');
27756:   Function GATEWAY_TIMEOUT : TReponseCode');
27757:   Function HTTP_VERSION_NOT_SUPPORTED : TReponseCode');
27758: end;
27759: end;
27760:
27761: (*-----*)
27762: procedure SIRegister_RestUtils(CL: TPSPascalCompiler);
27763: begin
27764:   MediaType_Json('String').SetString('application/json');
27765:   MediaType_Xml('String').SetString('text/xml');
27766:   LOCALE_PORTUGUESE_BRAZILIAN('String').SetString('pt-BR');
27767:   LOCALE_US('String').SetString('en-US');
27768:   CL.AddTypeS('TReponseCode', 'record StatusCode : Integer; Reason : string; end');
27769:   SIRegister_TStatusCode(CL);
27770:   SIRegister_TRestUtils(CL);
27771: end;
27772:
27773: procedure SIRegister_DataSetConverter4DUtil(CL: TPSPascalCompiler);
27774: begin
27775:   CL.AddTypeS('TBooleanFieldType', '( bftUnknown, bftBoolean, bftInteger )');
27776:   TDataSetFieldType(' ( dftUnknown, dftJSONObject, dftJSONArray )');
27777:   CL.AddDelphiFunction('Function DateTimeToISOTimeStamp( const dateTime : TDateTime) : string');
27778:   Function DateTimeToISODate( const date : TDateTime) : string;
27779:   Function TimeToISOTime( const time : TTime) : string;
27780:   Function ISOTimeStampToDateTime( const dateTime : string) : TDateTime;
27781:   Function ISODateToDate( const date : string) : TDate;
27782:   Function ISOTimeToTime( const time : string) : TTime;
27783:   Function NewDataSetField(dataSet:TDataSet;const fieldType:TFieldType;const fieldName:string;const
size:Integer; const origin:string;const displaylabel:string): TField;
27784:   Function BooleanToJSON( const value : Boolean) : TJSONValue;
27785:   Function BooleanFieldToType( const booleanField:TBooleanField): TBooleanFieldType;
27786:   Function DataSetFieldToType( const dataSetField:TDataSetField): TDataSetFieldType;
27787:   Function MakeValidIdent( const s : string) : string;
27788: end;
27789:
27790: function wget3(aURL, afile: string; opendoc: boolean): boolean;
27791: procedure RegisterProtocol(const Name, Description, ExecuteStr: string);
27792: procedure UnregisterProtocol(const Name: string);
27793: function GetNetworkConnections: string;
27794: function GetNetworkDrives: string;
27795:
27796: wget3('https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Ffoodanddrink.scotsman.com%2Fwp-
content%2Fuploads%2F2016%2F08%2Fitem-4-9-1024x683.jpg',
exepath+'examples/downpic.png', true);
27797:
27798:
27799: procedure SIRegister_PSResources(CL: TPSPascalCompiler);
2800: begin
28001:   CL.AddConstantN('sBeginCompile','String').SetString('Compiling');
28002:   ('sSuccessfullyCompiled','String').SetString('Successfully compiled');
28003:   ('sSuccessfullyExecuted','String').SetString('Successfully executed');
28004:   ('sRuntimeError','String').SetString('[Runtime error] %s(%d:%d), bytecode(%d:%d) : %s');
28005:   ('sTextNotFound','String').SetString('Text not found');
28006:   ('sUnnamed','String').SetString('Unnamed');
28007:   ('sEditorTitle','String').SetString('Editor');
28008:   ('sEditorTitleRunning','String').SetString('Editor - Running');
28009:   ('sEditorTitlePaused','String').SetString('Editor - Paused');
28010:   ('sEditorTitleStopped','String').SetString('Editor - Stopped');
28011:   ('sInputBoxTitle','String').SetString('Script');
28012:   ('sFileNotSaved','String').SetString('File has not been saved, save now?');
28013:   ('isRunningOrPaused','LongInt').Value.ts32 := ord(isRunning) or ord(isPaused);
28014:   SIRegister_TPSUtils(CL);
28015:   //https://github.com/fabriciocolombo/PascalScriptEx/tree/master/src
28016:   CL.AddDelphiFunction('function list_functions: TStringlist;');
28017:   function list_functions2: TStringlist;
28018:   procedure InternalRaiseException(E: Exception);
28019:   function ListFiles(ADirectory, AFilter: String; AList: TStringList): Boolean;
28020:   Procedure CommandToList(S : String; List : TStringList);
28021: end;
28022:
28023: CL.AddDelphiFunction('function list_functions: TStringlist;');
28024: procedure InternalRaiseException(E: Exception);
28025: function ListFiles(ADirectory, AFilter: String; AList: TStringList) : Boolean;

```

```

27826: Procedure CommandToList(S : String; List : TStrings);
27827: CL.AddDelphiFunction('procedure SetSynchroTime');
27828: CL.AddDelphiFunction('procedure letSynchroTime');
27829: procedure TimeSync'; //get internettime service
27830: function HttpGetDirect(const Url: string): string;
27831: function HttpGetDirect2(const Url: string): string;
27832: function HttpGetDirect4(const Url, agent: string): string;
27833: function HttpGet_Direct(const Url: string): string;');
27834: function wininet_HttpGetDirect(const Url: string): string;');
27835: function GetWinInetError(ErrorCode: Cardinal): string;
27836: function HttpGetDirect3(const ServerName, Resource, sUserAgent: string; Var Response: AnsiString): Integer;
27837: procedure WinInetCheck(Success: Boolean; aFunction: PChar);
27838: function WebGetData(const UserAgent: string; const Server: string; const Resource: string): string;
27839: procedure WebPostData(const UserAgent: string; const Server: string;
27840:   const Resource: string; const Data: AnsiString);
27841: function WebPostData2(const UserAgent: string; const Server: string; const Resource: string;
27842:   const Data: AnsiString): string;
27843: function URLEncode3(const Url: string): string;
27844:
27845: function FloatAsInteger(X: single): integer;
27846: begin
27847:   result:= PInteger(@X)^;
27848: end;
27849: function IntegerAsFloat(X: integer): single;
27850: begin
27851:   result:= PSingle(@X)^;
27852: end;
27853:
27854: procedure ParseString4(s, sep: string; sl: TStrings););
27855: procedure LoadGridFromFile(SG: TStringGrid; fname: string);
27856: procedure SaveGridToFile(SG: TStringGrid; fname: string);
27857: procedure WriteVectorToGrid(SG: TStringGrid; vname: string; wval: TAffineVector);
27858: procedure WriteStringToGrid(SG: TStringGrid; vname: string; icol: longword; wval: string);
27859: procedure WriteFloatToGrid(SG: TStringGrid; vname: string; icol: longword; wval: double);
27860: https://github.com/P33a/SimTwo/blob/master/Utils.pas
27861: procedure ExtractIconFileToImageList(ImageList: TImageList; const Filename: string);
27862:
27863: procedure SIRegister_SynCrtSock(CL: TPSPascalCompiler); -->SIRegister_synTHttpRequest
27864: begin
27865:   CL.AddConstantN('HTTP_RESP_STATICFILE', 'String').SetString( '!STATICFILE');
27866:   'HTTP_RESP_NORESPONSE', 'String').SetString( '!NORESPONSE');
27867:   'HTTP_DEFAULT_RESOLVETIMEOUT', 'LongInt').SetInt( 0);
27868:   'HTTP_DEFAULT_CONNECTTIMEOUT', 'LongInt').SetInt( 60000);
27869:   'HTTP_DEFAULT_SENDDTIMEOUT', 'LongInt').SetInt( 30000);
27870:   'HTTP_DEFAULT_RECEIVETIMEOUT', 'LongInt').SetInt( 30000);
27871:   CL.AddTypeS('SynUnicode', 'UnicodeString');
27872:   'SockString', 'RawByteString');
27873:   'SynUnicode', 'WideString');
27874:   'SockString', 'AnsiString');
27875:   S('PPointer', '^Pointer // will not work');
27876:   'synTTextLineBreakStyle', '( tlbsLF, tlbsCRLF)');
27877:   'synUTF8String', 'AnsiString');
27878:   'synUTF8Encode', 'AnsiString');
27879:   'synPtrInt', 'NativeInt');
27880:   'synPtrUInt', 'NativeUInt');
27881:   'synPtrInt', 'integer');
27882:   'synPtrUInt', 'cardinal');
27883:   THttpSocketCompress', 'function(var DataRawByteString: sockstring; Compress: boolean): AnsiString');
27884:   //CL.AddTypeS('PPtrInt', '^PtrInt // will not work');
27885:   //CL.AddTypeS('PPtrUInt', '^PtrUInt // will not work');
27886:   CL.AddTypeS('TSockHeaders', 'array of SockString');
27887:   SIRegister_ECrtSocket(CL);
27888:   //CL.AddTypeS('TCrtSocketClass', 'class of TCrtSocket');
27889:   CL.AddTypeS('TCrtSocketLayer', '( cslTCP, cslUDP, cslUNIX)');
27890:   CL.AddTypeS('TCrtSocketPending', '( cspSocketError, cspNoData, cspDataAvailable)');
27891:   //CL.AddTypeS('PTextFile', '^TextFile // will not work');
27892:   SIRegister_TCrtSocket(CL);
27893:   CL.AddTypeS('THttpSocketCompressRec', 'record Name : SockString; Func : THttp'
27894:     + 'SocketCompress; CompressMinSize : integer; end');
27895:   CL.AddTypeS('THttpSocketCompressRecDynArray', 'array of THttpSocketCompressRec');
27896:   CL.AddTypeS('THttpSocketCompressSet', 'Integer');
27897:   SIRegister_THttpSocket(CL);
27898:   CL.AddClassN(CL.FindClass('TOBJECT'), 'THttpServer');
27899:   SIRegister_THttpServerSocket(CL);
27900:   SIRegister_THttpClientSocket(CL);
27901:   //CL.AddTypeS('THttpClientSocketClass', 'class of THttpClientSocket');
27902:   CL.AddTypeS('TNotifyThreadEvent', 'Procedure ( Sender : TThread)');
27903:   SIRegister_TSynThread(CL);
27904:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TSynThreadPoolTHttpServer');
27905:   SIRegister_THttpServerResp(CL);
27906:   //CL.AddTypeS('THttpServerRespClass', 'class of THttpServerResp');
27907:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TSynThreadPool');
27908:   SIRegister_TSynThreadPoolSubThread(CL);
27909:   SIRegister_TSynThreadPool(CL);
27910:   SIRegister_TSynThreadPoolTHttpServer(CL);
27911:   CL.AddClassN(CL.FindClass('TOBJECT'), 'THttpServerGeneric');
27912:   CL.AddTypeS('THttpServerRequestAuthentication', '( hraNone, hraFailed, hraBas'
27913:     + 'ic, hraDigest, hraNtlm, hraNegotiate, hraKerberos)');
27914:   SIRegister_THttpServerRequest(CL);

```



```

27915: CL.AddTypeS('TOnHttpRequest','Function (Ctxt : THttpRequest):cardinal');
27916: SIRegister_THttpServerGeneric(CL);
27917: CL.AddTypeS('synULONGLONG','Int64');
27918: 'HTTP_OPAQUE_ID', 'ULONGLONG');
27919: 'HTTP_URL_GROUP_ID', 'HTTP_OPAQUE_ID');
27920: 'HTTP_SERVER_SESSION_ID', 'HTTP_OPAQUE_ID');
27921: 'THttpApiLoggingType', '( hltW3C, hltIIS, hltNCSA, hltRaw )');
27922: 'THttpApiLoggingRollOver', '( hlrSize, hlrDaily, hlrMonthly, hlrHourly )');
27923: 'THttpApiLoggingFlag', '( hlfLocalTimeRollOver, hlfUseUTF'
27924: + '8Conversion, hlfLogErrorsOnly, hlfLogSuccessOnly )');
27925: CL.AddTypeS('THttpApiLoggingFlags', 'set of THttpApiLoggingFlag');
27926: CL.AddTypeS('THttpApiLogField', '( hlfDate, hlfTime, hlfClientIP, hlf'
27927: + 'UserName, hlfSiteName, hlfComputerName, hlfServerIP, hlfMethod, hlfURIStem'
27928: + 'hlfURIQuery, hlfStatus, hlfWIN32Status, hlfBytesSent, hlfBytesRecv, hlfT'
27929: + 'imeTaken, hlfServerPort, hlfUserAgent, hlfCookie, hlfReferer, hlfVersion, hlfHost, hlfSubStatus )');
27930: 'THttpApiLogFields', 'set of THttpApiLogField');
27931: ('THttpApiRequestAuthentication', '( haBasic, haDigest, haNtlm, haNegotiate, haKerberos )');
27932: ('THttpApiRequestAuthentications', 'set of THttpApiRequestAuthentication');
27933: SIRegister_THttpApiServer(CL);
27934: SIRegister_THttpServer(CL);
27935: SIRegister_synTURI(CL);
27936: CL.AddTypeS('THttpRequestAuthentication', '( wraNone, wraBasic, wraDigest, wraNegotiate )');
27937: CL.AddTypeS('THttpRequestExtendedOptions', 'record IgnoreSSLCertificateErrors'
27938: + ' : Boolean; Auth : record UserName : SynUnicode; Password : SynUnicode; Sc'
27939: + 'heme : THttpRequestAuthentication; end ; end');
27940: {$IFDEF SIRegister_synTHttpRequest(CL); }
27941: //CL.AddTypeS('THttpRequestClass', 'class of THttpRequest');
27942: SIRegister_TWinHttpAPI(CL);
27943: SIRegister_TWinINet(CL);
27944: SIRegister_EWinINet(CL);
27945: SIRegister_TWinHTTP(CL);
27946: CL.AddClassN(CL.FindClass('TOBJECT'),'EWinHTTP');
27947: SIRegister_TCurlHTTP(CL);
27948: Function synOpenCrtSock(const aServer, aPort : SockString) : TCrtSocket';
27949: Function synOpenHttp(const aServer, aPort : SockString) : THttpClientSocket';
27950: Function synHttpGet(const server, port : SockString; const url : SockString) : SockString;';
27951: Function synHttpGet1(const aURI : SockString) : SockString;';
27952: Function synHttpPost(const server, port : SockString; const url, Data, DataType : SockString) : boolean;';
27953: 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;
27954: Function synSendEmailSubject( const Text : string) : SockString';
27955: 'STATUS_SUCCESS','LongInt').SetInt( 200);
27956: 'STATUS_CREATED','LongInt').SetInt( 201);
27957: 'STATUS_NOCONTENT','LongInt').SetInt( 204);
27958: 'STATUS_BADREQUEST','LongInt').SetInt( 400);
27959: 'STATUS_UNAUTHORIZED','LongInt').SetInt( 401);
27960: 'STATUS_FORBIDDEN','LongInt').SetInt( 403);
27961: 'STATUS_NOTFOUND','LongInt').SetInt( 404);
27962: 'STATUS_SERVERERROR','LongInt').SetInt( 500);
27963: 'STATUS_NOTIMPLEMENTED','LongInt').SetInt( 501);
27964: Function synStatusCodeToReason( Code : integer) : SockString';
27965: Function synResolveName(const Name:SockString;Family:Integer;SockProtocol:Int;SockType:int):SockString';
27966: Function synBase64Encode( const s : SockString) : SockString';
27967: Function synBase64Decode( const s : SockString) : SockString';
27968: Function synHtmlEncode( const s : SockString) : SockString';
27969: Function synGetRemoteMacAddress(const IP : SockString) : SockString';
27970: end;
27971:
27972: procedure SIRegister_THttpConnectionWinInet(CL: TPSPascalCompiler);
27973: begin
27974: //with RegClassS(CL,'TInterfacedObject', 'THttpConnectionWinInet') do
27975: with CL.AddClassN(CL.FindClass('TInterfacedObject'),'THttpConnectionWinInet') do begin
27976: OnConnectionLost, 'THttpConnectionLostEvent', iptrw);
27977: Constructor Create( ARaiseExceptionOnError : Boolean);
27978: Procedure Free;);
27979: Function SetAcceptTypes( AAcceptTypes : string) : IHttpConnection';
27980: Function SetAcceptedLanguages( AAcceptedLanguages : string) : IHttpConnection';
27981: Function SetContentTypes( AContentTypes : string) : IHttpConnection';
27982: Function SetHeaders( AHeaders : TStrings):IHttpConnection';
27983: Headers, 'TStrings', iptrw);
27984: Procedure Get( AUrl : string; AResponse : TStream);
27985: Procedure Post( AUrl : string; AContent : TStream; AResponse : TStream);
27986: Procedure Put( AUrl : string; AContent : TStream; AResponse : TStream);
27987: Procedure Patch( AUrl : string; AContent : TStream; AResponse : TStream);
27988: Procedure Delete( AUrl : string; AContent : TStream; AResponse : TStream);
27989: Function GetResponseCode : Integer);
27990: Function GetResponseHeader( const Header : string) : string';
27991: Function GetEnabledCompression : Boolean);
27992: Procedure SetEnabledCompression( const Value : Boolean);
27993: Function GetOnConnectionLost : THttpConnectionLostEvent';
27994: Procedure SetOnConnectionLost( AConnectionLostEvent : THttpConnectionLostEvent');
27995: Function ConfigureTimeout( const ATimeout : TTimeOut) : IHttpConnection';
27996: Function ConfigureProxyCredentials( AProxyCredentials : TProxyCredentials) : IHttpConnection';
27997: Procedure SetVerifyCert( const Value : boolean);
27998: Function GetVerifyCert : boolean);
27999: Function SetAsync( const Value : Boolean) : IHttpConnection';
28000: Procedure CancelRequest';
28001: Function SetOnAsyncRequestProcess( const Value : TAsyncRequestProcessEvent) : IHttpConnection';
28002: RegisterProperty('ResponseErrorStatusText', 'string', iptrw);

```



```

28003: CertificateContext', 'PCERT_CONTEXT', iptrw);
28004: 'BasicAuthentication_UserName', 'string', iptrw);
28005: 'BasicAuthentication_Password', 'string', iptrw);
28006: 'CertificateCheckDate', 'Boolean', iptrw);
28007: 'CertificateCheckHostName', 'boolean', iptrw);
28008: 'CertificateCheckAuthority', 'boolean', iptrw);
28009: 'CertificateIgnoreRevocation', 'boolean', iptrw);
28010: 'RaiseExceptionOnError', 'Boolean', iptrw);
28011: end;
28012: end;
28013:
28014: procedure SIRegister_VelthuisFloatUtils(CL: TPSPascalCompiler);
28015: begin
28016:   Function IsNegativeInfinity( const AValue : Single) : Boolean;);
28017:   Function IsNegativeInfinity1( const AValue : Double) : Boolean;);
28018:   Function IsNegativeInfinity2( const AValue : Extended) : Boolean;);
28019:   Function IsPositiveInfinity( const AValue : Single) : Boolean;);
28020:   Function IsPositiveInfinity1( const AValue : Double) : Boolean;);
28021:   Function IsPositiveInfinity2( const AValue : Extended) : Boolean;);
28022:   Function GetSignificand( const AValue : Single) : UInt32;);
28023:   Function GetSignificand1( const AValue : Double) : UInt64;);
28024:   Function GetSignificand2( const AValue : Extended) : UInt64;);
28025:   Function GetMantissa( const AValue : Single) : UInt32;);
28026:   Function GetMantissa1( const AValue : Double) : UInt64;);
28027:   Function GetMantissa2( const AValue : Extended) : UInt64;);
28028:   Function GetExponent( const AValue : Single) : Integer;);
28029:   Function GetExponent1( const AValue : Double) : Integer;);
28030:   Function GetExponent2( const AValue : Extended) : Integer;);
28031:   Function IsDenormal( const AValue : Single) : Boolean;);
28032:   Function IsDenormal1( const AValue : Double) : Boolean;);
28033:   Function IsDenormal2( const AValue : Extended) : Boolean;);
28034:   Function MakeSingle( Sign : byte; Significand: Longword; Exponent: Integer): Single;);
28035:   Function MakeDouble( Sign : byte; Significand: UInt64; Exponent: Integer): Double;);
28036:   Function MakeExtended( Sign : byte; Significand: UInt64; Exponent: Integer): Extended;);
28037:   //CL.AddTypeS('PUInt8', '^UInt8 // will not work');
28038:   //CL.AddTypeS('PUInt16', '^UInt16 // will not work');
28039:   //CL.AddTypeS('PUInt32', '^UInt32 // will not work');
28040:   //CL.AddTypeS('PExt80Rec', '^TExt80Rec // will not work');
28041:   CL.AddTypeS('TExt80Rec', 'record Significand : UInt64; ExponentAndSign : Word; end;');
28042:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work'); '+'; end;);
28043:   ('CSingleExponentShift', 'LongInt').SetInt( 23);
28044:   ('CDoubleExponentShift', 'LongInt').SetInt( 52);
28045:   ('CSingleExponentMask', 'LongWord').SetUInt( $FF);
28046:   ('CDoubleExponentMask', 'LongWord').SetUInt( $7FF);
28047:   ('CExtendedExponentMask', 'LongWord').SetUInt( $7FFF);
28048:   ('CSingleBias', 'LongInt').SetInt( CSingleExponentMask shr 1);
28049:   ('CDoubleBias', 'LongInt').SetInt( CDoubleExponentMask shr 1);
28050:   ('CExtendedBias', 'LongInt').SetInt( CExtendedExponentMask shr 1);
28051:   ('CSingleSignMask', 'LongInt').SetInt( UInt32( 1 ) shl 31);
28052:   ('CDoubleSignMask', 'LongInt').SetInt( UInt64( 1 ) shl 63);
28053: end;
28054:
28055: function BytesToHex2(const Bytes: TBytes): string;
28056: {Creates a hexadecimal representation of the bytes in an array.
28057:   @param Bytes [in] Byte array.
28058:   return Hexadecimal representation of bytes in array. }
28059: function TryHexToBytes(HexStr: string; out Bytes: TBytes): Boolean; *) UHexUtils.pas
28060:
28061: CL.AddTypeS('TExeFileKind', '( fkUnknown, fkError, fkDOS, fkExe32, fkExe16, fkDLL32, fkDLL16, fkVXD )');
28062: Function ExeFileType( const FileName : string) : TExeFileKind;
28063: Function getExeFileType( const FileName : string) : TExeFileKind;
28064: function ExeFileType(const FileName: string): TExeFileKind; UExeFileType.pas
28065: function ListFiles2(const Dir, Wildcard: string; const List: TStringList;
28066:   IncludeDirs: Boolean = True; RelativeNames: Boolean = False): Boolean;
28067:
28068: Function LongToShortFilePath( const LongName : string) : string;
28069: //Function IsDirectory( const DirName : string) : Boolean;
28070: Function FloatToInt( const F : Double) : Int64;
28071: Function RFC1123TimeStamp : string;
28072: Function NowGMT : TDateTime;
28073: Function ParseSQLDateTime( const SQLDateTime : string) : TDateTime;
28074: Function TryParseSQLDateTime( const SQLDateTime : string; out Value : TDateTime) : Boolean;
28075: Procedure GetInterface( const Instance : IInterface; const IID : TGUID; out Intf);
28076: Function IsBaseFileName( const FileName : string) : Boolean;
28077: Procedure SetPause( const ADelay : Cardinal);
28078: Procedure letPause( const ADelay : Cardinal);
28079: Function URIBaseName( const URI : string) : string;
28080: Procedure StrSliceRight(URI, Length(URI) - LastSlashPos);
28081: Function CharInSet4(C: Char; const CharSet: CharSet): Bool;
28082: Function TryStrToCardinal( const S : string; out Value : Cardinal) : Boolean;
28083: Function IsEqualBytes0( const BA1, BA2: TBytes; const Count: Cardinal) : Boolean;);
28084: Function IsEqualBytes1( const BA1, BA2: TBytes) : Boolean;);
28085: Function TryStrToWord( const S : string; out W : Word) : Boolean;);
28086: function IsHexDigit(C: Char): Boolean;
28087:
28088: * RIRegister_TCompilerRunner Implements a class that executes a compiler and captures its output and exit
28089: * code. Also provides specialised exception object that captures information
28090: * about errors. depends on TConsoleApp Class
28091:

```

```

28092: procedure SIRegister_TCompilerRunner(CL: TPSPascalCompiler);
28093: begin
28094:   //with RegClassS(CL, 'TObject', 'TCompilerRunner') do
28095:   with CL.AddClassN(CL.FindClass('TObject'), 'TCompilerRunner') do begin
28096:     Function Execute( const CommandLine, CurrentPath: string; const OutStream: TStream) : LongWord';
28097:   end;
28098: end;
28099: hStream:= TStringStream.Create('');
28100: with TCompilerRunner.Create do begin
28101:   //writeln(itoa(errorcode));
28102:   Execute('cmd /c dir *.*', exepath, hstream)
28103:   writeln(hstream.datastring);
28104:   free
28105: end;
28106:
28107: (*-----*)
28108: procedure SIRegister_TConsoleApp(CL: TPSPascalCompiler);
28109: begin
28110:   //with RegClassS(CL, 'TObject', 'TConsoleApp') do
28111:   with CL.AddClassN(CL.FindClass('TObject'), 'TConsoleApp') do begin
28112:     Constructor Create('');
28113:     Function Execute( const CmdLine, CurrentDir : string) : Boolean';
28114:     StdIn', 'THandle', iptrw);
28115:     StdOut', 'THandle', iptrw);
28116:     StdErr', 'THandle', iptrw);
28117:     Visible', 'Boolean', iptrw);
28118:     MaxExecTime', 'LongWord', iptrw);
28119:     TimeSlice', 'Integer', iptrw);
28120:     ProcessHandle', 'THandle', iptr);
28121:     ExitCode', 'LongWord', iptr);
28122:     ErrorCode', 'DWORD', iptr);
28123:     ErrorMessage', 'string', iptr);
28124:     OnWork', 'TNotifyEvent', iptrw);
28125:   end;
28126: end;
28127:
28128: procedure SIRegister_TPipe(CL: TPSPascalCompiler);
28129: begin
28130:   //with RegClassS(CL, 'TObject', 'TPipe') do
28131:   with CL.AddClassN(CL.FindClass('TObject'), 'TPipe') do begin
28132:     Constructor Create( const Size : LongWord');
28133:     Procedure Free('');
28134:     Function AvailableDataSize : LongWord';
28135:     Function ReadData( var Buf : string; const BufSize : LongWord; var BytesRead : LongWord) : Boolean';
28136:     Procedure CopyToStream( const Stm : TStream; Count : LongWord');
28137:     Procedure CopyFromStream( const Stm : TStream; Count : LongWord');
28138:     Function WriteData( const Buf : string; const BufSize : LongWord) : LongWord';
28139:     Procedure CloseWriteHandle';
28140:     RegisterProperty('ReadHandle', 'THandle', iptr);
28141:     WriteHandle', 'THandle', iptr);
28142:   end;
28143: end;
28144:
28145: Exception Hierarchy
28146: CL.AddClassN(CL.FindClass('TObject'), 'TClass');
28147: CL.AddClassN(CL.FindClass('Exception'), 'EAbort');
28148: SIRegister_EHeapException(CL);
28149: CL.AddClassN(CL.FindClass('EHeapException'), 'EOutOfMemory');
28150: SIRegister_EInOutError(CL);
28151: //CL.AddTypeS('PEExceptionRecord', '^TExceptionRecord // will not work');
28152: SIRegister_EExternal(CL);
28153:   EIntError = class(EExternal);
28154:     EDivByZero = class(EIntError);
28155:     ERangeError = class(EIntError);
28156:     EIntOverflow = class(EIntError);
28157:   EMathError = class(EExternal);
28158:     EInvalidOp = class(EMathError);
28159:     EZeroDivide = class(EMathError);
28160:     EOverflow = class(EMathError);
28161:     EUnderflow = class(EMathError);
28162:     EInvalidPointer = class(EHeapException);
28163:     EInvalidCast = class(Exception);
28164:     EConvertError = class(Exception);
28165:     EAccessViolation = class(EExternal);
28166:     EPrivilege = class(EExternal);
28167:     EStackOverflow = class(EExternal)
28168: end deprecated;
28169:
28170: procedure SIRegister_IHttpConnection2(CL: TPSPascalCompiler);
28171: begin
28172:   //with RegInterfaceS(CL, 'IUNKNOWN', 'IHttpConnection') do
28173:   with CL.AddInterface(CL.FindInterface('IUnknown'), StringToGuid('{B9611100-5243-4874-A777-D91448517116}'),
28174:     'IHttpConnection2') do
28175:     //with CL.AddInterface(CL.FindInterface('IUNKNOWN'), IHttpConnection2, 'IHttpConnection2') do
28176:   begin
28177:
28178: (*-----*)
28179: //https://github.com/maxkleiner/THHTPSender/blob/master/HTTPSender.pas
28180: procedure SIRegister_HTTPSsender(CL: TPSPascalCompiler);

```

```

28181: begin
28182:   CL.AddConstantN(' _ABOUT_', 'String').SetString( '(c) Z.Razor 20.05.2013 mX4 2022' );
28183:   CL.AddTypeS('THTTPMethodSend', '( hmGetSend, hmPutSend, hmPostSend, hmDeleteSend, hmHeadSend )');
28184:   CL.AddTypeS('THTTPCookie', 'record Domain : String; Name : String; Value : St
28185:     +ring; Expires : String; Path : String; HTTPOnly : boolean; end');
28186:   THTTPCookieArray', 'array of THTTPCookie');
28187:   SIRegister_THTTPCookieCollection(CL);
28188:   THTTPResponseSend', 'record StatusCode : Integer; StatusText : Strin
28189:     +g; RawHeaders : String; ContentLength: Integer; ContentEncoding: String;
28190:     + ' Location : String; Expires : String; end');
28191:   SIRegister_THTTPHeaders(CL);
28192:   SIRegister_THTTPBasicAuth(CL);
28193:   CL.AddTypeS('TCookieAddEvent', 'Procedure ( Sender : TObject; Cookie : THTTPCookie)');
28194:   TWorkBeginEventsend', 'Procedure ( Sender : TObject; WorkCountMax : int64)');
28195:   TWorkEventsend', 'Procedure ( Sender : TObject; WorkCount : int64)');
28196:   TWorkEndEventsend', 'Procedure ( Sender : TObject)');
28197:   THTTPPostContainerFormField', 'record Name : ansistring; Value : ansistring; end');
28198:   THTTPPostContainerFormFieldArray', 'array of THTTPPostContainerFormField');
28199:   THTTPPostContainerFile', 'record Name:ansistring;FileName:ansistring; ContentType:ansistring; end');
28200:   CL.AddTypeS('THTTPPostContainerFileArray', 'array of THTTPPostContainerFile');
28201:   SIRegister_THTTPPostContainer(CL);
28202:   SIRegister_THTTPSender(CL);
28203:   Function __HTTPEncode( const Text : String ) : String';
28204:   Function __HTTPDecode( const Text : String ) : String';
28205:   Function __HTMLDecode( const Text : String ) : String';
28206:   //CL.AddDelphiFunction('Procedure Register');
28207: end;
28208:
28209: procedure SIRegister_THTTPSender(CL: TPSPascalCompiler);
28210: begin
28211:   //with RegClassS(CL, 'TComponent', 'THTTPSender') do
28212:   with CL.AddClassN(CL.FindClass('TComponent'), 'THTTPSender') do begin
28213:     RegisterProperty('DefaultEncoding', 'TEncoding', iptrw);
28214:     Response', 'THTTPResponse', iptr);
28215:     ResponseText', 'ansistring', iptr);
28216:     Function Get( const URL : String ) : String';
28217:     Function Post( const URL : String; postData : ansistring ) : String';
28218:     Function Post1( const URL : String; PostContainer : THTTPPostContainer ) : String';
28219:     Function Post2( const URL : String; postData : TStringList ) : String';
28220:     Function Put( const URL : String ) : String';
28221:     Procedure Get1(const URL : String; Stream : TStream);
28222:     Procedure Post3(const URL : String; postData : ansistring; Stream : TStream);
28223:     Procedure Post4(const URL : String; postData : TStringList; Stream : TStream);
28224:     Procedure Post5(const URL : String; PostContainer : THTTPPostContainer; Stream : TStream);
28225:     Procedure Put1( const URL : String; Stream : TStream);
28226:     Procedure Free';
28227:     Constructor Create( AOwner : TComponent);
28228:     RegisterProperty('Cookies', 'THTTPCookieCollection', iptrw);
28229:     Proxy', 'String', iptrw);
28230:     ProxyBypass', 'String', iptrw);
28231:     AllowCookies', 'boolean', iptrw);
28232:     AutoRedirects', 'boolean', iptrw);
28233:     ConnectTimeout', 'Integer', iptrw);
28234:     ReadTimeout', 'Integer', iptrw);
28235:     SendTimeout', 'Integer', iptrw);
28236:     UseIECookies', 'boolean', iptrw);
28237:     Headers', 'THTTPHeaders', iptrw);
28238:     BasicAuth', 'THTTPBasicAuth', iptrw);
28239:     OnCookieAdd', 'TCookieAddEvent', iptrw);
28240:     OnWorkBegin', 'TWorkBeginEventsend', iptrw);
28241:     OnWork', 'TWorkEventsend', iptrw);
28242:     OnWorkEnd', 'TWorkEndEventsend', iptrw);
28243:     About', 'String', iptr);
28244:   end;
28245: end;
28246:
28247: procedure SIRegister_TRestResource(CL: TPSPascalCompiler);
28248: begin
28249:   //with RegClassS(CL, 'TObject', 'TRestResource') do
28250:   with CL.AddClassN(CL.FindClass('TObject'), 'TRestResource') do begin
28251:     Function GetAcceptTypes : string';
28252:     Function GetURL : string';
28253:     Function GetContent : TStream';
28254:     Function GetContentTypes : string';
28255:     Function GetHeaders : TStringList';
28256:     Function GetAcceptedLanguages : string';
28257:     Function GetAsync : Boolean';
28258:     Function Accept( AcceptType : String ) : TRestResource';
28259:     Function Async( const Value : Boolean ) : TRestResource';
28260:     Function Authorization( Authorization : String ) : TRestResource';
28261:     Function ContentType( ContentType : String ) : TRestResource';
28262:     Function AcceptLanguage( Language : String ) : TRestResource';
28263:     Function Header( Name : String; Value : string ) : TRestResource';
28264:     Function Get : string';
28265:     Procedure Get1( AHandler : TRestResponseHandler);
28266:     Function Get2( EntityClass : TClass ) : TObject';
28267:     Function Post( Content : TStream ) : String';
28268:     Function Post1( Content : string ) : String';
28269:     Procedure Post2( Content : TStream; AHandler: TRestResponseHandler);

```

```

28270:   Function Post3( Entity : TObject) : TObject;');
28271:   Function Post10( Content : string; ResultClass : TClass) : TObject;');
28272:   Function Post11( Content : TStream; ResultClass : TClass) : TObject;');
28273:   Function Post12( Entity : TObject; ResultClass : TClass) : TObject;');
28274:   Function Put13( Content : TStream) : string;');
28275:   Function Put14( Content : string) : string;');
28276:   Procedure Put15( Content : TStream; AHandler: TRestResponseHandler);');
28277:   Function Put16( Entity : TObject) : TObject;');
28278:   Function Put17( Content : string; ResultClass : TClass) : TObject;');
28279:   Function Put18( Content : TStream; ResultClass : TClass) : TObject;');
28280:   Function Put19( Entity : TObject; ResultClass : TClass) : TObject;');
28281:   Function Patch20( Content : TStream) : string;');
28282:   Function Patch21( Content : string) : string;');
28283:   Procedure Patch22( Content : TStream; AHandler: TRestResponseHandler);');
28284:   Function Patch23( Entity : TObject) : TObject;');
28285:   Function Patch24( Content : string; ResultClass : TClass) : TObject;');
28286:   Function Patch25( Content : TStream; ResultClass : TClass) : TObject;');
28287:   Function Patch26( Entity : TObject; ResultClass : TClass) : TObject;');
28288:   Procedure Delete27( );');
28289:   Procedure Delete28( Entity : TObject);');
28290:   Procedure Get29( AHandler : TRestResponseHandlerFunc);');
28291:   Procedure Post30( Content : TStream; AHandler : TRestResponseHandlerFunc);');
28292:   Procedure Post31( Entity : TObject; AHandler : TRestResponseHandlerFunc);');
28293:   Procedure Put32( Content : TStream; AHandler : TRestResponseHandlerFunc);');
28294:   Procedure Patch33( Content : TStream; AHandler: TRestResponseHandlerFunc);');
28295:   Function Get34( AListClass, AItemClass : TClass) : TObject;');
28296:   Function Post35( Adapter : IJsonListAdapter) : TObject;');
28297:   Function Put36( Adapter : IJsonListAdapter) : TObject;');
28298:   Function Patch37( Adapter : IJsonListAdapter) : TObject;');
28299:   Procedure GetAsDataSet38( ADataSet : TDataSet);');
28300:   Function GetAsDataSet39( ) : TDataSet;');
28301:   Function GetAsDataSet40( const RootElement: string) : TDataSet;');
28302:   RegisterMethod('Procedure Delete2( );');
28303:   Procedure Delete3( Entity : TObject);');
28304:   Function Get35( AListClass: TObjectList; AItemClass : TObject) : TObject;');
28305:   Function Get3( AListClass: TObjectList; AItemClass : TObject) : TObject;');
28306: end;
28307: end;
28308:
28309: (*-----*)
28310: procedure SIRegister_TCookie(CL: TPSPascalCompiler);
28311: begin
28312:   //with RegClassS(CL,'TOBJECT', 'TCookie') do
28313:   with CL.AddClassN(CL.FindClass('TOBJECT'),'TCookie') do begin
28314:     RegisterProperty('Name', 'String', iptr);
28315:     RegisterProperty('Value', 'String', iptr);
28316:     RegisterProperty('Version', 'Integer', iptr);
28317:     RegisterProperty('Path', 'String', iptr);
28318:     RegisterProperty('Domain', 'String', iptr);
28319:   end;
28320: end;
28321:
28322: (*-----*)
28323: procedure SIRegister_TRestClient(CL: TPSPascalCompiler);
28324: begin
28325:   //with RegClassS(CL,'TComponent', 'TRestClient') do
28326:   with CL.AddClassN(CL.FindClass('TComponent'),'TRestClient') do begin
28327:     OnBeforeRequest, 'TRestOnRequestEvent', iptrw);
28328:     OnAfterRequest, 'TRestOnRequestEvent', iptrw);
28329:     OnResponse, 'TRestOnResponseEvent', iptrw);
28330:     Constructor Create( Owner : TComponent);
28331:     Procedure Free;
28332:     ResponseCode, 'Integer', iptr);
28333:     ResponseHeader, 'string string', iptr);
28334:     Function Resource( URL : string) : TRestResource;
28335:     Function UnWrapConnection : IHttpConnection;
28336:     Function SetCredentials(const ALogin, APasssword : string) : TRestClient;
28337:     OnConnectionLost, 'THTTTPConnectionLostEvent', iptrw);
28338:     OnError, 'THTTTPErrorEvent', iptrw);
28339:     OnAsyncRequestProcess, 'TAsyncRequestProcessEvent', iptrw);
28340:     ConnectionType, 'THttpConnectionType', iptrw);
28341:     EnabledCompression, 'Boolean', iptrw);
28342:     VerifyCert, 'Boolean', iptrw);
28343:     OnCustomCreateConnection, 'TCustomCreateConnection', iptrw);
28344:     Timeout, 'TTimeout', iptr);
28345:     ProxyCredentials, 'TProxyCredentials', iptr);
28346:   end;
28347: end;
28348:
28349: procedure SIRegister_RestClient(CL: TPSPascalCompiler);
28350: begin
28351:   CL.AddConstantN('DEFAULT_COOKIE_VERSION','LongInt').SetInt( 1);
28352:   CL.AddTypeS('TRestRequestMethod', '( METHOD_GET, METHOD_POST, METHOD_PUT, METHOD_'
28353:     +'PATCH, METHOD_DELETE )');
28354:   CL.AddTypeS('TRestResponseHandler', 'Procedure ( ResponseContent : TStream)');
28355:   CL.AddClassN(CL.FindClass('TOBJECT'),'TRestResource');
28356:   CL.AddTypeS('TCustomCreateConnection', 'Procedure ( Sender : TObject; AConnec
28357:     +'tionType : THttpConnectionType; out AConnection : IHttpConnection)');
28358:   SIRegister_TJsonListAdapter(CL);

```

```

28359: CL.AddClassN(CL.FindClass('TOBJECT'), 'TRestClient');
28360: CL.AddTypeS('TRestOnRequestEvent', 'Procedure ( ARestClient : TRestClient; AR'
28361:   + 'esource : TRestResource; AMethod : TRestRequestMethod)');
28362: TRestOnResponseEvent', 'Procedure (ARestClient:TRestClient;ResponseCode:Integer;const
ResponseContent:string)');
28363: CL.AddTypeS('THTTPErrorEvent', 'Procedure ( ARestClient : TRestClient; AResou'
28364:   + 'rce : TRestResource; AMethod: TRestRequestMethod; AHTTPError:EHTTPError;var
ARetryMode:THTTPRetryMode)');
28365: SIRegister_TRestClient(CL);
28366: SIRegister_TRestCookie(CL);
28367: SIRegister_TRestResource(CL);
28368: //SIRegister_TMultiPartFormAttachment(CL);
28369: //SIRegister_TMultiPartFormData(CL);
28370: end;
28371:
28372: procedure TRestOnResponseEvent2(ARestClient: TRestClient; ResponseCode:Integer;
28373:   const ResponseContent: string);
28374: begin
28375:   print('@addr:'+objtostr(arestclient));
28376:   print('response cont: '+responsecontent);
28377:   print('response code: '+itoa(responsecode));
28378:   print('enabled compression '+botostr(arestclient.EnabledCompression));
28379:   print('content-encoding:'+arestclient.responseheader['Content-Encoding']);
28380:   print('verifycert: '+botostr(arestclient.verifycert));
28381: end;
28382:
28383: function TRestResource3_AskChatGPT(askstream: string;
28384:   aResponseHeader:TRestResponseHandler):string;
28385: var pstrm: TStream;
28386:   JPostdat: string;
28387:   jo: TJSON; arest: TRestResource;
28388: begin
28389:   JPostDat:= '{'+
28390:     '"model": "text-davinci-003",'+
28391:     '"prompt": "%s",'+
28392:     '"max_tokens": 2048,'+
28393:     '"temperature": 0.15}';
28394:
28395:   with TRestClient.create(self) do begin
28396:     arest:= Resource('https://api.openai.com/v1/completions');
28397:     arest.ContentType('application/json');
28398:     arest.Authorization('Bearer '+CHATGPT_APIKEY2);
28399:     ConnectionType:= hctWinInet;
28400:     try
28401:       pstrm:= TStringStream.create(format(JPostDat,[askstream]));
28402:       jo:= TJSON.Create();
28403:       jo.parse(arest.Post(pstrm));
28404:       writeln('respcode: '+itoa(responsecode)+' verifycert: '+botostr(verifycert));
28405:       result:= jo.values['choices'].asarray[0].asobject['text'].asstring;
28406:     finally
28407:       Free;
28408:       jo.Free;
28409:       pStrm.Free;
28410:     end;
28411:   end; //with
28412: end;
28413:
28414: //https://github.com/fabriciocolombo/delphi-rest-client-api/blob/master/src/RestClient.pas
28415:
28416: procedure SIRegister_OpenApiUtils(CL: TPSPascalCompiler);
28417: begin
28418:   CL.AddDelphiFunction('Function PercentEncode( const S : string) : string');
28419:   Procedure AppendQueryParam( var Query : string; const Name, Value : string);
28420:   Function __EncodeBase64( const Input : TBytes) : string;
28421:   Function __DecodeBase64( const Input : string) : TBytes;
28422:   Function __DateTimeToISO( const Value : TDateTime) : string;
28423:   Function DateToISO( const Value : TDate) : string;
28424:   Function __ISOToDateTime( const Value : string) : TDateTime;
28425:   Function ISOToDate( const Value : string) : TDate;
28426:   CL.AddConstantN('SInvalidDateFormat','String').SetString( 'Value %s is not a valid datetime');
28427: end;
28428:
28429: function postRequest(const AUrl, AData: AnsiString; blnSSL: Boolean): AnsiString;
28430: function CheckUrl(Url: string): boolean;
28431: function GetSHA1HashFromStream(AStream: TStream): string;
28432: function ToHexBytes(const AValue: TBytes): AnsiString;
28433: function SliceString(const AString: string; const ADelimiter: string): TStringArray;
28434: function FloatToJson(const value: Double): String;
28435: function CurrToJson(const value: Currency): String;
28436: function CheckUrl2(url: string): boolean;
28437: function GetAdapterInfo(Lana: Char): string;
28438: function IsConnectedToInternet: Boolean;
28439: function WindowsShellExecute(const aOperation: String;
28440:   const aCommandline: string; const aDirectory: string;
28441:   const aParameters: String; const aShow: Integer): Integer;
28442: function WindowsCaptureExecute(aCommandline: String; var aOutput: String): Integer;
28443: procedure WebBrowserLoadFromHTML(AWebBrowser: TWebBrowser; AHTML: tStringlist);
28444: procedure TNovusStreamUtilsCopyMemoryStreamToClipboard(St: TMemoryStream);
28445: procedure CopyMemoryStreamToClipboard(St: TMemoryStream);

```



```

28446: procedure CopyMemoryStreamFromClipboard(St: TMemoryStream);
28447: procedure MemoryStreamToStringList(St:TMemoryStream;Var AStringList:tStringlist);
28448: procedure SaveResourceToFile(const FileName, ResType: string; ResId: Integer);
28449: procedure ShowHintMsgCenter(s: string; _delay: Integer = 2000);
28450: function VarArrayToStr2(const vArray: variant): string;
28451: function VarStrNull2(const V: OleVariant): string;
28452: function CopyObject(Src, Dest: TObject; Related: Boolean = FALSE): Boolean;
28453: function VariantToString(V : OleVariant) : String;
28454: function OleVariantToString(const Value: OleVariant): string;
28455: function StringToOleVariant(const Value: string): OleVariant;
28456: function OleVariantToMemoryStream(const OV: OleVariant): TMemoryStream;
28457: function CreateWord(const aHiByte, aLoByte: byte): word;
28458: function ReverseBytesOrder(aBytes: TBytes): tBytes;
28459: function HexStrToBytes(aHexStr: String; aLittle_Endian: boolean): tBytes;
28460: function FilePathToURL(const aFilePath: string): string;
28461:
28462: procedure SIRegister_TWInHttpRequest(CL: TPSPascalCompiler);
28463: begin
28464:   //with RegClassS(CL,'TOleServer', 'TWInHttpRequest') do
28465:   with CL.AddClassN(CL.FindClass('TOleServer'),'TWInHttpRequest') do begin
28466:     RegisterMethod('Constructor Create( AOwner : TComponent)');
28467:     Procedure Connect;
28468:     Procedure ConnectTo( svrIntf : IWinHttpRequest);
28469:     Procedure Disconnect;
28470:     Procedure Free;
28471:     Procedure ConnectTo( svrIntf : IWinHttpRequest);
28472:     Procedure SetProxy(ProxySetting: HTTPREQUEST_PROXY_SETTING);
28473:     Procedure SetProxy1(ProxySetting: HTTPREQUEST_PROXY_SETTING; ProxyServer : OleVariant);
28474:     Procedure
SetProxy2(ProxySetting:HTTPREQUEST_PROXY_SETTING;ProxyServer:OleVariant;BypassList:OleVariant);
28475:     Procedure SetCredentials(const UserName:WideString;const Password:WideString;Flags:
HTTPREQUEST_SETCREDENTIALS_FLAGS);
28476:     Procedure Open( const Method : WideString; const Url : WideString);
28477:     Procedure Open1( const Method : WideString; const Url : WideString; Async: OleVariant);
28478:     Procedure SetRequestHeader( const Header : WideString; const Value : WideString);
28479:     Function GetResponseHeader( const Header : WideString) : WideString;
28480:     Function GetAllResponseHeaders : WideString;
28481:     Procedure Send;
28482:     Procedure Send1( Body : OleVariant);
28483:     Function WaitForResponse : WordBool;
28484:     Function WaitForResponse1(Timeout : OleVariant) : WordBool;
28485:     Procedure Abort;
28486:     Procedure SetTimeouts(ResolveTimeout:Int;ConnectTimeout:Int;SendTimeout:Int;ReceiveTimeout:Integer);
28487:     Procedure SetClientCertificate( const ClientCertificate : WideString);
28488:     Procedure SetAutoLogonPolicy( AutoLogonPolicy : WinHttpRequestAutoLogonPolicy);
28489:     RegisterProperty('DefaultInterface', 'IWinHttpRequest', iptr);
28490:     Status, 'Integer', iptr);
28491:     StatusText, 'WideString', iptr);
28492:     ResponseText, 'WideString', iptr);
28493:     ResponseBody, 'OleVariant', iptr);
28494:     ResponseStream, 'OleVariant', iptr);
28495:     Option, 'OleVariant WinHttpRequestOption', iptrw);
28496:     Server, 'TWInHttpRequestProperties', iptr);
28497:     OnResponseStart, 'TWInHttpRequestOnResponseStart', iptrw);
28498:     OnResponseDataAvailable, 'TWInHttpRequestOnResponseDataAvailable', iptrw);
28499:     OnResponseFinished, 'TNotifyEvent', iptrw);
28500:     OnError, 'TWInHttpRequestOnError', iptrw);
28501:   end;
28502: end;
28503:
28504: procedure SIRegister_IWinHttpRequest(CL: TPSPascalCompiler);
28505: begin
28506:   //with RegInterfaceS(CL,'IDispatch', 'IWinHttpRequest') do
28507:   // with CL.AddInterface(CL.FindInterface('IDispatch'),IWinHttpRequest, 'IWinHttpRequest') do
28508:   with CL.AddInterface(CL.FindInterface('IDispatch'), StringToGuid('{016FE2EC-B2C8-45F8-B23B-39E53A75396B}'),
'IWinHttpRequest') do begin
28509:     RegisterMethod('Procedure
SetProxy(ProxySetting:HTTPREQUEST_PROXY_SETTING;ProxyServer:OleVariant;BypassList : OleVariant)',
CdStdCall);
28511:     Procedure SetCredentials( const UserName : WideString; const Password : WideString; Flags :
HTTPREQUEST_SETCREDENTIALS_FLAGS)', CdStdCall);
28512:     Procedure Open( const Method : WideString; const Url : WideString; Async : OleVariant)', CdStdCall);
28513:     Procedure SetRequestHeader( const Header : WideString; const Value : WideString)', CdStdCall);
28514:     Function GetResponseHeader( const Header : WideString) : WideString', CdStdCall);
28515:     Function GetAllResponseHeaders : WideString', CdStdCall);
28516:     Procedure Send( Body : OleVariant)', CdStdCall);
28517:     Function Get_Status : Integer', CdStdCall);
28518:     Function Get_StatusText : WideString', CdStdCall);
28519:     Function Get_ResponseText : WideString', CdStdCall);
28520:     Function Get_ResponseBody : OleVariant', CdStdCall);
28521:     Function Get_ResponseStream : OleVariant', CdStdCall);
28522:     Function Get_Option( Option : WinHttpRequestOption) : OleVariant', CdStdCall);
28523:     Procedure Set_Option( Option : WinHttpRequestOption; Value : OleVariant)', CdStdCall);
28524:     Function WaitForResponse( Timeout : OleVariant) : WordBool', CdStdCall);
28525:     Procedure Abort', CdStdCall);
28526:     Procedure SetTimeouts(ResolveTimeout:Int;ConnectTimeout:Int;SendTimeout:Int;ReceiveTimeout:Int)',
CdStdCall);
28527:     Procedure SetClientCertificate( const ClientCertificate : WideString)', CdStdCall);

```

```

28528:     Procedure SetAutoLogonPolicy( AutoLogonPolicy : WinHttpRequestAutoLogonPolicy)', CdStdCall);
28529:   end;
28530: end;
28531:
28532: var
28533:   EmptyParam: OleVariant;    // "Empty parameter" standard constant which can be
28534:   {$EXTERNALSYM EmptyParam} // passed as an optional parameter on a dual
28535:                               // interface.
28536:
28537:
28538: procedure SIRegister_TNovusConsole(CL: TPSPascalCompiler);
28539: begin
28540:   //with RegClassS(CL, 'TOBJECT', 'TNovusConsole') do
28541:   with CL.AddClassN(CL.FindClass('TOBJECT'), 'TNovusConsole') do begin
28542:     Function IsAvailableKey( aHandle : THandle) : Boolean;
28543:     Function IsAvailableKeyEx( aHandle : THandle) : TKeyEvent3;
28544:     Function GetAvailableChar( aHandle : THandle) : char;
28545:     Function GetStdInputHandle : THandle;
28546:     Function GetStdOutputHandle : THandle;
28547:   end;
28548: end;
28549:
28550: V4.7.6.50
28551: -----
28552: procedure SIRegister_Pas2JSUtils(CL: TPSPascalCompiler);
28553: begin
28554:   Function ChompPathDelim2( const Path : string) : string;
28555:   Function GetNextDelimitedItem( const List : string; Delimiter : char; var Position : integer) : string;
28556:   CL.AddTypeS('TChangeStamp', 'Int64');
28557:   Function IncreaseChangeStamp( Stamp : TChangeStamp) : TChangeStamp;
28558:   CL.AddConstantN('EncodingUTF8', 'String').SetString( 'UTF-8');
28559:   CL.AddConstantN('EncodingSystem2', 'String').SetString( 'System2');
28560:   Function NormalizeEncoding( const Encoding : string) : string;
28561:   Function IsASCII2 const s : string) : boolean;
28562:   CL.AddConstantN('UTF8BOM2', 'String').SetString( #SEF#SBB#SBF );
28563:   Function UTF8CharacterStrictLength( P : AnsiChar) : integer;
28564:   Function UTF8ToUTF16( const s : RawByteString) : UnicodeString;
28565:   Function UTF16ToUTF8( const s : UnicodeString) : RawByteString;
28566:   Function IsNonUTF8System : boolean;
28567:   Function GetWindowsEncoding( AConsole : Boolean) : string;
28568:   Function GetUnixEncoding : string;
28569:   Function NonUTF8System : boolean;
28570:   Function GetDefaultTextEncoding : string;
28571:   Procedure SplitCmdLineParams2(const Params: string; ParamList: TStrings; ReadBackslash: boolean);
28572: end;
28573:
28574: procedure SIRegister_Pas2jsFileUtils(CL: TPSPascalCompiler);
28575: begin
28576:   Function jsFilenameIsAbsolute( const aFilename : string) : boolean;
28577:   Function jsFilenameIsWinAbsolute( const aFilename : string) : boolean;
28578:   Function jsFilenameIsUnixAbsolute( const aFilename : string) : boolean;
28579:   Function jsFileIsInPath( const Filename, Path : string) : boolean;
28580:   Function jsChompPathDelim( const Path : string) : string;
28581:   Function jsExpandFileNamePJ( const FileName : string; BaseDir : string) : string;
28582:   Function jsExpandDirectory( const aDirectory : string) : string;
28583:   Function jsIsUNCPath( const Path : String) : Boolean;
28584:   Function jsExtractUNCVolume( const Path : String) : String;
28585:   Function jsExtractFileRoot( FileName : String) : String;
28586:   Function jsTryCreateRelativePath(const Dest:String; const Source: String; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder:Boolean;out RelPath:String): Boolean;
28587:   Function jsResolveDots( const AFilename : string) : string;
28588:   Procedure jsForcePathDelims( var FileName : string);
28589:   Function jsGetForcedPathDelims( const FileName : string) : String;
28590:   Function jsExtractFilenameOnly( const aFilename : string) : string;
28591:   Function jsGetCurrentDirPJ : String;
28592:   Function jsCompareFileNames( const File1, File2 : string) : Integer;
28593:   Function jsFilenameToKey( const Filename : string) : string;
28594:   Function jsGetPhysicalFilename( const Filename : string; ExceptionOnError : boolean) : string;
28595:   Function jsResolveSymLinks( const Filename : string; ExceptionOnError : boolean) : string;
28596:   Function jsMatchGlobbing( Mask, Name : string) : boolean;
28597:   Function jsFileIsWritable( const AFilename : string) : boolean;
28598:   Function jsFileIsExecutable( const AFilename : string) : boolean;
28599:   Function jsGetEnvironmentVariableCountPJ : Integer;
28600:   Function jsGetEnvironmentStringPJ( Index : Integer) : string;
28601:   Function jsGetEnvironmentVariablePJ( const EnvVar : string) : String;
28602:   Function jsGetNextDelimitedItem( const List: string; Delimiter:char; var Position:Integer): string;
28603:   // CL.AddTypeS('TChangeStamp', 'SizeInt');
28604:   //CL.AddDelphiFunction('Procedure IncreaseChangeStamp( var Stamp : TChangeStamp)');
28605:   CL.AddConstantN('jsEncodingUTF8', 'String').SetString( 'UTF-8');
28606:   CL.AddConstantN('jsEncodingSystem', 'String').SetString( 'System');
28607:   Function jsNormalizeEncoding( const Encoding : string) : string;
28608:   Function jsIsNonUTF8System : boolean;
28609:   Function jsGetDefaultTextEncoding : string;
28610:   Function jsGetConsoleTextEncoding : string;
28611:   Function jsGetWindowsEncoding( AConsole : Boolean) : string;
28612:   Function jsGetUnixEncoding : string;
28613:   Function jsIsASCII( const s : AnsiString) : Boolean;
28614:   CL.AddConstantN('jsUTF8BOM', 'String').SetString( RawByteString ( #SEF#SBB#SBF ));
28615:   //CL.AddDelphiFunction('Function UTF8CharacterStrictLength( const P : PAnsiChar) : Integer');

```

```

28616: //Function UTF16CharacterStrictLength( const P : PWideChar) : Integer');
28617: Function jsUTF8ToUTF16( const s : RawByteString) : UnicodeString');
28618: Function jsUTF16ToUTF8( const s : UnicodeString) : RawByteString');
28619: {Function jsUTF8ToSystemCP( const s : RawByteString) : RawByteString');
28620: Function jsSystemCPToUTF8( const s : RawByteString) : RawByteString');
28621: Function jsUTF16ToSystemCP( const s : UnicodeString) : RawByteString');
28622: Function jsSystemCPToUTF16( const s : RawByteString) : UnicodeString');
28623: Function jsUTF8ToConsole( const s : RawByteString) : RawByteString');
28624: Function jsUTF16ToConsole( const s : UnicodeString) : RawByteString');}
28625: end;
28626:
28627: procedure SIRegister_TAsphyreTimer(CL: TPSPascalCompiler);
28628: begin
28629:   //with RegClassS(CL,'OBJECT', 'TAsphyreTimer') do
28630:   with CL.AddClassN(CL.FindClass('OBJECT'),'TAsphyreTimer') do begin
28631:     RegisterProperty('Delta', 'Single', iptr);
28632:     Latency', 'Single', iptr);
28633:     FrameRate', 'Integer', iptr);
28634:     Speed', 'Single', iptrw);
28635:     MaxFPS', 'Integer', iptrw);
28636:     Enabled', 'Boolean', iptrw);
28637:     Precision', 'TPerformancePrecision', iptr);
28638:     SingleCallOnly', 'Boolean', iptrw);
28639:     OnTimer', 'TNotifyEvent', iptrw);
28640:     OnProcess', 'TNotifyEvent', iptrw);
28641:     RegisterMethod('Procedure Process( )');
28642:     Procedure Reset( )');
28643:     Constructor Create( )');
28644:   end;
28645: end;
28646:
28647: procedure SIRegister_uXmlStorage(CL: TPSPascalCompiler);
28648: begin
28649:   SIRegister_TXmlStorage(CL);
28650:   CL.AddDelphiFunction('Function XmlStorage : TXmlStorage');
28651: end;
28652:
28653: procedure SIRegister_TProcess2(CL: TPSPascalCompiler);
28654: begin
28655:   //with RegClassS(CL,'TComponent', 'TProcess2') do
28656:   with CL.AddClassN(CL.FindClass('TComponent'),'TProcess2') do begin
28657:     RegisterMethod(Constructor Create( AOwner : TComponent));
28658:     Procedure Free;');
28659:     Procedure Execute;');
28660:     Procedure CloseInput;');
28661:     Procedure CloseOutput;');
28662:     Procedure CloseStderr;');
28663:     Function Resume : Integer;');
28664:     Function Suspend : Integer;');
28665:     Function Terminate( AExitCode : Integer) : Boolean;');
28666:     Function WaitOnExit : Boolean;');
28667:     RegisterProperty('WindowRect', 'Trect', iptrw);
28668:     Handle', 'THandle', iptr);
28669:     ProcessHandle', 'THandle', iptr);
28670:     ThreadHandle', 'THandle', iptr);
28671:     ProcessID', 'Integer', iptr);
28672:     ThreadID', 'Integer', iptr);
28673:     Input', 'TOutputPipeStream', iptr);
28674:     Output', 'TInputPipeStream', iptr);
28675:     Stderr', 'TinputPipeStream', iptr);
28676:     ExitStatus', 'Integer', iptr);
28677:     ExitCode', 'Integer', iptr);
28678:     InheritHandles', 'Boolean', iptrw);
28679:     OnForkEvent', 'TProcessForkEvent', iptrw);
28680:     PipeBufferSize', 'cardinal', iptrw);
28681:     Active', 'Boolean', iptrw);
28682:     ApplicationName', 'String', iptrw);
28683:     CommandLine', 'String', iptrw);
28684:     Executable', 'String', iptrw);
28685:     Parameters', 'TStrings', iptrw);
28686:     ConsoleTitle', 'String', iptrw);
28687:     CurrentDirectory', 'String', iptrw);
28688:     Desktop', 'String', iptrw);
28689:     Environment', 'TStrings', iptrw);
28690:     Options', 'TProcessOptions2', iptrw);
28691:     Priority', 'TProcessPriority2', iptrw);
28692:     StartupOptions', 'TStartupOptions2', iptrw);
28693:     Running', 'Boolean', iptr);
28694:     ShowWindow', 'TShowWindowOptions2', iptrw);
28695:     WindowColumns', 'Cardinal', iptrw);
28696:     WindowHeight', 'Cardinal', iptrw);
28697:     WindowLeft', 'Cardinal', iptrw);
28698:     WindowRows', 'Cardinal', iptrw);
28699:     WindowTop', 'Cardinal', iptrw);
28700:     WindowWidth', 'Cardinal', iptrw);
28701:     FillAttribute', 'Cardinal', iptrw);
28702:     XTermProgram', 'String', iptrw);
28703:   end;
28704: end;

```

```

28705:
28706: (*-----*)
28707: procedure SIRegister_dprocess(CL: TPSPascalCompiler);
28708: begin
28709:   CL.AddTypeS('TProcessOption2', (poRunSuspended2, poWaitOnExit2, poUsePipes2, poStderrToOutPut2, poNoConsole2,
28710:     poNewConsole2, poDefaultErrorMode2, poNewProcessGroup2, poDebugProcess2, poDebugOnlyThisProcess2 ));
28711:   TShowWindowOptions2', ' (swoNone2, swoHIDE2, swoMaximize2, swoMinimize2, swoRestore2, swoShow2,
28712:     swoShowDefault2, swoShowMaximized2, swoShowMinimized2, swoshowMinNOActive2, swoShowNA2, swoShowNoActivate2,
28713:     swoShowNormal2 ));
28714:   TStartupOption2', ' (suoUseShowWindow2, suoUseSize2, suoUsePosition2, suoUseCountChars2,
28715:     suoUseFillAttribute2 ));
28716:   CL.AddTypeS('TProcessPriority2', ' ( ppHigh2, ppIdle2, ppNormal2, ppRealTime2 ));
28717:   CL.AddTypeS('TProcessOptions2', 'set of TProcessOption2');
28718:   CL.AddTypeS('TStartupOptions2', 'set of TStartupOption2');
28719:   CL.AddTypeS('TProcessForkEvent', 'Procedure ( Sender : TObject)');
28720:   SIRegister_TProcess2(CL);
28721:   CL.AddClassN(CL.FindClass('TObject'), 'EProcess');
28722:   Procedure CommandToList2( S : String; List : TStrings);
28723:   Function DetectXTerm : String;
28724:   Function RunCommandIndir(const curdir:string;const exename:string;const commands:array of string;out
28725:     outputstring: ansistring; out exitstatus: integer;Options: TProcessOptions):integer;
28726:   Function RunCommandIndir1(const curdir: string; const exename:string; const commands:array of string;out
28727:     outputstring: ansistring;Options:TProcessOptions):boolean;
28728:   Function RunCommand(const exename:string;const commands:array of string;out
28729:     outputstring:ansistring;Options: TProcessOptions):boolean;
28730:   Function RunCommandInDir2(const curdir,cmdline: string; out outputstring: ansistring):boolean;
28731:   Function RunCommand1(const cmdline: string; out outputstring: ansistring): boolean;
28732: end;
28733: 1232 unit dwsWebUtils.pas - test for V5
28734: function DecodeURLEncoded(const src : String; start: Integer): String;
28735: procedure ParseURLEncoded(const data : RawByteString; dest : TStrings);
28736: function EncodeURLEncoded(const src : String) : String;
28737: function IsValidCookieName(const s : String) : Boolean;
28738: function IsValidCookieValue(const s : String) : Boolean;
28739: function VariantToString4(const v : Variant) : String;
28740: function PopCount32_Pascal(v : Int32) : Integer;
28741: function PopCount32(v : Integer) : Integer;
28742: function InRange (Lo,Hi,Val : Integer) : Boolean;
28743: CL.AddDelphiFunction('function getchatgpt: string;');
28744: function getChatGPT: string;
28745: function getChatGPT2: string;
28746: Function PlayMidiFile (FileName:string):word;
28747: function GetPIDbyProcessName(processName:String):integer;
28748: function RunCommandSilent(const aCommandline:string;const aDirectory:string;const
28749:   aParameters:String):Integer;
28750: function FindMenuItembyCaption (aCaptions:tCaptions; aMenuItems: tMenuItem): tMenuItem;
28751: function IsFileInUse(fName: string) : boolean;
28752: function GetBuildNumber(aFilename: string = '') : String;
28753: function DateDiff2(Period: Word; Date2, Date1: TDateTime): Longint;
28754: function FormatedMinutesBetween(aStart: tDateTime; aEnd: tDateTime) : String;
28755: function UnixTimeToJSONDate(aUnixTime: Int64) : String;
28756: function WNetGetNetworkInformation2( lpProvider: PChar; var lpNetInfoStruct:TNetInfoStruct2): DWORD);
28757: procedure FormatDotLine(var S: string);
28758: procedure FormatNonDotLine(var S: string);
28759: function GetClassProcessID(aclassname: string): DWORD;
28760: procedure ConvertBinDfmToText(const Filename: string);
28761: function FollowRelativeFilename(const RootDir: string; RelFilename: string): string;
28762: function PipeToFile(Filename: string; Cmd: string; Append: boolean): boolean;
28763:
28764: (*
28765:   "\\\Date(1335205592410)\\\" .NET JavaScriptSerializer
28766:   "\\\Date(1335205592410-0500)\\\" .NET DataContractJsonSerializer
28767:   "2012-04-23T18:25:43.511Z" JavaScript built-in JSON object
28768:   "2012-04-21T18:25:43-05:00" ISO 8601
28769: *)
28770: procedure SIRegister_TNovusWindows(CL: TPSPascalCompiler);
28771: begin
28772:   //with RegClassS(CL, 'TNovusUtilities', 'TNovusWindows') do
28773:   with CL.AddClassN(CL.FindClass('TNovusUtilities'), 'TNovusWindows') do begin
28774:     Function FormatMessStr( aString : String) : String;
28775:     Function GetStrRes( const Index : Integer) : String;
28776:     Function IsWin64 : Boolean;
28777:     Function CommonFilesDir : string;
28778:     Function WindowsSystemDir : string;
28779:     Function WindowsDir : string;
28780:     Function WindowsTempPath : string;
28781:     Function GetLocalComputerName : String;
28782:     Function SetEnvironmentVariableEx(const aVariableName:Str;const
28783:       aValue:string;aIsSystemVariable:Bool):Int);
28784:     Function SetSysEnvironmentVariable(const aVariableName: String; aValue: string): Boolean;
28785:     Function IsProcess32Exists( aFileName : string) : Boolean;
28786:     Function IsStringUniCode( aString : String) : boolean;
28787:     Function GetModuleFileName : String;
28788:   end;
28789: end;
28790: procedure SIRegister_TNovusUtilities(CL: TPSPascalCompiler);

```

```

28785: begin
28786:   //with RegClassS(CL,'TObject', 'TNovusUtilities') do
28787:   with CL.AddClassN(CL.FindClass('TObject'),'TNovusUtilities') do begin
28788:     Function GetExceptMess : String;
28789:     Function GetLastSysErrorMess : string;
28790:     Function CopyObject( Src, Dest : TObject; Related : Boolean) : Boolean;
28791:     Procedure FreeObject( q : TObject);
28792:     Function FindStringListValue( const Strings : TStringList; Name : String) : String;
28793:     Function FindFileSize( Afile : String) : Integer;
28794:     Function GetPropertyasClass( aObject : TObject; aPropertyName : string) : TObject;
28795:     Function IsProperty( aObject : TObject; aPropertyName : string) : Boolean;
28796:     Function GetParamValue( const aParamKey : string; var aValue : string) : Boolean;
28797:     Procedure ClearStringList( aStringList : TStringList);
28798:     Procedure CloneStringList( aSource, aDestination : TStringList);
28799:     Function RegExMatch( aInput : string; aPattern : string; aInversed : Boolean) : String;
28800:   end;
28801: end;
28802:
28803: procedure SIRegister_TNovusStringUtils(CL: TPSPascalCompiler);
28804: begin
28805:   //with RegClassS(CL,'TNovusUtilities', 'TNovusStringUtils') do
28806:   with CL.AddClassN(CL.FindClass('TNovusUtilities'),'TNovusStringUtils') do begin
28807:     Function IsAlphaNumeric( aStr : string) : boolean;
28808:     Function RightTrim( aStr : String) : String;
28809:     Function LeftTrim( aStr : String) : String;
28810:     Function FormatMessStrOptions( aString : String; aFormatOptions : Integer) : String;
28811:     Function MemoryStreamToString( Stream : TMemoryStream) : string;
28812:     Function Str2Float( aStr : String) : single;
28813:     Function Str2Curr( aStr : String) : Currency;
28814:     Function Replicate( c : Char; iLen : Integer) : string;
28815:     Function PadLeft( const s : string; iLen : Integer; const sFillChar : Char) : string;
28816:     Function JustFilename( const aPathName : String) : String;
28817:     Function JustPathName( const aPathName : String) : String;
28818:     Function Str2DateTime( s : String) : TDateTime;
28819:     Procedure VallongInt( s : ShortString; var LI : Longint; var ErrorCode : Integer);
28820:     Function Str2LongS( const s : ShortString; var I : Longint) : Boolean;
28821:     Function IsNumericChar( AChar : Char) : Boolean;
28822:     Function IsAlphaChar( AChar : Char) : Boolean;
28823:     Function IsNumeric( s : String) : Boolean;
28824:     Function IsAlpha( s : String) : Boolean;
28825:     Procedure GetNames( AText : string; AList : TStringList);
28826:     Function SubstCharSim( P : string; OC, NC : ANSIChar) : string;
28827:     Function RootDirectory : String;
28828:     Function StripChar( s : String; Ch : Char) : string;
28829:     Function ReplaceChar( s : String; aFromCh, aToCh : Char) : String;
28830:     Function ReplaceBetweenPositions(const aSource,aReplaceWith:string;aStartPos:Int;aEndPos:Integer)
28831:     String;
28832:     Function GetStrToken(const s:string;sTokens:array of string;var iPos:Integer;var
28833:     sLastToken:String):string;
28834:     Function GetStrTokenA( const s, sDelim : string; var iPos : Integer) : string;
28835:     Function Bool2Str( aValue : Boolean) : String;
28836:     Function Str2Int( aStr : String) : Integer;
28837:     Function Str2Int64( aStr : String) : Int64;
28838:     Function Str2UInt16( aStr : String) : UInt16;
28839:     Function FormatStrVar( const aFormat : string; const Args : array of Variant) : string;
28840:     Function StrChInsertL( const s : AnsiString; c : ANSIChar; Pos : Cardinal) : AnsiString;
28841:     Function IsIntStr( s : String) : Boolean;
28842:     Function MakeTmpFileName( AExt : string; AUseGUID : Boolean) : String;
28843:     Function ReplaceStrPos(const sString,sOldStr,sNewStr:string; APos:Integer; AUseBlank: Boolean) :
28844:     string;
28845:     Function IsLowerChar( Ch : Char) : Boolean;
28846:     Function IsUpperChar( Ch : Char) : Boolean;
28847:     Function UpLower( AName : String; FirstOnly : Boolean) : String;
28848:     Function UpLowerA( AName : String; FirstOnly : Boolean) : String;
28849:     Function BooleanToStr( bValue : Boolean) : string;
28850:     Function StrToBoolean( const sValue : string) : Boolean;
28851:     Function CopyString( const aStr : string; aStartPos, aEndPos : Integer) : String;
28852:     Function ReplaceStr( const sString, sOldStr, sNewStr : string; ACheckUpper : Boolean) : string;
28853:     Procedure ParseStringList( sDelimiter : string; sStr : string; var Lines : TStringList);
28854:     Procedure ParseStringList1( aDelimiter : Char; sStr : string; var Lines : TStringList);
28855:     Procedure String2StringList( aString : string; var AStringList : TStringList);
28856:     Procedure String2Strings( aString : string; var AStrings : TStrings);
28857:     Function VarArrayToStr( const vArray : variant) : string;
28858:     Function VarStrNull( const V : OleVariant) : string;
28859:     Function IsBoolean( const sValue : string) : Boolean;
28860:     Function StrToUInt64( const s : String) : UInt64;
28861:     Function StrToUInt8( const s : String) : UInt8;
28862:     Procedure ClearStringList( aStringList : TStringList);
28863:     Function PadLeft( const Str : string; Ch : Char; Count : Integer) : string;
28864:     Function PadRight( const Str : string; Ch : Char; Count : Integer) : string;
28865:   end;
28866: end;
28867:
28868: procedure SIRegister_TNovusNumUtils(CL: TPSPascalCompiler);
28869: begin
28870:   //with RegClassS(CL,'TNovusUtilities', 'TNovusNumUtils') do
28871:   with CL.AddClassN(CL.FindClass('TNovusUtilities'),'TNovusNumUtils') do begin
28872:     Function HexToInt64( HexStr : String) : Int64;
28873:     Function HexToUInt64( HexStr : String) : UInt64;
28874:   end;
28875: end;

```



```

28871:   Function StrToUInt64( aStr : String) : UInt64);
28872:   Function StrToUInt8( aStr : String) : byte);
28873:   Function HexToUInt16( HexStr : String) : UInt16);
28874:   Function HexToUInt8( HexStr : String) : byte);
28875:   Function Int64ToBin( IValue : Int64; NumBits : word) : string);
28876:   Function BinToInt64( BinStr : string) : Int64);
28877:   Function BinToUInt64( BinStr : string) : UInt64);
28878:   Function BinToUInt8( BinStr : string) : byte);
28879:   Function BinToUInt16( BinStr : string) : word);
28880:   Function StrToUInt16( aStr : String) : word);
28881:   Function CreateWord( const aHiByte, aLoByte : byte) : word);
28882:   Function HexStrToBytes( aHexStr : String; aLittle_Endian : boolean) : tBytes);
28883:   Function ReverseBytesOrder( aBytes : TBytes) : tBytes);
28884: end;
28885: end;
28886:
28887: procedure SIRegister_TNovusFileUtils(CL: TPSPascalCompiler);
28888: begin
28889:   //with RegClassS(CL,'TNovusUtilities', 'TNovusFileUtils') do
28890:   with CL.AddClassN(CL.FindClass('TNovusUtilities'),'TNovusFileUtils') do begin
28891:     Function SwapFilenameExtension( aFilename, aNewExtension : String) : String);
28892:     Function AppRootDirectory : String);
28893:     Function IsFileInUse( fName : string) : boolean);
28894:     Function FilePathToURL( const aFilePath : string) : string);
28895:     Function IsFileReadOnly( fName : string) : boolean);
28896:     Function MoveDir( aFromDirectory, aToDirectory : String) : boolean);
28897:     Function CopyDir( aFromDirectory, aToDirectory : String) : boolean);
28898:     Function ExtractName( aFullFileName : String) : String);
28899:     Function ExtractFileExtA( aFileExt : String) : String);
28900:     Function IsJustFilenameOnly( aFilename : String) : Boolean);
28901:     Function AbsoluteFilePath( aFilename : String) : String);
28902:     Function TrailingBackSlash( const aFilename : string) : string);
28903:     Function GetWindowsSpecialFolder( const CSIDL : integer) : string);
28904:     Function IsOnlyFolder( aFolder : string) : boolean);
28905:     Function IsValidFolder( aFolder : String) : boolean);
28906:     Function IsTextFile( aFilename : String; var aEncoding : tEncoding) : integer);
28907:   end;
28908: end;
28909:
28910: (*-----*)
28911: procedure SIRegister_WebUtils(CL: TPSPascalCompiler); //beta version - not stable!
28912: begin
28913:   //with RegClassS(CL,'TObject', 'WebUtils') do
28914:   with CL.AddClassN(CL.FindClass('TObject'),'WebUtils') do begin
28915:     Procedure ParseURLEncoded( const data : RawByteString; dest : TStrings);
28916:     Function DecodeURLEncoded( const src : RawByteString; start, count : Integer) : String);
28917:     Function DecodeURLEncoded1( const src : RawByteString; start : Integer) : String);
28918:     Function EncodeURLEncoded( const src : String) : String);
28919:     Procedure ParseMIMEHeaderValue( const src : RawByteString; dest : TStrings);
28920:     Procedure ParseMultiPartFormData( const src, dashBoundary : RawByteString; var dest :
TIMIMEBodyParts);
28921:     Function DecodeHex2( p : PAnsiChar) : Integer);
28922:     Function HasFieldName( const list : TStrings; const name : String) : Boolean);
28923:     Function EncodeEncodedWord( const s : String) : String);
28924:     Function DateTimeToRFC822( const dt : TdwsDateTime) : String);
28925:     Function DateTimeToRFC822_1( const dt : TDateTime) : String);
28926:     Function RFC822ToDateTime( const str : String) : TDateTime);
28927:     Function HTMLTextEncode( const s : UnicodeString) : UnicodeString);
28928:     Function HTMLTextDecode( const s : UnicodeString) : UnicodeString);
28929:     Function HTMLCharacterDecode( p : PWideChar) : WideChar);
28930:     Function HTMLAttributeEncode( const s : UnicodeString) : UnicodeString);
28931:     Function HTMLAttributeDecode( const s : UnicodeString) : UnicodeString);
28932:     Function CSSTextEncode( const s : UnicodeString) : UnicodeString);
28933:     Function XMLTextEncode( const s : UnicodeString; unsupportedXML10CharactersMode: Integer) :
UnicodeString);
28934:     Function XMLTextDecode( const s : UnicodeString) : UnicodeString);
28935:     Function IsValidCookieName( const s : String) : Boolean);
28936:     Function IsValidCookieValue( const s : String) : Boolean);
28937:   end;
28938: end;
28939:
28940: procedure SIRegister_TOldPerson(CL: TPSPascalCompiler);
28941: begin
28942:   //with RegClassS(CL,'TPersistent', 'TOldPerson') do
28943:   with CL.AddClassN(CL.FindClass('TPersistent'),'TOldPerson') do begin
28944:     RegisterMethod(Function NewFrom( Id : Integer; Name, EMail : String) : TOldPerson);
28945:     RegisterProperty('id', 'Integer', iptrw);
28946:     RegisterProperty('name', 'String', iptrw);
28947:     RegisterProperty('email', 'String', iptrw);
28948:     RegisterProperty('createDate', 'TDateTime', iptrw);
28949:   end;
28950: end;
28951:
28952: (*-----*)
28953: procedure SIRegister_TPerson(CL: TPSPascalCompiler);
28954: begin
28955:   //with RegClassS(CL,'TObject', 'TPerson') do
28956:   with CL.AddClassN(CL.FindClass('TObject'),'TNewPerson') do begin
28957:     RegisterProperty('id', 'Integer', iptrw);

```

```

28958:   RegisterProperty('name', 'String', iptrw);
28959:   RegisterProperty('email', 'String', iptrw);
28960:   RegisterProperty('createDate', 'TDateTime', iptrw);
28961:   RegisterMethod(Function NewFrom( Id : Integer; Name, EMail : String) : TNewPerson);
28962: end;
28963: end;
28964:
28965: procedure SIRegister_superobject(CL: TPSPascalCompiler);
28966: begin
28967:   CL.AddTypeS('suPtrInt', 'Int64');
28968:   CL.AddTypeS('suPtrUInt', 'UInt64');
28969:   CL.AddTypeS('suPtrInt', 'longint');
28970:   CL.AddTypeS('suPtrUInt', 'Longword');
28971:   CL.AddTypeS('suSuperInt', 'Int64');
28972:   CL.AddTypeS('suSOChar', 'WideChar');
28973:   CL.AddTypeS('suSOIChar', 'Word');
28974:   //CL.AddTypeS('PSOChar', 'PWideChar');
28975:   CL.AddTypeS('suSOStringu', 'UnicodeString');
28976:   CL.AddTypeS('suSOStringw', 'WideString');
28977:   CL.AddTypeS('suSOChar', 'Char');
28978:   CL.AddTypeS('suSOIChar', 'Word');
28979:   CL.AddTypeS('suPSOChar', 'PChar');
28980:   CL.AddTypeS('suSOString', 'string');
28981:   CL.AddConstantN('SUPER_ARRAY_LIST_DEFAULT_SIZE', 'LongInt').SetInt( 32);
28982:   CL.AddConstantN('SUPER_TOKENER_MAX_DEPTH', 'LongInt').SetInt( 32);
28983:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TSuperObject');
28984:   CL.AddInterface(CL.FindInterface('IUNKNOWN'), ISuperObject, 'ISuperObject');
28985:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TSuperArray');
28986:   //CL.AddTypeS('TSuperAvlBitArray', 'set of Integer');
28987:   CL.AddTypeS('TSuperAvlSearchType', '( sustEqual, sustLess, sustGreater )');
28988:   CL.AddTypeS('TSuperAvlSearchTypes', 'set of TSuperAvlSearchType');
28989:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TSuperAvlIterator');
28990:   SIRegister_TSuperAvlEntry(CL);
28991:   SIRegister_TSuperAvlTree(CL);
28992:   SIRegister_TSuperTableString(CL);
28993:   SIRegister_TSuperAvlIterator(CL);
28994:   //CL.AddTypeS('PSuperObjectArray', '^TSuperObjectArray // will not work');
28995:   SIRegister_TSuperArray(CL);
28996:   SIRegister_TSuperWriter(CL);
28997:   SIRegister_TSuperWriterString(CL);
28998:   SIRegister_TSuperWriterStream(CL);
28999:   SIRegister_TSuperAnsiWriterStream(CL);
29000:   SIRegister_TSuperUnicodeWriterStream(CL);
29001:   SIRegister_TSuperWriterFake(CL);
29002:   SIRegister_TSuperWriterSock(CL);
29003:   CL.AddTypeS('TSuperTokenizerError', '( suteSuccess, suteContinue, suteDepth, sutePars'
29004:   + 'eEof, suteParseUnexpected, suteParseNull, suteParseBoolean, suteParseNumber, sutePar'
29005:   + 'seArray, suteParseObjectName, suteParseObjectKeySep, suteParseObjectValueSep, '
29006:   + 'suteParseString, suteParseComment, suteEvalObject, suteEvalArray, suteEvalMethod, suteEvalInt )');
29007:   CL.AddTypeS('TSuperTokenizerState', '( sutsEatws, sutsStart, sutsFinish, sutsNull, sutsCo'
29008:   + 'mmentStart, sutsComment, sutsCommentEol, sutsCommentEnd, sutsString, sutsStringEscap'
29009:   + 'e, sutsIdentifier, sutsEscapeUnicode, sutsEscapeHexadecimal, sutsBoolean, sutsNumber'
29010:   + ', sutsArray, sutsArrayAdd, sutsArraySep, sutsObjectFieldStart, sutsObjectField, sutsOb'
29011:   + 'jectUnquotedField, sutsObjectFieldEnd, sutsObjectValue, sutsObjectValueAdd, sutsOb'
29012:   + 'jectSep, sutsEvalProperty, sutsEvalArray, sutsEvalMethod, sutsParamValue, sutsParamP'
29013:   + 'ut, sutsMethodValue, sutsMethodPut )');
29014:   //CL.AddTypeS('PSuperTokenizerSrec', '^TSuperTokenizerSrec // will not work');
29015:   SIRegister_TSuperEnumerator(CL);
29016:   SIRegister_ISuperObject(CL);
29017:   SIRegister_TSuperObject(CL);
29018:   CL.AddTypeS('TSuperTokenizerSrec', 'record state : TSuperTokenizerState; saved_st'
29019:   + 'ate : TSuperTokenizerState; obj : ISuperObject; current : ISuperObject; fiel'
29020:   + 'd_name : suSOString; parent : ISuperObject; gparent : ISuperObject; end');
29021:   SIRegister_TSuperTokenizer(CL);
29022:   CL.AddTypeS('TSuperType', (sustNull, sustBoolean, sustDouble, sustCurrency, sustInt, sustObject, sustArray,
sustString, sustMethod ));
29023:   CL.AddTypeS('TSuperValidateError', '( suveRuleMalformatted, suveFieldIsRequired, suv'
29024:   + 'eInvalidDataType, suveFieldNotFound, suveUnexpectedField, suveDuplicateEntry, suve'
29025:   + 'ValueNotInEnum, suveInvalidLength, suveInvalidRange )');
29026:   CL.AddTypeS('TSuperFindOption', '( sufoCreatePath, sufoPutValue, sufoDelete, sufoCallMethod )');
29027:   CL.AddTypeS('TSuperFindOptions', 'set of TSuperFindOption');
29028:   CL.AddTypeS('TSuperCompareResult', '( sucplLess, sucplEgu, sucplGreat, sucplError )'); // }
29029:   //SIRegister_TSuperEnumerator(CL);
29030:   //SIRegister_ISuperObject(CL);
29031:   //SIRegister_TSuperObject(CL);
29032:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TSuperRttiContext');
29033:   SIRegister_TSuperAttribute(CL);
29034:   CL.AddClassN(CL.FindClass('TOBJECT'), 'SOName');
29035:   CL.AddClassN(CL.FindClass('TOBJECT'), 'SODefault');
29036:   CL.AddClassN(CL.FindClass('TOBJECT'), 'SOIgnore');
29037:   CL.AddClassN(CL.FindClass('TOBJECT'), 'SOIgnoreSerialize');
29038:   SIRegister_TSuperRttiContext(CL);
29039:   CL.AddTypeS('TSuperObjectIter', 'record key : suSOString; val : ISuperObject; Ite : TSuperAvlIterator;
end');
29040:   Function suObjectIsError( obj : TSuperObject) : boolean);
29041:   Function suObjectIsType( const obj : ISuperObject; typ : TSuperType) : boolean);
29042:   Function suObjectGetType( const obj : ISuperObject) : TSuperType);
29043:   Function suObjectIsNull( const obj : ISuperObject) : Boolean);
29044:   Function suObjectFindFirst( const obj : ISuperObject; var F : TSuperObjectIter) : boolean);

```

```

29045: Function suObjectFindNext( var F : TSuperObjectIter) : boolean');
29046: Procedure suObjectFindClose( var F : TSuperObjectIter)');
29047: Function suSO38( const s : suSOString) : ISuperObject;');
29048: Function suSO39( const value : Variant) : ISuperObject;');
29049: Function suSO40( const Args : array of const) : ISuperObject;');
29050: Function suSA41( const Args : array of const) : ISuperObject;');
29051: Function suJavaToDelphiDateTime( const dt : int64) : TDateTime');
29052: Function suDelphiToJavaDateTime( const dt : TDateTime) : int64');
29053: Function suTryObjectToDate( const obj : ISuperObject; var dt : TDateTime) : Boolean');
29054: Function suUUIDToString( const g : TGUID) : suSOString');
29055: Function siStringToUUID( const str : suSOString; var g : TGUID) : Boolean');
29056: CL.AddTypeS('TSuperInvokeResult', '( suirSuccess, suirMethothodError, suirParamError, suirError )');
29057: //CL.AddDelphiFunction(Function TrySOInvoke42( var ctx : TSuperRttiContext; const obj : TValue; const
method : string; const params : ISuperObject; var Return : ISuperObject) : TSuperInvokeResult;');
29058: //CL.AddDelphiFunction('Function SOInvoke43(const obj:TValue;const method:string; const
params:ISuperObject; ctx:TSuperRttiContext):ISuperObject;');
29059: //CL.AddDelphiFunction(Function SOInvoke44(const obj:TValue; const method:string;const params:string;ctx
: TSuperRttiContext):ISuperObject;');
29060: //CL.AddDelphiFunction(Function IsGenericType( TypeInfo : PTypeInfo) : Boolean');
29061: //Function GetDeclaredGenericType( RttiContext : TRttiContext; TypeInfo : PTypeInfo) : TRttiType');
29062: //Function IsList( RttiContext : TRttiContext; TypeInfo : PTypeInfo) : Boolean');
29063: //Function CreateInstance( RttiContext : TRttiContext; TypeInfo : PTypeInfo) : TValue');
29064: end;
29065:
29066: procedure SIRegister_TRestDM(CL: TPSPascalCompiler);
29067: begin
29068:     //with RegClassS(CL, 'TDataModule', 'TDM') do
29069:     with CL.AddClassN(CL.FindClass('TDataModule'), 'TRestDM') do begin
29070:         RegisterProperty('RestClient', 'TRestClient', iptrw);
29071:     end;
29072: end;
29073:
29074: (*-----*)
29075: procedure SIRegister_uDM(CL: TPSPascalCompiler);
29076: begin
29077:     CL.AddConstantN('REST_CONTEXT_PATH', 'String').SetString('http://localhost:8080/java-rest-server/rest/');
29078:     //CL.AddClassN(CL.FindClass('TOBJECT'), 'EProcess');
29079:     CL.AddClassN(CL.FindClass('TComponent'), 'TRestClient');
29080:     SIRegister_TRestDM(CL);
29081: end;
29082:
29083: type
29084:     TSHFileOpStruct = record
29085:         hwnD: HWND;
29086:         wFunc: UINT;
29087:         pFrom: string;
29088:         pTo: string;
29089:         fFlags: Word;
29090:         fAnyOperationsAborted: BOOL;
29091:         hNameMappings: HWND;
29092:         lpSzProgressTitle: string;
29093:     end;
29094:
29095:     CL.AddTypeS('_NETINFOSTRUCT2', 'record cbStructure : DWORD; dwProviderVersion '
29096:         +: DWORD; dwStatus : DWORD; dwCharacteristics : DWORD; dwHandle : DWORD; wN'
29097:         +etType : Word; dwPrinters : DWORD; dwDrives : DWORD; end');
29098:
29099: Function SHFileOperation( const lpFileOp : TSHFileOpStruct) : Integer');
29100: Function SHFileOperationA( const lpFileOp : TSHFileOpStructA) : Integer');
29101:
29102: procedure SIRegister_dpipes(CL: TPSPascalCompiler);
29103: begin
29104:     CL.AddClassN(CL.FindClass('TOBJECT'), 'EPipeError2');
29105:     TOBJECT', 'EPipeSeek2');
29106:     TOBJECT', 'EPipeCreation2');
29107:     SIRegister_TInputPipeStream2(CL);
29108:     SIRegister_TOutputPipeStream2(CL);
29109: Function CreatePipeHandles2( var Inhandle, OutHandle : THandle; APipeBufferSize : Cardinal) : Boolean');
29110: Procedure CreatePipeStreams2( var InPipe : TInputPipeStream2; var OutPipe : TOutputPipeStream2);
29111:     CL.AddConstantN('EPipeMsg2', 'String').SetString('Failed to create pipe. ');
29112:     CL.AddConstantN('ENoSeekMsg2', 'String').SetString('Cannot seek on pipes');
29113: end;
29114:
29115: procedure SIRegister_tNovusStringBuilder(CL: TPSPascalCompiler);
29116: begin
29117:     //with RegClassS(CL, 'tStringBuilder', 'tNovusStringBuilder') do
29118:     with CL.AddClassN(CL.FindClass('tStringBuilder'), 'tNovusStringBuilder') do begin
29119:         Function AppendasString( const Value : string) : TStringBuilder');
29120:         Function ToStringAll : string');
29121:         Function SaveToFile( aFilename : string) : boolean');
29122:     end;
29123: end;
29124:
29125: (*-----*)
29126: procedure SIRegister_TNovusDateUtils(CL: TPSPascalCompiler);
29127: begin
29128:     //with RegClassS(CL, 'tNovusUtilities', 'tNovusDateUtils') do
29129:     with CL.AddClassN(CL.FindClass('tNovusUtilities'), 'tNovusDateUtils') do begin
29130:         Function IsValidYear( sStr : string) : Boolean');

```

```

29131:   Function IsValidMonth( iMonth : Integer) : Boolean');
29132:   Function isLeapYear( year : Integer) : Boolean');
29133:   Function GetIntMonth( sStr : String) : Integer');
29134:   Function GetIntYear( sStr : String; iPos : Integer) : Integer');
29135:   Function UnixTimeToDateTime( const aUnixDate : Int64) : TDateTime');
29136:   Function DateTimeToISO8601( const aDateTime : TDateTime) : string');
29137:   Function JSONDateToDatetime( aJSONDate : string) : TDateTime');
29138: end;
29139: end;
29140:
29141: (*-----*)
29142: procedure SIRegister_NovusDateUtils(CL: TPSPascalCompiler);
29143: begin
29144:   CL.AddConstantN('novUnixStartDate','Extended').setExtended( 25569.0);
29145:   noviMinYIndex,'LongInt').SetInt( 1);
29146:   noviMaxYIndex,'LongInt').SetInt( 10);
29147:   novTWOZERO,'String').SetString( '20');
29148:   novONENINE,'String').SetString( '19');
29149:   novMonthOnly,'String').SetString( 'm');
29150:   novDayOnly,'String').SetString( 'd');
29151:   novYearOnly,'String').SetString( 'y');
29152:   SIRegister_TNovusDateUtils(CL);
29153: end;
29154:
29155: procedure SIRegister_TPJCBViewer(CL: TPSPascalCompiler);
29156: begin
29157:   //with RegClassS(CL,'TComponent', 'TPJCBViewer') do
29158:   //https://github.com/ddablib/cbview
29159:   with CL.AddClassN(CL.FindClass('TComponent'),'TPJCBViewer') do begin
29160:     RegisterMethod('Constructor Create( AOwner : TComponent)');
29161:     RegisterMethod('Procedure Free;');
29162:     RegisterProperty('OnClipboardChanged', 'TNotifyEvent', iptrw);
29163:     TriggerOnCreation, 'Boolean', iptrw);
29164:     RegisterProperty(Enabled, 'Boolean', iptrw);
29165:   end;
29166: end;
29167:
29168: procedure TDemoFormPJCBViewer1ClipboardChanged(Sender: TObject);
29169: begin
29170:   // Clipboard has changed - emit beep and decide whether to enable memo2
29171:   // (only if text is on clipboard)
29172:   MessageBeep(0);
29173:   if Clipboard.HasFormat(CF_TEXT) then
29174:     Memo2.Lines.Add(Clipboard.AsText);
29175: end;
29176: Example: with TPJCBViewer.create(self) do begin
29177:   //RegisterMethod('Constructor Create( AOwner : TComponent)');
29178:   //RegisterProperty('OnClipboardChanged', 'TNotifyEvent', iptrw);
29179:   //RegisterProperty('TriggerOnCreation', 'Boolean', iptrw);
29180:   OnClipboardChanged:= @TDemoFormPJCBViewer1ClipboardChanged;
29181:   Enabled:= true;
29182:   free;
29183: end;
29184:
29185: procedure SIRegister_TNovusWinVersionUtils(CL: TPSPascalCompiler);
29186: begin
29187:   //with RegClassS(CL,'TNovusUtilities', 'TNovusWinVersionUtils') do
29188:   CL.AddTypeS(novTTranslation, 'record Language : word; CharSet : Word; end');
29189:   with CL.AddClassN(CL.FindClass('TNovusUtilities'),'TNovusWinVersionUtils') do begin
29190:     Function GetFixedFileInfo( FileInfo : __Pointer) : PVFixedFileInfo');
29191:     Function GetFullVersionNumber( aFileName : String) : string');
29192:     Function GetReleaseNumber( aFileName : String) : string');
29193:     Function GetMagMinVersionNumber( aFileName : String) : string');
29194:     Function CreateFileInfo( aFileName : String) : __Pointer');
29195:     Procedure FreeFileInfo( FileInfo : __Pointer);
29196:     Function GetProductName( aFileName : String) : string');
29197:     Function GetLegalCopyright( aFileName : String) : string');
29198:     Function GetTranslationCount( FileInfo : __Pointer) : UINT');
29199:     Function GetTranslation( FileInfo : __Pointer; i : UINT) : novTTranslation');
29200:     Function GetFileInfoString(FileInfo: __Pointer;
Translation:novTTranslation;StringName:string):string);
29201:     RegisterMethod('Function GetBuildNumber( aFilename : string) : string');
29202:   end;
29203: end;
29204:
29205: procedure SIRegister_PJResFile(CL: TPSPascalCompiler);
29206: begin
29207:   CL.AddConstantN('pjRES_MF_MOVEABLE','LongWord').SetUInt( $0010);
29208:   pjRES_MF_PURE,'LongWord').SetUInt( $0020);
29209:   pjRES_MF_PRELOAD,'LongWord').SetUInt( $0040);
29210:   pjRES_MF_DISCARDABLE,'LongWord').SetUInt( $1000);
29211:   pjRT_HTML,'LongInt').SetInt( ( 23 ));
29212:   pjRT_MANIFEST,'LongInt').SetInt( ( 24 ));
29213:   // CL.AddTypeS('TBytes', 'array of Byte');
29214:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPJResourceEntry');
29215:   SIRegister_TPJResourceFileEnumerator(CL);
29216:   SIRegister_TPJResourceFile(CL);
29217:   SIRegister_TPJResourceEntry(CL);
29218:   CL.AddClassN(CL.FindClass('TOBJECT'),'EPJResourceFile');

```

```

29219: Function IsIntResource2( const ResID : PChar) : Boolean');
29220: Function IsEqualResID2( const R1, R2 : PChar) : Boolean');
29221: Function ResIDToStr2( const ResID : PChar) : string';
29222: end;
29223:
29224: Example with subfunction _____:
29225: function GetClassProcessID(aclassname: string): DWORD;
29226: function EnumWinProc(Wnd: HWND; var ProcessID: Integer): Bool; stdcall;
29227: var Buf: array[Byte] of Char;
29228: begin
29229:   Result:= True;
29230:   GetClassName(Wnd, Buf, sizeof(Buf));
29231:   if (StrIComp(Buf, pansichar(aclassname)) = 0) then
29232:     begin
29233:       Result:= False;
29234:       GetWindowThreadProcessId(Wnd, @ProcessID);
29235:     end;
29236:   end;
29237: begin
29238:   EnumWindows(@EnumWinProc, Integer(@Result));
29239: end;
29240:
29241: procedure ConvertBinDfmToText(const Filename: string);
29242: var InStream, OutStream: TStream;
29243: begin
29244:   OutStream:= TMemoryStream.Create;
29245:   try
29246:     InStream:= TFileStream.Create(Filename, fmOpenRead or fmShareDenyWrite);
29247:     try
29248:       ObjectResourceToText(InStream, OutStream);
29249:     finally
29250:       InStream.Free;
29251:     end;
29252:     TMemoryStream(OutStream).SaveToFile(Filename); // overwrite file
29253:   finally
29254:     OutStream.Free;
29255:   end;
29256: end;
29257:
29258: procedure SIRegister_JVCLHelpUtils(CL: TPSPascalCompiler);
29259: begin
29260:   CL.AddConstantN('jvclSpaceChar', 'Char').SetString( #32);
29261:   jvclTabChar, 'Char').SetString( #8);
29262:   MaxGuidListSize, 'LongInt').SetInt( Maxint div 32);
29263:   cUnknownGroupStr, 'String').SetString( 'JVCL.??');
29264:   cSummaryDefaultText, 'String').SetString( 'Write here a summary (1 line)');
29265:   cSummaryDefaultTextForBuild, 'String').SetString( 'Not documented');
29266:   cDescriptionDefaultText, 'String').SetString( 'Write here a description');
29267:   cEditLink, 'String').SetString( '<EXTLINK http://help.delphi-jedi.org/item.php?Name=%s>%s</EXTLINK>');
29268:   cDescriptionDefaultTextForBuild, 'String').SetString('This topic is undocumented. Click on "Edit topic"
to document it. ');
29269:   cDescriptionOverrideDefaultText, 'String').SetString( 'This is an overridden method, you don't have to
describe these if it does ' + 'the same as the inherited method');
29270:   cDescriptionOverloadDefaultText, 'String').SetString( 'This is an overloaded function/procedure, if
possible you may combine the ' + 'description of all these functions into 1 general description. If you do
so, ' + 'combine all "Parameter" lists into 1 list, and leave the "Summary", "Description" ' + 'etc.
fields empty for all other overloaded functions with the same name. ');
29271:   cSeeAlsoDefaultText, 'String').SetString( 'List here other properties, methods (comma seperated) Remove
the "See Also" ' + 'section if there are no references');
29272:   cParameterDefaultText, 'String').SetString( 'Description for this parameter');
29273:   cParameterDefaultTextForBuild, 'string').SetString( 'cSummaryDefaultTextForBuild');
29274:   cReturnValueDefaultText, 'String').SetString( 'Describe here what the function returns');
29275:   cEnumerateDefaultText, 'String').SetString( 'Description for %s');
29276:   cEnumerateDefaultTextForBuild, 'string').SetString( 'cSummaryDefaultTextForBuild');
29277:   CL.AddTypeS('TTopicToken', '( tkSummary, tkNote, tkReturnValue, tkDescription'
+ ', tkParameters, tkSeeAlso, tkJVCLInfo, tkGroup, tkFlag, tkAuthor, tkAlias, '
+ ' tkAliasOf, tkCombine, tkCombineWith, tkTitle, tkTitleImg, xtkHasTocEntry, '
+ ' tkDonator, tkGroup2, tkHasPasFileEntry, tkPlatform, tkTopicOrder, tkOther)');
29281:   TSymbolToken, '( toEOF, toString, toEmpty, toDot, toNumber, xto'
+ Link, toTableBegin, toTableEnd, toBoldBegin, toBoldEnd, toItalicBegin, toI'
+ talicEnd, toUnderlineBegin, toUnderlineEnd, toParagraph, toPreBegin, toPre'
+ End, toCodeBegin, toCodeEnd, toExtLinkBegin, toExtLinkEnd, toImage, toAuto'
+ Link, toNoWhiteSpace, toBOF, toInclude, toColorBegin, toColorEnd, toFlag, '
+ toKeywords, toImplementation, toVersionSpecific, toDelphiLink ');
29287:   CL.AddTypeS('TDtxSelectionKind', '( dskMax, dskPercentage, dskSpecific ');
29288:   CL.AddClassN(CL.FindClass('TObject'), 'TGUIDList');
29289:   //CL.AddTypeS('PGuidItem', '^TGuidItem // will not work');
29290:   CL.AddTypeS('TGuidItem', 'record FGuid : TGuid; FObject : TObject; end');
29291:   //CL.AddTypeS('PGuidItemList', '^TGuidItemList // will not work');
29292:   SIRegister_TGUIDList(CL);
29293:   SIRegister_TIntegerList2(CL);
29294:   SIRegister_TStringStack(CL);
29295:   SIRegister_ITaskManager(CL);
29296:   SIRegister_TProgress2(CL);
29297:   SIRegister_TTask(CL);
29298:   SIRegister_TXMLWriter(CL);
29299:   SIRegister_TDtxBaseItem(CL);
29300:   SIRegister_TSimpleDtxComment(CL);
29301:   SIRegister_TSimpleDtxTopic(CL);

```



```

29302:   SIRegister_TSimpleDtxList(CL);
29303:   CL.AddClassN(CL.FindClass('TObject'),'TStringObjectList');
29304:   SIRegister_TStringObjectList(CL);
29305:   SIRegister_TRegisteredComponents(CL);
29306:   SIRegister_TOwnedStringList(CL);
29307:   Function jvclPad( const S : string; const Width : Integer; const PadChar : Char) : string';
29308:   Procedure jvclWrap( var S : string; const Width, Indent, NextIndent : Integer)';
29309:   Function jvclMaxWordLengthInString( const S : string) : Integer';
29310:   Function jvclLinkStr( const S : string) : string';
29311:   Procedure jvclGetFirstToken( const Source : string; out Token : string; out RestIndex : Integer)';
29312:   Procedure jvclEatChars( var S : string; const Chars : TSysCharSet)';
29313:   Procedure jvclEatChar( var S : string; const Chars : TSysCharSet)';
29314:   Function jvclIsNullStr( const S : string) : Boolean';
29315:   Procedure jvclStatusMsg( const Msg : string)';
29316:   Procedure jvclStatusMsgFmt( const Msg : string; const Args : array of const)';
29317:   Procedure jvclErrorMsg( const Msg : string)';
29318:   Procedure jvclErrorMsgFmt( const Msg : string; const Args : array of const)';
29319:   Procedure jvclWarningMsg( const Msg : string)';
29320:   Procedure jvclWarningMsgFmt( const Msg : string; const Args : array of const)';
29321:   Procedure jvclErrorHandlerFmt( const Msg : string; const Args : array of const)';
29322:   Procedure jvclErrorHandler( const Msg : string)';
29323:   Procedure jvclHintMsg( const Msg : string)';
29324:   Procedure jvclHintMsgFmt( const Msg : string; const Args : array of const)';
29325:   Procedure jvclEnsureMinCount( SS : TStrings; const MinCount : Integer)';
29326:   Procedure jvclEnsureMinCount1( SS : TList; const MinCount : Integer)';
29327:   Function jvclIsParam( const S : string) : Boolean';
29328:   Function jvclIsParam1( const S : string; out Token : string; out RestIndex : Integer) : Boolean';
29329:   Function jvclStartWith2Spaces( const S : string) : Boolean';
29330:   Procedure jvclDiffLists( Source1, Source2, InBoth, NotInSource1, NotInSource2: TStrings; const
CaseSensitive: Bool)';
29331:   Function jvclHasDuplicates( Strings : TStrings) : Boolean';
29332:   Function jvclIsReadOnlyFile( const AFileName : string) : Boolean';
29333:   Function jvclCheckDir( const ADir : string) : Boolean';
29334:   Function jvclCheckFile( const AFileName : string) : Boolean';
29335:   Function jvclAfterDot( const S : string) : string';
29336:   Function jvclBeforeDot( const S : string) : string';
29337:   Procedure jvclGetAllFilesFrom( const ADir, AFilter: string; AFiles: TStrings; const
IncludeSubDirs: Boolean; const StripExtAndPath: Boolean)';
29338:   Function jvclPackageToTopicName( const S : string) : string';
29339:   Function jvclStripOverloadAt( const S : string) : string';
29340:   Function jvclLocateString( Strings : TStrings; const StartIndex : Integer; const S : string) : Integer';
29341:   Function jvclIsPrefix( const Prefix, S : string) : Boolean';
29342:   Function jvclIsPostfix( const Postfix, S : string) : Boolean';
29343:   Function jvclIsSubString( const SubString, S : string) : Boolean';
29344:   Function jvclLocatePrefix( Strings : TStrings; const StartIndex : Integer; const S : string) : Integer';
29345:   Function jvclLocatePostfix( Strings : TStrings; const StartIndex : Integer; const S : string) : Integer';
29346:   Function jvclLocateSubString( Strings : TStrings; const StartIndex : Integer; SubStr: string) : Integer';
29347:   Function jvclChangeSubString( const S, OldStr, NewStr : string) : string';
29348:   Function jvclRegisteredClassNameToImageFileName( const ARegisteredClassName : string) : string';
29349:   Function jvclHasDot( const S : string) : Boolean';
29350:   Function jvclHasAt( const S : string) : Boolean';
29351: end;
29352:
29353: procedure SIRegister_TJvCreateProcess3(CL: TPSPascalCompiler);
29354: begin
29355:   //with RegClassS(CL, 'TJvComponent', 'TJvCreateProcess') do
29356:   with CL.AddClassN(CL.FindClass('TJvComponent'),'TJvCreateProcess2') do begin
29357:     RegisterMethod('Constructor Create( AOwner : TComponent)');
29358:     Procedure Free';
29359:     Function CloseApplication( SendQuit : Boolean) : Boolean';
29360:     Procedure Run';
29361:     Procedure StopWaiting';
29362:     Procedure Terminate';
29363:     Function Write( const S : AnsiString) : Boolean';
29364:     Function WriteLn( const S : AnsiString) : Boolean';
29365:     ProcessInfo, 'TProcessInformation', iptr);
29366:     State, 'TJvCPSSState2', iptr);
29367:     ConsoleOutput, 'TStrings', iptr);
29368:     InputReader, 'TJvBaseReader2', iptr);
29369:     ErrorReader, 'TJvBaseReader2', iptr);
29370:     ApplicationName, 'string', iptrw);
29371:     CommandLine, 'string', iptrw);
29372:     CreationFlags, 'TJvCPSSFlags2', iptrw);
29373:     CurrentDirectory, 'string', iptrw);
29374:     Environment, 'TStrings', iptrw);
29375:     Priority, 'TJvProcessPriority2', iptrw);
29376:     StartupInfo, 'TJvCPSSStartupInfo', iptrw);
29377:     WaitForTerminate, 'Boolean', iptrw);
29378:     ConsoleOptions, 'TJvConsoleOptions2', iptrw);
29379:     OnTerminate, 'TJvCPSTerminateEvent2', iptrw);
29380:     OnRead, 'TJvCPSReadEvent2', iptrw);
29381:     OnRead2, 'TJvCPSReadEvent2', iptrw);
29382:     OnRawRead, 'TJvCPSRawReadEvent2', iptrw);
29383:     OnErrorRead, 'TJvCPSReadEvent2', iptrw);
29384:     OnErrorRawRead, 'TJvCPSRawReadEvent2', iptrw);
29385:   end;
29386: end;
29387:
29388: procedure SIRegister_JvCreateProcess2(CL: TPSPascalCompiler);

```

```

29389: begin
29390:   CL.AddConstantN('CCPS_BufferSize','LongInt').SetInt( 1024);
29391:   CCPS_MaxBufferSize,'LongInt').SetInt( 65536);
29392:   CL.AddTypeS('TJvProcessError','EJVCLException');
29393:   TJvProcessPriority2','( ppIdle2, ppNormal2, ppHigh2, ppRealTime2 ));
29394:   TJvConsoleOption2','( coOwnerData2, coRedirect2, coSeparateError2 ));
29395:   TJvConsoleOptions2','set of TJvConsoleOption2');
29396:   TJvCPSRawReadEvent2','Procedure ( Sender : TObject; const S : string)');
29397:   TJvCPSReadEvent2','Procedure (Sender: TObject; const S:string; const StartsOnNewLine: Boolean)');
29398:   TJvCPSReadEvent3','Procedure (Sender: TObject; const S:string; const StartsOnNewLine: Boolean)');
29399:   TJvCPSTerminateEvent2','Procedure ( Sender : TObject; ExitCode : DWORD)');
29400:   SIRegister_TJvProcessEntry2(CL);
29401:   TJvCPSState2','( psReady2, psRunning2, psWaiting2 ));
29402:   TJvCPSFlag2','( cfDefaultErrorMode2, cfNewConsole2, cfNewProcGroup2
29403:   +, cfSeparateWdm2, cfSharedWdm2, cfSuspended2, cfUnicode2, cfDetached2 ));
29404:   TJvCPSFlags2','set of TJvCPSFlag2');
29405:   TJvCPSShowWindow2','( swHide2, swMinimize2, swMaximize2, swNormal2 ));
29406:   SIRegister_TJvCPSStartupInfo2(CL);
29407:   CL.AddClassN(CL.FindClass('TObject'),'TJvCreateProcess');
29408:   CL.AddClassN(CL.FindClass('TObject'),'TJvCreateProcess2');
29409:   //CL.AddTypeS('TJvCreateProcess2','TJvCreateProcess');
29410:   //type TJvCreateProcess2 = TJvCreateProcess;
29411:   SIRegister_TJvBaseReader2(CL);
29412:   SIRegister_TJvBaseReader3(CL);
29413:   SIRegister_TJvCreateProcess2(CL);
29414:   SIRegister_TJvCreateProcess3(CL);
29415:   //CL.AddClassN(CL.FindClass('TObject'),'TJvCreateProcess2');
29416:   //SIRegister_TJvBaseReader3(CL);
29417: end;
29418:
29419: procedure SIRegister_TModuleLoader(CL: TPSPascalCompiler);
29420: begin
29421:   //with RegClassS(CL,'TObject','TModuleLoader') do
29422:   with CL.AddClassN(CL.FindClass('TObject'),'TModuleLoader') do begin
29423:     Function IsAvaliable( const ADLLName : string; const AProcName : string) : Boolean);
29424:     Constructor Create( const ADLLName : string; LoadMethods : TModuleLoadMethods);
29425:     Procedure Free);
29426:     Function GetProcedure( const AName : string; var AProc : __Pointer) : Boolean);
29427:     Function GetExportedSymbol( const AName : string; var Buffer, Size: Integer): Boolean);
29428:     Function SetExportedSymbol( const AName : string; var Buffer, Size: Integer): Boolean);
29429:     RegisterProperty('Loaded','Boolean', iptr);
29430:     DLLName,'string', iptr);
29431:     Handle,'TModuleHandle', iptr);
29432:   end;
29433: end;
29434:
29435: (*-----*)
29436: procedure SIRegister_JvLogClasses(CL: TPSPascalCompiler);
29437: begin
29438:   CL.AddTypeS('TJvLogEventSeverity','( lesError, lesWarning, lesInformation ));
29439:   SIRegister_TJvLogRecord(CL);
29440:   SIRegister_TJvLogRecordList(CL);
29441:   Function GetSeverityString( const Severity : TJvLogEventSeverity) : string);
29442:   Function GetSeverityFromString( const SeverityString : string) : TJvLogEventSeverity);
29443: end;
29444:
29445: Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo2));
29446: function DeleteFileEx2(const FileName: string): Boolean;
29447: function DeleteFileEx2(const FileName: string): Boolean;
29448: const
29449:   cSuffix = '_del_';
29450: begin
29451:   if FileExists(FileName) then begin
29452:     Result:= RenameFile(FileName, FileName + cSuffix);
29453:     if Result then
29454:       Result:= DeleteFile(Pansichar(FileName + cSuffix));
29455:   end else
29456:     Result := False;
29457: end;
29458:
29459: function BytesOf(const Value: AnsiString): TBytes;
29460: begin
29461:   SetLength(Result, Length(Value));
29462:   if Value <> '' then
29463:     Move(Pointer(Value)^, Result[0], Length(Value));
29464: end;
29465:
29466: procedure SIRegister_uExporter(CL: TPSPascalCompiler);
29467: begin
29468:   CL.AddTypeS('TExporterNavigateDirections','( ndBeginning, ndPrevious, ndNext, ndEnd ));
29469:   CL.AddTypeS('TExporterNavigateResult','( nrBOF, nrOK, nrEOF ));
29470:   CL.AddTypeS('TExporterPos','array of Integer');
29471:   SIRegister_TExporterSourceBase(CL);
29472:   SIRegister_TExporterSource1DBase(CL);
29473:   SIRegister_TExporterSource2DBase(CL);
29474:   SIRegister_TExporterSource3DBase(CL);
29475:   SIRegister_TExporterDestinationBase(CL);
29476:   //CL.AddTypeS('TExporterDestinationBaseClass','class of TExporterDestinationBase');
29477:   SIRegister_TExporter(CL);

```

```

29478:   SIRegister_uExporterDestinationCSV(CL);
29479: end;
29480:
29481: procedure SIRegister_uOptionParser(CL: TPSPascalCompiler);
29482: begin
29483:   //CL.AddConstantN('DefaultIgnoreCase','Boolean')BoolToStr( true);
29484:   //CL.AddConstantN('DefaultIgnoreCase','Boolean')BoolToStr( true);
29485:   CL.AddConstantN('DefaultOptionChars','String').SetString( '-');
29486:   DefaultValueChars,'String').SetString( '=');
29487:   SErrDuplicateOption,'String').SetString( 'Duplicate option "%s"');
29488:   SErrRequiredValue,'String').SetString( 'Option "%s" requires a value');
29489:   SErrRequiredOption,'String').SetString( 'Missing required option "%s"');
29490:   SErrNoValueNeeded,'String').SetString( 'Option "%s" need no value');
29491:   CL.AddClassN(CL.FindClass('TOBJECT'),'TOptionParser');
29492:   CL.AddClassN(CL.FindClass('TOBJECT'),'TOptionDefs');
29493:   CL.AddTypeS('TOptionFlag', '(ofRequired, ofValuePossible, ofValueRequired, ofAllowMultiple)');
29494:   CL.AddTypeS('TOptionFlags', 'set of TOptionFlag');
29495:   SIRegister_TOptionDef(CL);
29496:   SIRegister_TOptionDefs(CL);
29497:   CL.AddTypeS('TExceptionMode', '( emIgnore, emCollect, emRaise )');
29498:   TParsingFlag', '( pfIgnoreCase )');
29499:   TParsingFlags', 'set of TParsingFlag');
29500:   TParserEnvironment', 'record ParamIndex : Integer; ParamValue : '
29501:   +String; LongOption : String; ShortOption : Char; OptionValue : String; end');
29502:   EOptionParserException', 'Exception');
29503:   SIRegister_TOptionParser(CL);
29504: end;
29505:
29506: procedure SIRegister_GUIUtils(CL: TPSPascalCompiler);
29507: begin
29508:   CL.AddConstantN('CChildControlNameToken','String').SetString( '/');
29509:   CChildControlNameWindowToken','String').SetString( '@');
29510:   Function duFindControlInstance( const AComp : TComponent; const AControlName: string) : TControl;');
29511:   Function duFindControlInstance1(const AParentHwnd:HWND;const AControlName:string;var AX:Int;var
   AY:Int):HWND;
29512:   Function duFindParentWinControl( const AControl : TControl) : TWinControl');
29513:   Function duGetWinControl(var AControl:TControl;var AX:Integer;var AY:Int;const
   AISTargetControl:Bool):bool');
29514:   Function duFindComponentNested( const AParent : TComponent; const AName : string) : TComponent');
29515:   Function duWindowClassName( const AHwnd : HWND) : string');
29516:   Function duWindowText( const AHwnd : HWND) : string');
29517:   Function duGetTopmostWindow : HWND');
29518:   //CL.AddDelphiFunction('Procedure SaveScreenshot( const AHwnd : HWND; const AFileName : string)');
29519: end;
29520:
29521: procedure SIRegister_GUIAutomation(CL: TPSPascalCompiler);
29522: begin
29523:   CL.AddConstantN('rcs_id_gui','string').SetString('#(@)$Id: GUIAutomation.pas,v 1.35 2010/05/04 09:55:00
   jarrodh Exp $');
29524:   //CL.AddConstantN('VK_TAB','').SetString( KEY_TAB);
29525:   'CDefaultControlWaitInterval','LongInt').SetInt( 5000);
29526:   'CDefaultWindowChangeWaitInterval','LongInt').SetInt( 10000);
29527:   'CDefaultMouseMovePixelsPerSecond','LongInt').SetInt( 1000);
29528:   CL.AddClassN(CL.FindClass('TOBJECT'),'EGUIAutomation');
29529:   CL.AddClassN(CL.FindClass('TOBJECT'),'EGUIAutomationControlNotFound');
29530:   CL.AddTypeS('TOnGetContinueExecutionEvent', 'Procedure ( var AContinueExecution : boolean)');
29531:   SIRegister_TGUIAutomation(CL);
29532:   'CDefaultGUIActionDelay','LongInt').SetInt( 100);
29533:   'CDefaultGUIMouseMoveDelay','LongInt').SetInt( 100);
29534:   'CDefaultGUIKeyDownDelay','LongInt').SetInt( 50);
29535:   'CDefaultGUITextEntryDelay','LongInt').SetInt( 100);
29536:   'CDefaultGUIControlWaitPeriod','LongInt').SetInt( 1000);
29537:   'CGUIPositionalClickDelay','LongInt').SetInt( 400);
29538:   'CChildControlNameToken','String').SetString( '/');
29539:   'CChildControlNameWindowToken','String').SetString( '@');
29540: end;
29541:
29542: procedure SIRegister_TGUIAutomation(CL: TPSPascalCompiler);
29543: begin
29544:   //with RegClassS(CL,'TObject', 'TGUIAutomation') do
29545:   with CL.AddClassN(CL.FindClass('TObject'),'TGUIAutomation') do begin
29546:     Constructor Create';
29547:     Procedure Free';
29548:     Procedure ThreadedExecute( AProc : TThreadProcedure)');
29549:     Function FindControl13( const AControlName : string; const AAddr : __Pointer) : TControl;');
29550:     Function FindControl14(const AComp:TComponent;const AControlName:string; const AAddr:
   __Pointer):TControl;
29551:     Function FindControlWindow15(const AParentHwnd:HWND; const AControlName:string; var AX:Integer;var AY:
   Integer; const AAddr: __Pointer) : HWND;');
29552:     Function FindControlWindow16(const AControlName:string;var AX:Int;var AY:Int;const
   AAddr: __Pointer):HWND;
29553:     Function FindWinControl(const AControlName:string;var AX:Int;var AY:Int;const
   AAddr: __Pointer):TWinControl;
29554:     Function WaitForWindowChange17(const AInterval : Cardinal) : boolean;');
29555:     Function WaitForWindowChange18(const AWindowCaption : string; const AInterval : Cardinal) : boolean;');
29556:     Function ControlExists( const AControlName : string) : boolean;');
29557:     Function ControlVisible( const AControlName : string) : boolean;');
29558:     Function ControlEnabled( const AControlName : string) : boolean;');
29559:     Function WaitForControlExists(const AControlName:string;const AExists:Bool;const AInterval:Cardinal):
   bool;

```

```

29560:     Function WaitForControlVisible(const AControlName:string;const AVisible:Bool;const
AInterval:Cardinal):bool;
29561:     Function WaitForControlEnabled(const AControlName:string;const AEnabled:Bool;const
AInterval:Cardinal):bool;
29562:     Procedure MoveMouseTo19(const AHwnd : HWND; const AX : Integer; const AY : Integer; const
APixelInterval : Integer; const APixelsPerSecond : Integer);'';
29563:     Procedure LeftMouseDownAt(const AX : Integer; const AY : Integer);'';
29564:     Procedure LeftMouseUpAt(const AX : Integer; const AY : Integer);'';
29565:     Procedure LeftClickAt(const AX : Integer; const AY : Integer);'';
29566:     Procedure LeftDoubleClickAt(const AX : Integer; const AY : Integer);'';
29567:     Procedure RightMouseDownAt(const AX : Integer; const AY : Integer);'';
29568:     Procedure RightMouseUpAt(const AX : Integer; const AY : Integer);'';
29569:     Procedure RightClickAt(const AX : Integer; const AY : Integer);'';
29570:     Procedure RightDoubleClickAt(const AX : Integer; const AY : Integer);'';
29571:     Procedure MoveMouseTo20(const AX:Int;const AY:Int;const APixelInterval:Int;const
APixelsPerSecond:Int);'';
29572:     Procedure LeftMouseDown21(const AX : Integer; const AY : Integer);'';
29573:     Procedure LeftMouseUp22(const AX : Integer; const AY : Integer);'';
29574:     Procedure LeftClick23(const AX : Integer; const AY : Integer);'';
29575:     Procedure LeftDoubleClick24(const AX : Integer; const AY : Integer);'';
29576:     Procedure RightMouseDown25(const AX : Integer; const AY : Integer);'';
29577:     Procedure RightMouseUp26(const AX : Integer; const AY : Integer);'';
29578:     Procedure RightClick27(const AX : Integer; const AY : Integer);'';
29579:     Procedure RightDoubleClick28(const AX : Integer; const AY : Integer);'';
29580:     Procedure MoveMouseTo29(const AControlName : string; const AX : Integer; const AY : Integer; const
APixelInterval : Integer; const APixelsPerSecond : Integer);'';
29581:     Procedure MoveMouseTo30(const AControl : TControl; const AX : Integer; const AY : Integer; const
APixelInterval : Integer; const APixelsPerSecond : Integer);'';
29582:     Procedure LeftMouseDown31(const AControlName : string; const AX : Integer; const AY : Integer);'';
29583:     Procedure LeftMouseDown32(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29584:     Procedure LeftMouseUp33(const AControlName : string; const AX : Integer; const AY : Integer);'';
29585:     Procedure LeftMouseUp34(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29586:     Procedure LeftClick35(const AControlName : string; const AX : Integer; const AY : Integer);'';
29587:     Procedure LeftClick36(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29588:     Procedure LeftDoubleClick37(const AControlName : string; const AX : Integer; const AY : Integer);'';
29589:     Procedure LeftDoubleClick38(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29590:     Procedure RightMouseDown39(const AControlName : string; const AX : Integer; const AY : Integer);'';
29591:     Procedure RightMouseDown40(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29592:     Procedure RightMouseUp41(const AControlName : string; const AX : Integer; const AY : Integer);'';
29593:     Procedure RightMouseUp42(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29594:     Procedure RightClick43(const AControlName : string; const AX : Integer; const AY : Integer);'';
29595:     Procedure RightClick44(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29596:     Procedure RightDoubleClick45(const AControlName : string; const AX : Integer; const AY : Integer);'';
29597:     Procedure RightDoubleClick46(const AControl : TControl; const AX : Integer; const AY : Integer);'';
29598:     Procedure EnterKey47(const AKey : Word; const AShiftState : TShiftState; const ACount : Integer);'';
29599:     Procedure EnterKeyInto48(const AControlHwnd:HWND;const AKey:Word;const AShiftState:TShiftState;const
ACount: Integer);'';
29600:     Procedure EnterKeyInto49(const AControl:TControl;const AKey:Word;const AShiftState:TShiftState;const
ACount:Integer);'';
29601:     Procedure EnterKeyInto50(const AControlName:string;const AKey:Word;const AShiftState:TShiftState;const
ACount: Integer);'';
29602:     Procedure EnterKey51(const AKey : Char; const AShiftState : TShiftState; const ACount : Integer);'';
29603:     Procedure EnterKeyInto52(const AControlHwnd : HWND; const AKey : Char; const AShiftState :
TShiftState; const ACount: Integer);'';
29604:     Procedure EnterKeyInto53(const AControl:TControl;const AKey:Char;const AShiftState:TShiftState;const
ACount: Int);'';
29605:     Procedure EnterKeyInto54(const AControlName:string;const AKey:Char;const AShiftState:TShiftState;const
ACount:Int);'';
29606:     Procedure EnterText(const AText : string);'';
29607:     Procedure EnterTextInto55(const AControlHwnd : HWND; const AText : string);'';
29608:     Procedure EnterTextInto56(const AControl : TControl; const AText : string);'';
29609:     Procedure EnterTextInto57(const AControlName : string; const AText : string);'';
29610:     Procedure SelectMenuItem(const AID : Integer);'';
29611:     Procedure HorizontalScroll58(const AControlHwnd:HWND;const ACommand:Word;const ACount:Integer;const
APosition: Word);'';
29612:     Procedure HorizontalScroll59(const AControl:TControl;const ACommand:Word;const ACount:Integer;const
APosition: Word);'';
29613:     Procedure HorizontalScroll60(const AControlName:string;const ACommand:Word;const ACount:Integer;const
APosition: Word);'';
29614:     Procedure HorizontalScroll61(const AX:Integer;const AY:Int;const ACommand:Word;const ACount:Int;const
APosition:Word);'';
29615:     Procedure HorizontalScroll62(const ACommand:Word;const ACount:Integer;const APosition:Word);'';
29616:     Procedure VerticalScroll63(const AControlHwnd:HWND;const ACommand:Word;const ACount:Int;const
APosition : Word);'';
29617:     Procedure VerticalScroll64(const AControl:TControl;const ACommand:Word;const ACount:Integer;const
APosition :Word);'';
29618:     Procedure VerticalScroll65(const AControlName:string;const ACommand:Word;const ACount:Integer;const
APosition:Word);'';
29619:     Procedure VerticalScroll66(const AX:Integer;const AY:Integer;const ACommand:Word;const
ACount:Integer;const APosition:Word);'';
29620:     Procedure VerticalScroll67(const ACommand : Word; const ACount : Integer; const APosition : Word);'';
29621:     Procedure Show68(const AControl : TControl; AOnOff : boolean);'';
29622:     Procedure Show69(const AControlName : string; AOnOff : boolean);'';
29623:     Procedure Hide70(const AControl : TControl);'';
29624:     Procedure Hide71(const AControlName : string);'';
29625:     Procedure Tab(const n : Integer);'';
29626:     Procedure SetFocus72(const AControl : TControl; const AAddr : __Pointer);'';
29627:     Procedure SetFocus73(const AControlName : string);'';
29628:     Procedure SyncSleep(const AInterval : Cardinal);'';

```

```

29629:   Procedure WakeUp');
29630:   RegisterProperty('ActionDelay', 'Cardinal', iptrw);
29631:   MouseMoveDelay', 'Cardinal', iptrw);
29632:   KeyDownDelay', 'Cardinal', iptrw);
29633:   TextEntryDelay', 'Cardinal', iptrw);
29634:   ControlWaitPeriod', 'Cardinal', iptrw);
29635:   MoveMouseCursor', 'boolean', iptrw);
29636:   OnGetContinueExecution', 'TOnGetContinueExecutionEvent', iptrw);
29637: end;
29638: end;
29639:
29640: procedure SIRegister_GUIActionRecorder(CL: TPSPascalCompiler);
29641: begin
29642:   CL.AddConstantN('rcs_id_guirec', 'string').SetString('#(@)$Id: GUIActionRecorder.pas,v 1.35 2010/05/04
09:55:00 jarrodh Exp $');
29643:   CL.AddConstantN('CWindowChangeCommandName', 'String').SetString('WaitForWindowChange');
29644:   CEnterTextIntoCommandName', 'String').SetString('EnterTextInto');
29645:   CEnterKeyIntoCommandName', 'String').SetString('EnterKeyInto');
29646:   CEnterTextCommandName', 'String').SetString('EnterText');
29647:   CEnterKeyCommandName', 'String').SetString('EnterKey');
29648:   CSelectMenuItemCommandName', 'String').SetString('SelectMenuItem');
29649:   CHorizontalScrollCommandName', 'String').SetString('HorizontalScroll');
29650:   CVerticalScrollCommandName', 'String').SetString('VerticalScroll');
29651:   CLeftButtonCommand', 'String').SetString('Left');
29652:   CRightButtonCommand', 'String').SetString('Right');
29653:   CMiddleButtonCommand', 'String').SetString('Middle');
29654:   CMouseMoveCommand', 'String').SetString('Move');
29655:   CMouseCommand', 'String').SetString('Mouse');
29656:   CMouseDownCommand', 'String').SetString('Down');
29657:   CMouseUpCommand', 'String').SetString('Up');
29658:   CDoubleClickCommand', 'String').SetString('Double');
29659:   CClickCommand', 'String').SetString('Click');
29660:   CClickAtCommand', 'String').SetString('At');
29661:   CClickToCommand', 'String').SetString('To');
29662:   CChildControlNameToken', 'String').SetString('/');
29663:   CChildControlNameWindowToken', 'String').SetString('@');
29664:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TGUIActionAbs');
29665:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TGUIActionMouseButtonAbs');
29666:   CL.AddClassN(CL.FindClass('TOBJECT'), 'TGUIActionRecorder');
29667:   CL.AddTypeS('TGUIActionRecorderEvent', 'Procedure (Sender : TGUIActionRecorder)');
29668:   CL.AddTypeS('TGUIActionRecorderStopRecordingEvent', 'Procedure (Sender : TGU
+IActionRecorder; var AStopRecording: boolean)');
29669:   CL.AddTypeS('TGUIActionRecorderActionEvent', 'Procedure(AAction:TGUIActionAbs;var AContinue: boolean)');
29670:   CL.AddTypeS('TMouseButtonClickState', '( mcsSingle, mcsDouble )');
29671:   CL.AddTypeS('TGUIActionCommandFormat', '( acfNative, acfScript )');
29672:   CL.AddTypeS('TGUIActionMode', '( amMonitor, amRecord )');
29673:   SIRegister_TGUIActionAbs(CL);
29674:   SIRegister_TGUIActionList(CL);
29675:   SIRegister_TGUIActionRecorder(CL);
29676:   SIRegister_TGUIActionWindowChange(CL);
29677:   SIRegister_TGUIActionEnterTextInto(CL);
29678:   SIRegister_TGUIActionEnterKey(CL);
29679:   SIRegister_TGUIActionMouseButtonAbs(CL);
29680:   SIRegister_TGUIActionMouseMove(CL);
29681:   SIRegister_TGUIActionMouseButtonAbs(CL);
29682:   SIRegister_TGUIActionMouseDown(CL);
29683:   SIRegister_TGUIActionMouseUp(CL);
29684:   SIRegister_TGUIActionClick(CL);
29685:   SIRegister_TGUIActionSelectMenuItem(CL);
29686:   SIRegister_TGUIActionScrollAbs(CL);
29687:   //CL.AddTypeS('TGUIActionScrollClass', 'class of TGUIActionScrollAbs');
29688:   SIRegister_TGUIActionHorizontalScroll(CL);
29689:   SIRegister_TGUIActionVerticalScroll(CL);
29690:   Function GGUIActionRecorder : TGUIActionRecorder;
29691:   function NextWord(var P: pchar): String;
29692: end;
29693:
29694:
29695: function GGUIActionRecorder: TGUIActionRecorder;
29696: begin
29697:   if Assigned(URecorder) then
29698:     Result:= URecorder
29699:   else
29700:     Result:= TGUIActionRecorder.Create;
29701:   end;
29702:
29703: https://github.com/coderserdar/DelphiComponents/blob/main/Other/API%20Pack/API\_source/API\_base.pas
29704: procedure SIRegister_API_base(CL: TPSPascalCompiler);
29705: begin
29706:   //CL.AddConstantN('DTMONESECOND', 'LongInt').SetInt( 1 / ( 24 * 60 * 60 ));
29707:   CL.AddConstantN('aBIT0', 'LongInt').SetInt( 1);
29708:   CL.AddConstantN('aBIT1', 'LongInt').SetInt( 2);
29709:   CL.AddConstantN('aBIT2', 'LongInt').SetInt( 4);
29710:   CL.AddConstantN('aBIT3', 'LongInt').SetInt( 8);
29711:   CL.AddConstantN('aBIT4', 'LongInt').SetInt( 16);
29712:   CL.AddConstantN('aBIT5', 'LongInt').SetInt( 32);
29713:   CL.AddConstantN('aBIT6', 'LongInt').SetInt( 64);
29714:   CL.AddConstantN('aBIT7', 'LongInt').SetInt( 128);
29715:   CL.AddConstantN('aBIT8', 'LongInt').SetInt( 256);
29716:   CL.AddConstantN('aBIT9', 'LongInt').SetInt( 512);

```



```

29717: CL.AddConstantN('aBIT10','LongInt').SetInt( 1024);
29718: CL.AddConstantN('aBIT11','LongInt').SetInt( 2048);
29719: CL.AddConstantN('aBIT12','LongInt').SetInt( 4096);
29720: CL.AddConstantN('aBIT13','LongInt').SetInt( 8192);
29721: CL.AddConstantN('aBIT14','LongInt').SetInt( 16384);
29722: CL.AddConstantN('aBIT15','LongInt').SetInt( 32768);
29723: SIRegister_TAPI_Custom_Component(CL);
29724: SIRegister_TAPI_Custom_PaintBox(CL);
29725: SIRegister_TAPI_Custom_Panel(CL);
29726: SIRegister_TAPI_Custom_ListBox(CL);
29727: SIRegister_TAPI_Custom_Edit(CL);
29728: SIRegister_TAPI_Custom_Label(CL);
29729: Procedure apiSetBit( var OfByte : Integer; const BitIndex : integer; const State : Boolean);
29730: Procedure apiSetBit1( var OfByte : Byte; const BitIndex : byte; const State : Boolean);
29731: Function apiBitIsSet( const OfByte, BitIndex : Integer ) : Boolean;
29732: Function apiBinToInt( const ABinStr : AnsiString ) : Integer;
29733: Function apiCeil( const value : double ) : integer;
29734: Function apiFloor( const value : double ) : integer;
29735: Function TickDiff( const StartTick, EndTick : LongWord ) : LongWord;
29736: Function LoadAndRunDLLProcedure(const DLLfile,FunctionName: AnsiString;const
ShowMessages:Boolean):boolean);
29737: end;
29738:
29739: https://github.com/coderserdar/DelphiComponents/blob/main/Other/API%20Pack/API_source/API_audio.pas
29740: procedure SIRegister_API_audio(CL: TPSPascalCompiler);
29741: begin
29742:   CL.AddTypeS('TmmMode', '( ampNotReady, ampStopped, ampPlaying, ampRecording, ampSe'
29743:   + 'eking, ampPaused, ampOpen, ampUnknown )');
29744:   CL.AddTypeS('TmmTimeFormat', '( atfMilliseconds, atfHMS, atfMSF, atfFrames, atfSMP'
29745:   + 'TE24, atfSMPTE25, atfSMPTE30, atfSMPTE30Drop, atfBytes, atfSamples, atfTMSF )');
29746:   SIRegister_TAPI_audiofile(CL);
29747:   SIRegister_TAPI_audio(CL);
29748:   //CL.AddDelphiFunction('Procedure Register');
29749: end;
29750:
29751: procedure SIRegisterTFILESTREAM(Cl: TPSPascalCompiler);
29752: begin
29753:   with Cl.AddClassN(Cl.FindClass('THandleStream'), 'TFileStream') do begin
29754:     RegisterMethod('constructor Create(FileName:String;Mode:Word)');
29755:     constructor Create1(FileName:String;Mode:Word;Rights: Cardinal);
29756:     // constructor Create(const AFileName: string; Mode: Word; Rights: Cardinal); overload;
29757:     //RegisterMethod('function ReadInt(var Buffer;Count:LongInt):LongInt');
29758:     //RegisterMethod('function WriteInt(const Buffer: integer;Count:integer):integer');
29759:     function WriteByteArray(const Buffer:TByteArray;Count:LongInt):LongInt;
29760:     function ReadByteArray(var Buffer: TByteArray;Count:LongInt):LongInt;
29761:     function Testint(aint: integer): integer;
29762:     function ReadBytes(var Buffer: TByteDynArray;Count:LongInt):LongInt;
29763:     function WriteBytes(const Buffer:TByteDynArray;Count:LongInt):LongInt;
29764:     function ReadChars(var Buffer: TCharDynArray;Count:LongInt):LongInt;
29765:     RegisterMethod('Procedure Free');
29766:     RegisterProperty('FileName', 'string', iptr);
29767:     //property FileName: string read FFileName;
29768:   end;
29769: end;
29770:
29771: //https://github.com/coderserdar/DelphiComponents/blob/main/Other/API%20Pack/API_source/API_ledgrid.pas
29772: procedure SIRegister_API_ledgrid(CL: TPSPascalCompiler);
29773: begin
29774:   SIRegister_TAPI_ledgrid(CL);
29775:   //CL.AddDelphiFunction('Procedure Register');
29776:   function BitmapToRTF(const pict: TBitmap): Ansistring;
29777:   BMP:= TBitmap.Create;
29778:   bmp.loadfromfile('C:\maxbox\maxbox3\maxbox3\maxbox3\examples2\max_locomotion.bmp');
29779:   writeln(BitmapToRTF(bmp));
29780:   bmp.free;
29781: end;
29782:
29783: procedure SIRegister_API_graphics(CL: TPSPascalCompiler);
29784: begin
29785:   Procedure apiTextOut( ACanvas : TCanvas; Angle, X, Y : Integer; Text : string);
29786:   Function apiCaptureScreenRect( ARect : TRect ) : TBitmap;
29787:   Function apiCaptureClientImage( Control : TControl ) : TBitmap;
29788:   Function apiInvertColor( color : TColor ) : TColor;
29789:   Function apiInvertBitmap( MyBitmap : TBitmap ) : TBitmap;
29790:   Procedure apiScaleBitmap( bmp : TBitmap; Percent : Double);
29791:   Procedure apiScaleBitmap1( bmp : TBitmap; Percent : Integer);
29792:   Procedure apiReSizeBitmap( bmp : TBitmap; const MaxWidth, MaxHeight : Integer);
29793:   Procedure apiHighlight( aSource, ATarget : TBitmap; AColor : TColor);
29794:   Procedure apiHighlight1( src : TBitmap; ARect : TRect; WhiteWashValue : integer);
29795:   Procedure apiEmboss( ABitmap : TBitmap; Amount : Integer);
29796:   Procedure apiGrayscale( Bmp : TBitmap);
29797:   Procedure apiMosaic( Bitmap : TBitmap; Size : Integer);
29798:   Procedure apiFlipHorizontal( src : Tbitmap);
29799:   Procedure apiFlipVertical( src : Tbitmap);
29800:   Procedure apiPosterize( src, dst : tbitmap; amount : integer);
29801:   Function BitmapToRTF2( const pict : TBitmap ) : Ansistring;
29802:   CL.AddTypeS('TGradientStyle', '( gsHorizontal, gsVertical, gsEllipse, gsHorCenter, gsVerCenter )');
29803:   Procedure apiGradientFill(Canvas:TCanvas;Rect:TRect;StartColor,EndColor:TColor; Style: TGradientStyle);
29804: end;

```

```

29805:
29806: procedure SIRegister_API_files(CL: TPSPascalCompiler);
29807: begin
29808:   CL.AddTypeS('TAPIOnFileFoundEvent', 'Procedure ( Sender : TObject; const Found : string)');
29809:   CL.AddTypeS('TAPIOnFileSearchReady', 'Procedure ( sender : tobjct; const count : integer)');
29810:   SIRegister_TAPI_files(CL);
29811:   //CL.AddDelphiFunction('Procedure Register');
29812:   Function apiAddBackSlash( const Path : string) : string;
29813:   Function apiDeleteBackSlash( const Path : string) : string;
29814:   Function apiPathAppend( const Path, DirToAdd : string) : string;
29815:   Function apiGetUniversalName( const Filename : string) : string;
29816:   Function apiFileExtension( const Filename : String) : string;
29817:   Function apiFileExtension1( const Filename, NewExtension : String) : String;
29818:   Function apiPathCombine( const Folder1, Folder2 : AnsiString) : AnsiString;
29819:   Function apiFileInUse( filename : string) : boolean;
29820:   Function apiGetFileSize( const Filename : string) : int64;
29821:   Function apiIsOnLocalDrive( fname : string) : boolean;
29822:   Function apiGetFileType( const strFilename : string) : string;
29823:   Function apiGetRevision( fname : String; var major, minor, release, build : dword) : boolean;
29824:   Function apiIsASCIIFile( const Filename : string; const CheckOnlyBeginning : Boolean) : Boolean;
29825:   Function apiFileTimeToTDateTime( const AFileTime : TFileTime) : TDateTime;
29826:   Function apiTDateTimeToFileTime( const ADateTime : TDateTime) : TFileTime;
29827:   Function apiFileDateTime( const FileName : string; NewDateTime : TDateTime) : Boolean;
29828:   Function apiFileDateTime1( const filename : string) : tdatetime;
29829:   Function apiGetFileTimes(const FileName:string; var Created:TDateTime;var Accessed:TDateTime;var Modified
: TDateTime):Bool ');
29830:   Function apiAssociatedIcon( FName : String; Idx : Word; var Icon : TIcon) : Boolean;
29831:   Function apiIconFromFile( FName : String; idx : Word; var Icon : TIcon) : Boolean;
29832:   Procedure apiAssociateFile( const Ext, Filetype, Description, Appname : string);
29833:   Function apiExtractIcon( const Filename : string) : ticon;
29834:   Function apiGetFileOwner( const FileName : string; var Domain, Username : string) : Boolean;
29835:   Function apiStartAssociatedExe( const FileName : string; var ErrorCode : Cardinal) : Boolean;
29836:   Function apiDriveSerial( const Drive : Char) : String;
29837:   Function apiParseListDetails(const ListItem:String;var AFileName:String;var AFileTime:TDateTime; var
AFileSize: int64; var AAttributes: integer;const ASeparatorForDetails: char) : Boolean;
29838:   Function apiLastChanged( const Folder: string;const Filter:String; const IncludeSubs:Boolean):
tdatetime;
29839:   Function apiFindFiles(List:Tstrings;const AFolder:String;const AFilter:String;const
ARelativeRoot:String;const AIncludeSubs:Boolean;const AReadOnly:Boolean;const AHidden:Boolean;const
ASeparatorForDetails:char;const ABreakOnFirstFileFound:Boolean):int64;
29840:   Function apiGetSpecialFolder( const fCLSID : integer) : string;
29841:   Function apiIsEmptyFolder( const Folder : String) : boolean;
29842:   Procedure apiFindFolders(List: Tstrings;const Folder: string;const ReadOnly:Boolean;const Hidden :
Boolean);
29843:   Function apiAppDataFolder(const ApplicationName,CompanyName:String;const Personal:Boolean const
CreateDirectory : Boolean) : String;
29844:   Function apiBrowseFolderDialog( const Caption, startfolder : string; CreateFolders : boolean) : string;
29845:   Function apiCopyDir( const fromDir, toDir : string; const AShowProgress : Boolean; const AConfirm :
Boolean; const AAllowUndo : Boolean; const ARenameOnCollision : Boolean) : boolean;
29846:   Function apiMoveDir( const fromDir, toDir : string; const AShowProgress : Boolean; const AConfirm :
Boolean; const AAllowUndo : Boolean) : boolean;
29847:   Function apiDeleteDir(const dir:string;const AShowProgress:Bool;const AConfirm:Bool;const
AAllowUndo:Bool): bool;
29848:   Function apiCopyFile( const Source, Dest : String; const AShowProgress : Boolean; const AConfirm :
Boolean; const AAllowUndo : Boolean; const ARenameOnCollision : Boolean) : Boolean;
29849:   Function apiMoveFile( const Source, Dest : String; const AShowProgress : Boolean; const AConfirm :
Boolean; const AAllowUndo : Boolean) : Boolean;
29850:   Function apiDeleteFile(const Filename:string; const AShowProgress: Boolean;const AConfirm : Boolean;
const AAllowUndo : Boolean) : boolean;
29851:   Function apiRenameFile( const source, dest : string; const AShowProgress : Boolean; const AConfirm :
Boolean; const AAllowUndo : Boolean) : boolean;
29852:   Function apiUpdateFiles(const FromFolder,ToFolder: String;const AIncludeSubs:Boolean;const
AIncludeHiddenFiles:Boolean; const AShowProgress:Boolean;const AConfirm: Boolean; const ACreateFolders
Boolean) : boolean;
29853:   Function apiCombineFiles( const ATargetFilename : String; AFileList: TStrings) : Boolean;
29854:   Function apiGetTempFilename : String;
29855:   Function apiGetTempFile( const Extension : string) : string;
29856:   Function apiCreateLink(const filename,Description,ShortcutTo,Parameters,WorkingDir,IconFilename : String;
iconIndex, ShowCmd : Integer) : Boolean;
29857:   Function apiCRC32( p : __pointer; ByteCount : dword) : dword;
29858:   Function apiCRC321( Filename : string) : dword;
29859:   Procedure apiPasteFileNamesFromClipboard( Filenames : TStrings);
29860:   Function apiCopyFileNamesToClipboard( Filenames : TStrings) : Boolean;
29861:   Function apiCompareFiles( const File1, File2 : TFileName) : Boolean;
29862:   Function apiGetDiskFreeSpace( sDrv : string; var cFree, cSize : int64) : Boolean;
29863: end;
29864:
29865:
29866: procedure SIRegister_API_tools(CL: TPSPascalCompiler);
29867: begin
29868:   CL.AddTypeS('texitttype', '( apiLogOff, apiShutDown, apiReboot, apiPowerOff, apiHibernate )');
29869:   CL.AddTypeS('tostype', '( apiWin95, apiWin98, apiWinMe, apiWinNT, apiWin2000, apiWinXP, apiWin2003,
apiWinVista, apiWinUnknown )');
29870:   SIRegister_TAPI_tools(CL);
29871:   //CL.AddDelphiFunction('Procedure Register');
29872:   Function apiBrowseURL( const URL : string) : boolean;
29873:   Procedure apiShowBalloonToolTip(Control:TWinControl;icon:integer;Title:pchar;Text:WideChar;BackCL,TextCL:
TColor);
29874:   Procedure apiOsBuildInfo( var v1, v2, v3, v4 : word);

```

```

29875: Function apiConnectedToInternet : boolean');
29876: Function apiGetIfTable( pIfTable : __Pointer; var pdwSize : LongInt; bOrder : LongInt) : LongInt');
29877: Function apiNetSend( dest, Source, Msg : string) : Longint');
29878: Procedure apiGetNetworkAddresses( var IPAddress : string; var SubnetMask : string; var Gateway :
string)');
29879: end;
29880:
29881: https://github.com/coderserdar/DelphiComponents/blob/main/Other/API%20Pack/API\_source/API\_winprocess.pas
29882: procedure SIRegister_API_winprocess(CL: TPSPascalCompiler);
29883: begin
29884:   CL.AddTypeS('tprocesspriorities', '( _high_priority_class, _idle_priority_cla
29885:     + 'ss, _normal_priority_class, _realtime_priority_class )');
29886:   SIRegister_TAPI_winprocess(CL);
29887:   //CL.AddDelphiFunction('Procedure Register');
29888: end;
29889:
29890: procedure SIRegister_TAPI_winprocess(CL: TPSPascalCompiler);
29891: begin
29892:   //with RegClassS(CL, 'TComponent', 'TAPI_winprocess') do
29893:   with CL.AddClassN(CL.FindClass('TComponent'), 'TAPI_winprocess') do begin
29894:     RegisterMethod('Constructor Create( aowner : tcomponent)');
29895:     Procedure Free;');
29896:     Procedure show;');
29897:     Procedure hide;');
29898:     Procedure restore;');
29899:     Procedure minimize;');
29900:     Procedure shownoactive;');
29901:     Function running : boolean;');
29902:     Function run( showmode : integer; waitforidle : boolean) : boolean;');
29903:     Function kill( justhandle : boolean) : boolean;');
29904:     RegisterProperty('Version', 'string', iptrw);
29905:     SetPriorities, 'boolean', iptrw);
29906:     WaitIdleTime, 'cardinal', iptrw);
29907:     Filename, 'string', iptrw);
29908:     ProcessPriority, 'integer', iptrw);
29909:     ThreadPriority, 'integer', iptrw);
29910:     LastError, 'string', iptrw);
29911:     OnError, 'tnotifyevent', iptrw);
29912:     AfterRun, 'tnotifyevent', iptrw);
29913:     AfterKill, 'tnotifyevent', iptrw);
29914:   end;
29915: end;
29916:
29917: procedure SIRegister_TAPI_services(CL: TPSPascalCompiler);
29918: begin
29919:   //with RegClassS(CL, 'TComponent', 'TAPI_services') do
29920:   with CL.AddClassN(CL.FindClass('TComponent'), 'TAPI_services') do begin
29921:     RegisterMethod('Constructor Create( aowner : tcomponent)');
29922:     RegisterMethod('Procedure Free;');
29923:     RegisterProperty('Version', 'string', iptr);
29924:     RegisterProperty('Computer', 'string', iptrw);
29925:     RegisterProperty('Service', 'string', iptrw);
29926:     RegisterProperty('Running', 'boolean', iptrw);
29927:     RegisterProperty('Services', 'tstringlist', iptrw);
29928:   end;
29929: end;
29930:
29931: procedure SIRegister_API_strings(CL: TPSPascalCompiler);
29932: begin
29933:   CL.AddConstantN('RETRYCOUNTONREADINGORWRITINGFILE', 'LongInt').SetInt( 10);
29934:   Function apiPos( const LocateThis, FromThisString: AnsiString; const NoCase: Bool; const
StartFrom: Int): integer');
29935:   Function apiPosFromEnd( const LocateThis, FromThisString : AnsiString; const NoCase : boolean) :
integer');
29936:   Procedure apiCharReplace( var InputString : AnsiString; const OldCh, NewCh : AnsiChar)');
29937:   Procedure apiCharsReplace( var InputString : AnsiString; const OldChs, NewChs : AnsiString)');
29938:   Procedure apiStringReplace( var InputString : AnsiString; const OldS, NewS : AnsiString; NoCase :
Boolean)');
29939:   Function apiReplaceFirst( const SourceStr, FindStr, ReplaceStr : AnsiString) : AnsiString');
29940:   Function apiLowerCase( s : AnsiString) : AnsiString');
29941:   Function apiUpperCase( s : AnsiString) : AnsiString');
29942:   Function apiIntToStr( I : int64; LeadingZeros : Integer) : string');
29943:   Function apiBytesToStr( const Size : Int64) : string');
29944:   Function apiTimeToString( T : TDateTime) : string');
29945:   Function apiStringToTime( const S : String; var T : TDateTime) : boolean);
29946:   Function RFC1123ToDateTime( Date : string) : TDateTime');
29947:   Function apiDateTimeToRFC1123( aDate : TDateTime) : string');
29948:   Function apiBinToString( stream : tstream) : string');
29949:   Function apiStringToBin( s : string; stream : tstream) : boolean);
29950:   Function apiStrToPChar( var S : String) : PChar');
29951:   Function apiShortenString(const S:AnsiString;const
MaxLength:Integer;ReplaceCRWith:AnsiString;ReplaceLFWith : AnsiString):AnsiString');
29952:   Procedure apiSwapStrings( var s1, s2 : string)');
29953:   Function apiQuoteIfSpaces( const InputString : AnsiString) : AnsiString');
29954:   Function apiGetNextToken( var SourceString, TokenFound : String; const Separators : AnsiString) :
boolean);
29955:   Procedure apiSplitString( const S : String; var List : TStringlist; const Separators : AnsiString)');
29956:   Function apiCompareStringsInPercent( const Str1, Str2 : AnsiString) : Byte');
29957:   Procedure apiSort( var Strings : TStringList; const Start : Integer; const Stop : Integer)');

```

```

29958: Function apiListToString( const List : TStrings; const Separator : AnsiChar) : AnsiString;');
29959: Procedure apiListToString1(const List:TStrings; var OutputString : AnsiString; const Separator :
AnsiChar);');
29960: Procedure apiStringToList( const InputString : AnsiString; List : Tstrings; const Separator : AnsiChar);');
29961: Function apiOpenFileToStrings(const Filename: String; Strings: tstrings; const AllowRetry: Boolean) :
boolean);
29962: Function apiOpenFileToString(const FileName : string; var S : String; const AllowRetry: Boolean) :
boolean);
29963: Function apiSaveStringsToFile(const Filename: String; Strings: tstrings; const AllowRetry: Boolean) :
boolean);
29964: Function apiSaveStringToFile( const Filename, S : string; const AllowRetry : Boolean) : boolean);
29965: Function apiAppendStringsToFile( const Filename : String; Strings : tstrings) : boolean);
29966: Function apiAppendStringToFile( const Filename, S : String) : boolean);
29967: Function apiStreamWrite( const Str : string; Stream : TStream) : Int64);
29968: Function apiStreamWriteLn(const S: String; Stream: TStream; const Append: Boolean; const EOLN: String):
Int64);
29969: Function apiStreamReadLn( Stream : TStream; const EOLN : String) : String');
29970: Function apiMatch( const N1, N2 : AnsiString; const NoCase : Boolean) : Boolean);
29971: Function apiSameString( const s1, s2 : AnsiString; const NoCase : Boolean) : boolean);
29972: Function apiStringExists( const LookFor, FromThisString : AnsiString; const NoCase : Boolean) : boolean);
29973: Function apiStringBeginsWith( const LookFor, FromThisString : AnsiString; const NoCase : Boolean) :
boolean);
29974: Function apiStringEndsWith( const LookFor, FromThisString : AnsiString; const NoCase : Boolean) :
boolean);
29975: Function apiAddBackslash( const AFilename : AnsiString) : AnsiString');
29976: Function apiRemoveBackslash( const AFilename : AnsiString) : AnsiString');
29977: Function apiFindTagText( const OrigText:AnsiString;StartFrom:Integer;var Position:integer;var
TagContentStart:integer;var TagLength:integer; const StartTag: AnsiString;const EndTag:AnsiString):
AnsiString;');
29978: Function apiFindTagText1(const
OrigText:AnsiString;StartFrom:int;StartTag:AnsiString;EndTag:AnsiString):AnsiString;
29979: Function apiFindTagText2( const OrigText: AnsiString;StartFrom: integer; var Position : integer; StartTag
: AnsiString; EndTag : AnsiString) : AnsiString;');
29980: Function apiHtmlParamsToString( S : AnsiString) : AnsiString');
29981: Function apiURLDecode( Value : AnsiString) : AnsiString');
29982: Function apiStringToHtmlParams( Str : AnsiString) : AnsiString');
29983: Function apiURLEncode( Value : AnsiString) : AnsiString');
29984: Function apiBitmapToRTF( pict : TBitmap) : string');
29985: Function apiEncode64( S : AnsiString) : AnsiString');
29986: Function apiDecode64( S : AnsiString) : AnsiString');
29987: Function apiEncrypt( const InString : AnsiString; StartKey, MultKey, AddKey : Integer) : AnsiString');
29988: Function apiDecrypt( const InString : AnsiString; StartKey, MultKey, AddKey : Integer) : AnsiString');
29989: Function apiHash( const text : AnsiString) : Integer);
29990: Function apiGeneratePass( syllables, numbers : Byte) : string');
29991: end;
29992:
29993:
29994: *****
29995: Release Notes maXbox 4.7.6.50 June 2023 mX476
29996: *****
29997: Add 51 Units + 14 Tutorials
29998:
29999: 1527 unit uPSI_dprocess; TProcess2
30000: 1528 unit uPSI_uXmlStorage.pas
30001: 1529 unit uPSI_AsphyreTimer.pas
30002: 1530 unit uPSI_Pas2JSUtils.pas
30003: 1531 unit uPSI_pacMain; (Formlpac: TFormlpac;)
30004: 1532 unit dwsWebUtils.pas; DWS
30005: 1533 unit uPSI_dwsWebUtils.pas; DWS beta
30006: 1534 unit uPSI_RestUtils2.pas;
30007: 1535 unit uPSI_Pas2jsFileUtils.pas; beta
30008: 1536 unit uPSI_JPerson.pas;
30009: 1537 unit uPSI_OldRttiMarshal;
30010: 1538 unit uPSI_superxmlparser;
30011: 1539 unit uPSI_superobject.pas; beta
30012: 1540 unit uPSI_NovusWindows.pas
30013: 1541 unit uPSI_NovusStringUtils.pas
30014: 1542 unit uPSI_NovusUtilities.pas;
30015: 1543 unit uPSI_NovusNumUtils;
30016: 1544 unit uPSI_NovusFileUtils;
30017: 1545 unit uPSI_NovuscURLUtils.pas; curl beta
30018: 1546 unit uPSI_uDM.pas;
30019: 1547 unit uPSI_dpipes.pas;
30020: 1548 unit uPSI_ShellAPI2;
30021: 1549 unit uPSI_NovusStringBuilder.pas
30022: 1550 unit NovusDateDiffUtil.pas
30023: 1551 unit NovusDateUtils;
30024: 1552 unit NovusDateStringUtils;
30025: 1553 unit uPSI_PJCBView;
30026: 1554 unit uPSI_NovusWinVersionUtils.pas
30027: 1555 unit uPSI_PJResFile.pas;
30028: 1556 unit PJResFile_Routines2;
30029: 1557 unit uPSI_JvCreateProcess2
30030: 1558 unit uPSI_JVCLHelpUtils.pas;
30031: 1559 unit uPSI_ModuleLoader;
30032: 1560 unit uPSI_JvLogClasses;
30033: 1561 unit uPSI_uExporter.pas;
30034: 1562 unit uExporterDestinationCSV.pas;
30035: 1563 unit uPSI_uOptionParser.pas

```

```
30036: 1564 uPSI_TOptionDefs;
30037: 1565 unit uPSI_GUIUtils.pas //dunit
30038: 1566 unit uPSI_GUIAutomation.pas; //dunit
30039: 1567 unit uPSI_GUIActionRecorder; //dunit
30040: 1568 unit TypeHelpers.pas //dunit
30041: 1569 unit uPSI_API_base; //API
30042: 1570 unit uPSI_API_audio; //API
30043: 1571 unit uPSI_API_ledgrid;
30044: 1572 uPSI_API_graphics.pas;
30045: 1573 uPSI_API_files.pas;
30046: 1574 uPSI_API_tools.pas;
30047: 1575 unit uPSI_API_winprocess;
30048: 1576 unit API_strings.pas;
30049: 1577 unit uPSI_API_services.pas
30050:
30051: Total of Function Calls: 37372
30052: SHA1: 4.7.6.50 D047DBD5412C3E4A436089018B9C7FACF17A2EB5
30053: CRC32: 38562FA8 34.04 MB (35,697,944 bytes)
30054: Compilation Timestamp 2023-06-15 06:40:19 UTC Signtime 15 June 2023 08:42:33
30055: Entry Point 25484072 - Contained Sections 10
30056: sha1: d047dbd5412c3e4a436089018b9c7facf17a2eb5
30057: sha256: 193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7
30058: Docu: http://www.softwareschule.ch/maxbox_functions.txt
30059: ZIP maxbox4.zip SHA1: CEAB242D25E264FB95D5792EBC569C4EAD31B732
30060: https://www.hybrid-analysis.com/sample/193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7
30061:
30062: *****
30063: Release Notes maXbox 4.7.6.50 June 2023 mX476
30064: *****
30065: Add 42 Units + 12 Tutorials
30066:
30067: 1527 unit uPSI_dprocess; TProcess2
30068: 1528 unit uPSI_uXmlStorage.pas
30069: 1529 unit uPSI_AsphyreTimer.pas
30070: 1530 unit uPSI_Pas2JSUtils.pas
30071: 1531 unit uPSI_pacMain; (Formlpac: TFormlpac;)
30072: 1532 unit dwsWebUtils.pas; DWS
30073: 1533 unit uPSI_dwsWebUtils.pas; DWS beta
30074: 1534 unit uPSI_RestUtils2.pas;
30075: 1535 unit uPSI_Pas2jsFileUtils.pas; beta
30076: 1536 unit uPSI_JPerson.pas;
30077: 1537 unit uPSI_OldRttiMarshal;
30078: 1538 unit uPSI_superxmlparser;
30079: 1539 unit uPSI_superobject.pas; beta
30080: 1540 uPSI_NovusWindows.pas
30081: 1541 uPSI_NovusStringUtils.pas
30082: 1542 unit uPSI_NovusUtilities.pas;
30083: 1543 unit uPSI_NovusNumUtils;
30084: 1544 unit uPSI_NovusFileUtils;
30085: 1545 unit uPSI_NovuscURLUtils.pas; curl beta
30086: 1546 unit uPSI_uDM.pas;
30087: 1547 unit uPSI_dpipes.pas;
30088: 1548 unit uPSI_ShellAPI2;
30089: 1549 uPSI_NovusStringBuilder.pas
30090: 1550 unit NovusDateDiffUtil.pas
30091: 1551 unit NovusDateUtils;
30092: 1552 unit NovusDateStringUtils;
30093: 1553 unit uPSI_PJCBView;
30094: 1554 uPSI_NovusWinVersionUtils.pas
30095: 1555 unit uPSI_PJResFile.pas;
30096: 1556 unit PJResFile_Routines2;
30097: 1557 unit uPSI_JvCreateProcess2
30098: 1558 unit uPSI_JVCLHelpUtils.pas;
30099: 1559 unit uPSI_ModuleLoader;
30100: 1560 unit uPSI_JvLogClasses;
30101: 1561 uPSI_uExporter.pas;
30102: 1562 unit uExporterDestinationCSV.pas;
30103: 1563 uPSI_uOptionParser.pas
30104: 1564 uPSI_TOptionDefs;
30105: 1565 unit uPSI_GUIUtils.pas
30106: 1566 unit uPSI_GUIAutomation.pas;
30107: 1567 unit uPSI_GUIActionRecorder; //dunit
30108: 1568 unit TypeHelpers.pas //dunit
30109:
30110: Total of Function Calls: 37121
30111: SHA1: 4.7.6.50 9E9D9D10762AE0D0BC8AFD747C2381EA5478AE49
30112: CRC32: 01B431EE 33.91 MB (35,558,168 bytes)
30113: Compilation Timestamp 2023-06-12 13:17:28 UTC Signtime 12 June 2023 15:20:10
30114: Entry Point 25377544 - Contained Sections 10
30115: sha1: 9e9d9d10762ae0d0bc8afd747c2381ea5478ae49
30116: sha256: fa0f30abf34292e91070a5bd4682040eb6af79a8f6c7f55111c9692153120988
30117: Docu: http://www.softwareschule.ch/maxbox_functions.txt
30118:
30119: Recompiled: dir /s /o N *.dcu | find "11/06/2023" > C:\maXbox\maxboxunitalldisk_sort.txt
30120:
30121: 11/06/2023 23:12 489,126 fMain.dcu
30122: 11/06/2023 23:12 71,764 ALHttpClient2.dcu
30123: 11/06/2023 23:12 13,520 BlocksUnit.dcu
30124: 11/06/2023 23:12 5,720 FPCTypes.dcu
```



```

30125: 11/06/2023 23:10 49,763 GUIAutomation.dcu
30126: 11/06/2023 22:57 7,645 GUIUtils.dcu
30127: 11/06/2023 23:14 30,858 MathsLib.dcu
30128: 11/06/2023 23:12 7,881 NovusDateUtils.dcu
30129: 11/06/2023 23:12 9,464 NovusNumUtils.dcu
30130: 11/06/2023 23:12 26,397 NovusStringUtils.dcu
30131: 11/06/2023 23:12 6,566 PXLTiming.dcu
30132: 11/06/2023 23:12 7,427 PythonAction.dcu
30133: 11/06/2023 23:12 278,921 PythonEngine.dcu
30134: 11/06/2023 23:12 6,446 SeSHA256.dcu
30135: 11/06/2023 23:12 126,239 SuperObject.dcu
30136: 11/06/2023 18:48 153,589 uPSI_AfUtils.dcu
30137: 11/06/2023 23:12 66,636 uPSI_ALHttpClient2.dcu
30138: 11/06/2023 23:12 20,496 uPSI_BlocksUnit.dcu
30139: 11/06/2023 22:57 37,828 uPSI_GUIAutomation.dcu
30140: 11/06/2023 21:28 4,798 uPSI_GUIUtils.dcu
30141: 11/06/2023 23:12 14,466 uPSI_NovusStringUtils.dcu
30142: 11/06/2023 23:12 20,134 uPSI_NovusUtilities.dcu
30143: 11/06/2023 21:05 124,494 uPSI_PsAPI.dcu
30144: 11/06/2023 23:12 8,645 uPSI_PXLTiming.dcu
30145: 11/06/2023 23:12 175,131 uPSI_PythonEngine.dcu
30146: 11/06/2023 23:12 9,517 uPSI_VelthuisFloatUtils.dcu
30147: 11/06/2023 23:12 46,249 VarPyth.dcu
30148: 11/06/2023 23:12 5,682 VelthuisFloatUtils.dcu
30149:
30150: 06/06/2023 15:47 30,882 uPSUtils.dcu
30151: 06/06/2023 07:55 64,580 JVCLHelpUtils.dcu
30152: 06/06/2023 15:47 30,854 MathsLib.dcu
30153: 06/06/2023 15:47 105,422 uPSI_AzuliaUtils.dcu
30154: 06/06/2023 08:10 46,235 uPSI_JVCLHelpUtils.dcu
30155: 06/06/2023 08:31 26,153 uPSI_LazFileUtils.dcu
30156: 06/06/2023 15:14 112,192 uPSI_PersistSettings.dcu
30157:
30158: 05/06/2023 21:20 488,462 fMain.dcu
30159: 05/06/2023 21:20 71,768 ALHttpClient2.dcu
30160: 05/06/2023 21:20 13,520 BlocksUnit.dcu
30161: 05/06/2023 21:20 5,720 FPCTypes.dcu
30162: 05/06/2023 20:11 63,516 JvCreateProcess.dcu
30163: 05/06/2023 21:20 7,879 NovusDateUtils.dcu
30164: 05/06/2023 21:20 9,465 NovusNumUtils.dcu
30165: 05/06/2023 21:20 26,408 NovusStringUtils.dcu
30166: 05/06/2023 21:20 6,566 PXLTiming.dcu
30167: 05/06/2023 21:20 7,429 PythonAction.dcu
30168: 05/06/2023 21:20 278,927 PythonEngine.dcu
30169: 05/06/2023 21:20 6,446 SeSHA256.dcu
30170: 05/06/2023 21:20 126,255 SuperObject.dcu
30171: 05/06/2023 21:20 66,640 uPSI_ALHttpClient2.dcu
30172: 05/06/2023 21:20 20,494 uPSI_BlocksUnit.dcu
30173: 05/06/2023 20:44 36,468 uPSI_JvCreateProcess.dcu
30174: 05/06/2023 15:39 19,024 uPSI_JvSysComp.dcu
30175: 05/06/2023 21:20 14,466 uPSI_NovusStringUtils.dcu
30176: 05/06/2023 21:20 20,134 uPSI_NovusUtilities.dcu
30177: 05/06/2023 21:20 8,645 uPSI_PXLTiming.dcu
30178: 05/06/2023 21:20 175,131 uPSI_PythonEngine.dcu
30179: 05/06/2023 21:20 9,517 uPSI_VelthuisFloatUtils.dcu
30180: 05/06/2023 21:20 46,252 VarPyth.dcu
30181: 05/06/2023 21:20 5,682 VelthuisFloatUtils.dcu
30182:
30183: Recompiled: dir /s /o N *.dcu | find "30/05/2023" > C:\maxXbox\maxboxunitalldisk_sort.txt
30184:
30185: 29/05/2023 23:35 255,335 uPSI_ComCtrls.dcu
30186: 30/05/2023 08:32 16,685 uPSI_ShellAPI.dcu
30187: 30/05/2023 09:04 26,921 dprocess.dcu
30188: 30/05/2023 00:02 33,458 RestClient.dcu
30189: 30/05/2023 00:02 1,664 uDM.dcu
30190: 30/05/2023 09:58 23,975 uPSI_dprocess.dcu
30191: 30/05/2023 00:13 30,416 uPSI_RestClient.dcu
30192: 30/05/2023 00:02 25,875 uPSI_RestUtils.dcu
30193: 30/05/2023 08:32 16,685 uPSI_ShellAPI.dcu
30194: 30/05/2023 11:13 8,158 dpipes.dcu
30195: 30/05/2023 15:17 30,863 MathsLib.dcu
30196: 30/05/2023 15:03 7,884 NovusDateUtils.dcu
30197: 30/05/2023 13:49 2,281 NovusStringBuilder.dcu
30198: 30/05/2023 14:31 102,632 uPSI_AzuliaUtils.dcu
30199: 30/05/2023 15:08 16,776 uPSI_NovusUtilities.dcu
30200:
30201: Recompiled:
30202: c:\>dir /s /o N *.dcu | find "31/05/2023" > C:\maxXbox\maxboxunitalldisk_sort.txt
30203: 31/05/2023 10:25 30,854 MathsLib.dcu
30204: 31/05/2023 09:31 7,618 NovusWinVersionUtils.dcu
30205: 31/05/2023 09:01 5,177 PJCBView.dcu
30206: 31/05/2023 09:27 104,722 uPSI_AzuliaUtils.dcu
30207: 31/05/2023 09:43 20,141 uPSI_NovusUtilities.dcu
30208: 31/05/2023 10:24 121,280 uPSI_PsAPI.dcu
30209: 31/05/2023 14:30 488,274 fMain.dcu
30210: 31/05/2023 14:04 316,953 uPSCompiler.dcu
30211: 31/05/2023 14:05 20,906 uPSPreProcessor.dcu
30212: 31/05/2023 14:03 257,726 uPSRuntime.dcu
30213: 31/05/2023 14:33 71,774 ALHttpClient2.dcu

```

```

30214: 31/05/2023 14:33 13,510 BlocksUnit.dcu
30215: 31/05/2023 14:33 5,720 FPCTypes.dcu
30216: 31/05/2023 14:44 30,869 MathsLib.dcu
30217: 31/05/2023 14:33 7,889 NovusDateUtils.dcu
30218: 31/05/2023 14:33 9,475 NovusNumUtils.dcu
30219: 31/05/2023 14:33 26,441 NovusStringUtils.dcu
30220: 31/05/2023 09:31 7,618 NovusWinVersionUtils.dcu
30221: 31/05/2023 09:01 5,177 PJCBView.dcu
30222: 31/05/2023 14:07 24,904 PJResFile.dcu
30223: 31/05/2023 14:33 6,567 PXLTiming.dcu
30224: 31/05/2023 14:33 7,434 PythonAction.dcu
30225: 31/05/2023 14:33 278,926 PythonEngine.dcu
30226: 31/05/2023 14:33 6,446 SeSHA256.dcu
30227: 31/05/2023 14:33 126,287 SuperObject.dcu
30228: 31/05/2023 14:33 66,636 uPSI_ALHttpClient2.dcu
30229: 31/05/2023 09:27 104,722 uPSI_AzuliasUtils.dcu
30230: 31/05/2023 14:33 20,494 uPSI_BlocksUnit.dcu
30231: 31/05/2023 14:33 14,466 uPSI_NovusStringUtils.dcu
30232: 31/05/2023 14:33 20,138 uPSI_NovusUtilities.dcu
30233: 31/05/2023 14:30 14,168 uPSI_PJResFile.dcu
30234: 31/05/2023 10:24 121,280 uPSI_PsAPI.dcu
30235: 31/05/2023 14:33 8,642 uPSI_PXLTiming.dcu
30236: 31/05/2023 14:33 175,125 uPSI_PythonEngine.dcu
30237: 31/05/2023 14:33 9,515 uPSI_VelthuisFloatUtils.dcu
30238: 31/05/2023 14:33 46,241 VarPyth.dcu
30239: 31/05/2023 14:33 5,682 VelthuisFloatUtils.dcu
30240:
30241: 2023-05 Cumulative Update for Windows 10 Version 22H2 for x64-based Systems (KB5026361)
30242: Feature update to Windows 10, version 22H2
30243: Contains ModulesCountEnd: 3506
30244: Total of Function Calls: 36781
30245: SHA1: 4.7.6.50 FBC9E7794EEF888DCBE94350D8F53A8FD05BD51F
30246: fbc9e7794eef888dcbe94350d8f53a8fd05bd51f
30247: CRC32: 94780E06 33.37 MB (34,993,432 bytes)
30248: Compilation Timestamp 2023-05-26 07:25:55 UTC Signtime 26 May 2023 09:30:37
30249: Entry Point 25209520 - Contained Sections 10
30250: sha1: fbc9e7794eef888dcbe94350d8f53a8fd05bd51f
30251: sha256: c9f275808708fa8bbe56f816a6ae90f2438dd8a7085ecf8e4bbcff1e1747571
30252:
30253: Recompiled V4.7.6.50 II
30254: c:\>dir /s /o N *.dcu | find "31/05/2023" > C:\maxXbox\maxboxunitalldisk_sort.txt
30255: 31/05/2023 10:25 30,854 MathsLib.dcu
30256: 31/05/2023 09:31 7,618 NovusWinVersionUtils.dcu
30257: 31/05/2023 09:01 5,177 PJCBView.dcu
30258: 31/05/2023 09:27 104,722 uPSI_AzuliasUtils.dcu
30259: 31/05/2023 09:43 20,141 uPSI_NovusUtilities.dcu
30260: 31/05/2023 10:24 121,280 uPSI_PsAPI.dcu
30261:
30262: Recompiled V4.7.6.50 I
30263:
30264: 26/05/2023 09:25 30,854 MathsLib.dcu
30265: 26/05/2023 08:57 10,314 NovusFileUtils.dcu
30266: 26/05/2023 08:33 9,451 NovusNumUtils.dcu
30267: 26/05/2023 09:25 101,764 uPSI_AzuliasUtils.dcu
30268: 26/05/2023 09:02 12,574 uPSI_NovusUtilities.dcu
30269:
30270: 24/05/2023 17:38 71,721 uPSI_ExtCtrls2.dcu
30271: 24/05/2023 15:21 3,679 JPerson.dcu
30272: 24/05/2023 16:44 21,941 Pas2jsFileUtils.dcu
30273: 24/05/2023 22:14 33,155 RestClient.dcu
30274: 24/05/2023 15:47 19,789 uPSI_LazFileUtils.dcu
30275: 24/05/2023 17:02 110,118 uPSI_PersistSettings.dcu
30276:
30277: 25/05/2023 22:21 488,230 fMain.dcu
30278: 25/05/2023 22:04 316,943 uPSCompiler.dcu
30279: 25/05/2023 22:04 20,897 uPSPreProcessor.dcu
30280: 25/05/2023 22:04 257,656 uPSRuntime.dcu
30281: 25/05/2023 22:21 30,877 uPSUtils.dcu
30282: 25/05/2023 22:24 71,784 ALHttpClient2.dcu
30283: 25/05/2023 22:24 13,520 BlocksUnit.dcu
30284: 25/05/2023 22:24 5,720 FPCTypes.dcu
30285: 25/05/2023 22:44 30,868 MathsLib.dcu
30286: 25/05/2023 22:24 26,424 NovusStringUtils.dcu
30287: 25/05/2023 21:47 7,201 NovusUtilities.dcu
30288: 25/05/2023 21:50 2,452 NovusVariants.dcu
30289: 25/05/2023 21:58 8,942 NovusWindows.dcu
30290: 25/05/2023 13:09 6,176 OldRttiMarshal.dcu
30291: 25/05/2023 22:24 6,567 PXLTiming.dcu
30292: 25/05/2023 22:24 7,434 PythonAction.dcu
30293: 25/05/2023 22:24 278,942 PythonEngine.dcu
30294: 25/05/2023 22:24 6,446 SeSHA256.dcu
30295: 25/05/2023 22:24 126,275 SuperObject.dcu
30296: 25/05/2023 13:25 24,484 superxmlparser.dcu
30297: 25/05/2023 22:24 66,632 uPSI_ALHttpClient2.dcu
30298: 25/05/2023 16:42 100,168 uPSI_AzuliasUtils.dcu
30299: 25/05/2023 22:24 20,494 uPSI_BlocksUnit.dcu
30300: 25/05/2023 22:24 14,466 uPSI_NovusStringUtils.dcu
30301: 25/05/2023 22:21 5,662 uPSI_NovusUtilities.dcu
30302: 25/05/2023 22:21 7,167 uPSI_NovusWindows.dcu

```

```

30303: 25/05/2023 13:14 3,272 uPSI_OldRttiMarshal.dcu
30304: 25/05/2023 22:42 120,317 uPSI_PsAPI.dcu
30305: 25/05/2023 22:24 8,642 uPSI_PXLTiming.dcu
30306: 25/05/2023 22:24 175,127 uPSI_PythonEngine.dcu
30307: 25/05/2023 15:45 64,679 uPSI_superobject.dcu
30308: 25/05/2023 15:03 4,192 uPSI_superxmlparser.dcu
30309: 25/05/2023 22:24 9,515 uPSI_VelthuisFloatUtils.dcu
30310: 25/05/2023 22:24 46,226 VarPyth.dcu
30311: 25/05/2023 22:24 5,682 VelthuisFloatUtils.dcu
30312:
30313: c:\maxXbox>dir /s /o N *.dcu | find "26/05/2023" > C:\maxXbox\maxboxunitalldisk_sort.txt
30314:
30315: Recompiled V4.7.6.20 IX
30316:
30317: C:\maxXbox>dir /s /o N *.dcu | find "26/01/2023" > C:\maxXbox\maxboxunitalldisk_sort.txt
30318: 26/01/2023 16:36 30,869 MathsLib.dcu
30319: 26/01/2023 13:37 2,826 NovusConsole.dcu
30320: 26/01/2023 16:19 97,116 uPSI_AzulialUtils.dcu
30321: 26/01/2023 14:52 40,631 uPSI_HttpConnectionWinInet.dcu
30322: 26/01/2023 16:21 24,727 uPSI_RestUtils.dcu
30323: 26/01/2023 14:13 20,468 uPSI_UBigIntsV4.dcu
30324:
30325: C:\maxXbox>dir /s /o N *.dcu | find "22/01/2023" > C:\maxXbox\maxboxunitalldisk_sort.txt
30326: 22/01/2023 16:29 485,422 fMain.dcu
30327: 22/01/2023 16:31 71,769 ALHttpClient2.dcu
30328: 22/01/2023 16:31 13,520 BlocksUnit.dcu
30329: 22/01/2023 15:08 2,317 HttpConnectionFactory.dcu
30330: 22/01/2023 13:50 2,909 JsonConverter.dcu
30331: 22/01/2023 14:26 693 JsonListAdapter.dcu
30332: 22/01/2023 15:12 6,783 JsonToDataSetConverter.dcu
30333: 22/01/2023 23:21 30,866 MathsLib.dcu
30334: 22/01/2023 15:40 10,859 OldRttiUnMarshal.dcu
30335: 22/01/2023 16:31 6,566 PXLTiming.dcu
30336: 22/01/2023 16:31 7,427 PythonAction.dcu
30337: 22/01/2023 16:31 278,957 PythonEngine.dcu
30338: 22/01/2023 16:05 33,172 RestClient.dcu
30339: 22/01/2023 14:27 2,533 RestException.dcu
30340: 22/01/2023 16:31 6,446 SeSHA256.dcu
30341: 22/01/2023 16:31 126,388 SuperObject.dcu
30342: 22/01/2023 16:31 66,636 uPSI_ALHttpClient2.dcu
30343: 22/01/2023 23:20 75,718 uPSI_AzulialUtils.dcu
30344: 22/01/2023 16:31 20,496 uPSI_BlocksUnit.dcu
30345: 22/01/2023 14:21 9,758 uPSI_neuralbit.dcu
30346: 22/01/2023 16:31 8,645 uPSI_PXLTiming.dcu
30347: 22/01/2023 16:31 175,131 uPSI_PythonEngine.dcu
30348: 22/01/2023 16:55 29,665 uPSI_RestClient.dcu
30349: 22/01/2023 13:52 8,071 uPSI_RestJsonUtils.dcu
30350: 22/01/2023 16:31 9,516 uPSI_VelthuisFloatUtils.dcu
30351: 22/01/2023 16:31 46,253 VarPyth.dcu
30352: 22/01/2023 16:31 5,682 VelthuisFloatUtils.dcu
30353:
30354: 15/12/2022 14:50 485,115 fMain.dcu
30355: 15/12/2022 15:10 71,781 ALHttpClient2.dcu
30356: 15/12/2022 15:10 13,520 BlocksUnit.dcu
30357: 15/12/2022 12:12 2,593 DataSetUtils.dcu
30358: 15/12/2022 13:48 9,213 HttpConnection.dcu
30359: 15/12/2022 15:19 36,445 HTTPSender.dcu
30360: 15/12/2022 15:45 30,863 MathsLib.dcu
30361: 15/12/2022 15:10 6,568 PXLTiming.dcu
30362: 15/12/2022 15:10 7,434 PythonAction.dcu
30363: 15/12/2022 15:10 278,943 PythonEngine.dcu
30364: 15/12/2022 15:10 6,446 SeSHA256.dcu
30365: 15/12/2022 15:10 66,632 uPSI_ALHttpClient2.dcu
30366: 15/12/2022 12:17 69,868 uPSI_AzulialUtils.dcu
30367: 15/12/2022 15:10 20,494 uPSI_BlocksUnit.dcu
30368: 15/12/2022 12:17 3,523 uPSI_DataSetUtils.dcu
30369: 15/12/2022 14:16 10,563 uPSI_HttpConnection.dcu
30370: 15/12/2022 14:10 20,018 uPSI_HttpConnectionWinInet.dcu
30371: 15/12/2022 15:30 25,313 uPSI_HTTPSender.dcu
30372: 15/12/2022 15:10 8,645 uPSI_PXLTiming.dcu
30373: 15/12/2022 15:10 175,123 uPSI_PythonEngine.dcu
30374: 15/12/2022 15:10 9,511 uPSI_VelthuisFloatUtils.dcu
30375: 15/12/2022 15:10 46,228 VarPyth.dcu
30376: 15/12/2022 15:10 5,682 VelthuisFloatUtils.dcu
30377:
30378: Recompiled Dez. 4.7.6.20
30379:
30380: 07/10/2022 21:16 16,829 FindReplDlg.dcu
30381: 07/10/2022 20:53 483,995 fMain.dcu
30382: 07/10/2022 21:16 35,172 gsUtils.dcu
30383: 07/10/2022 21:16 194,742 IFSI_WinFormlpuzzle.dcu
30384: 07/10/2022 21:16 145,651 uPSI_fMain.dcu
30385: 07/10/2022 21:16 10,776 uPSI_IdPOP3.dcu
30386: 07/10/2022 21:16 5,649 uPSI_JvParsing.dcu
30387: 07/10/2022 21:16 21,122 uPSI_uTPLb_AES.dcu
30388: 07/10/2022 21:16 25,457 VListView.dcu
30389: 07/10/2022 21:16 70,846 WinForm1.dcu
30390: 07/10/2022 17:59 82,034 uPSI_Chart.dcu
30391: 07/10/2022 21:16 11,936 uPSI_SynEditMiscProcs.dcu

```

```

30392: 07/10/2022 11:08 10,972 uPSI_SynHighlighterPas.dcu
30393: 07/10/2022 17:44 100,843 uPSI_TeCanvas.dcu
30394: 07/10/2022 18:28 177,508 uPSI_TeEngine.dcu
30395: 07/10/2022 17:52 74,924 uPSI_TeeProcs.dcu
30396: 07/10/2022 21:16 105,297 AdoMain.dcu
30397: 07/10/2022 21:16 71,777 ALHttpClient2.dcu
30398: 07/10/2022 21:16 13,508 BlocksUnit.dcu
30399: 07/10/2022 21:16 11,799 frmExportMain.dcu
30400: 07/10/2022 21:16 30,871 MathsLib.dcu
30401: 07/10/2022 21:16 37,441 neuraldatasets.dcu
30402: 07/10/2022 21:16 50,133 neuralfit.dcu
30403: 07/10/2022 21:16 317,297 neuralnetworkCAI.dcu
30404: 07/10/2022 21:16 114,799 neuralvolume.dcu
30405: 07/10/2022 21:16 11,119 PSResources.dcu
30406: 07/10/2022 21:16 6,569 PXLTiming.dcu
30407: 07/10/2022 21:16 7,434 PythonAction.dcu
30408: 07/10/2022 21:16 278,943 PythonEngine.dcu
30409: 07/10/2022 21:16 6,447 SeSHA256.dcu
30410: 07/10/2022 21:16 3,879 simplecomport.dcu
30411: 07/10/2022 21:16 11,331 U_Splines.dcu
30412: 07/10/2022 21:16 66,636 uPSI_ALHttpClient2.dcu
30413: 07/10/2022 21:16 20,494 uPSI_BlocksUnit.dcu
30414: 07/10/2022 21:16 16,019 uPSI_DFFUtils.dcu
30415: 07/10/2022 11:08 12,554 uPSI_frmExportMain.dcu
30416: 07/10/2022 21:16 25,351 uPSI_JclSysUtils.dcu
30417: 07/10/2022 21:16 24,307 uPSI_neuraldatasets.dcu
30418: 07/10/2022 20:55 6,188 uPSI_PSResources.dcu
30419: 07/10/2022 21:16 8,644 uPSI_PXLTiming.dcu
30420: 07/10/2022 21:16 175,125 uPSI_PythonEngine.dcu
30421: 07/10/2022 21:16 46,231 VarPyth.dcu
30422: 07/10/2022 21:16 19,484 ViewToDoFm.dcu
30423: 07/10/2022 11:08 28,736 uconvMain.dcu
30424: 07/10/2022 16:52 63,798 uPSI_Series.dcu
30425: 07/10/2022 11:08 19,799 uPSI_uconvMain.dcu
30426: 07/10/2022 11:45 47,688 SynHighlighterPas.dcu
30427:
30428: 08/10/2022 00:43 484,364 fMain.dcu
30429: 08/10/2022 13:00 82,689 uPSI_Chart.dcu
30430: 08/10/2022 12:22 102,809 uPSI_TeCanvas.dcu
30431: 08/10/2022 12:46 75,065 uPSI_TeeProcs.dcu
30432: 08/10/2022 00:44 71,779 ALHttpClient2.dcu
30433: 08/10/2022 00:44 13,517 BlocksUnit.dcu
30434: 08/10/2022 16:05 30,866 MathsLib.dcu
30435: 08/10/2022 14:53 19,247 process.dcu
30436: 08/10/2022 16:03 13,555 PSResources.dcu
30437: 08/10/2022 00:44 6,568 PXLTiming.dcu
30438: 08/10/2022 00:44 7,433 PythonAction.dcu
30439: 08/10/2022 00:44 278,838 PythonEngine.dcu
30440: 08/10/2022 00:44 6,446 SeSHA256.dcu
30441: 08/10/2022 00:44 66,636 uPSI_ALHttpClient2.dcu
30442: 08/10/2022 00:33 37,434 uPSI_AzuliasUtils.dcu
30443: 08/10/2022 00:44 20,494 uPSI_BlocksUnit.dcu
30444: 08/10/2022 14:50 17,956 uPSI_process.dcu
30445: 08/10/2022 14:53 6,869 uPSI_PSResources.dcu
30446: 08/10/2022 00:44 8,643 uPSI_PXLTiming.dcu
30447: 08/10/2022 00:44 175,123 uPSI_PythonEngine.dcu
30448: 08/10/2022 00:44 46,252 VarPyth.dcu
30449: 08/10/2022 14:50 22,801 W32VersionInfo.dcu
30450:
30451: 09/10/2022 16:35 211,349 IFSI_SysUtils_max.dcu
30452: 09/10/2022 16:54 195,259 IFSI_WinFormlpuzzle.dcu
30453: 09/10/2022 19:16 30,861 MathsLib.dcu
30454: 09/10/2022 19:14 14,498 PSResources.dcu
30455: 09/10/2022 17:54 48,542 uPSI_AzuliasUtils.dcu
30456: 09/10/2022 18:12 7,017 uPSI_PSResources.dcu
30457:
30458: 13/10/2022 16:31 82,689 uPSI_Chart.dcu
30459: 13/10/2022 18:24 178,625 uPSI_TeEngine.dcu
30460: 13/10/2022 16:36 75,069 uPSI_TeeProcs.dcu
30461: 13/10/2022 19:00 30,862 MathsLib.dcu
30462: 13/10/2022 17:01 49,467 uPSI_AzuliasUtils.dcu
30463:
30464: 15/10/2022 12:08 348,647 TeEngine.dcu
30465: 15/10/2022 12:05 179,899 uPSI_TeEngine.dcu
30466: 15/10/2022 17:57 30,859 MathsLib.dcu
30467: 15/10/2022 17:53 59,779 neuralfit.dcu
30468: 15/10/2022 13:45 53,663 uPSI_AzuliasUtils.dcu
30469: 15/10/2022 16:33 29,161 uPSI_neuralfit.dcu
30470:
30471: 16/10/2022 17:27 484,338 fMain.dcu
30472: 16/10/2022 17:30 71,766 ALHttpClient2.dcu
30473: 16/10/2022 17:30 13,517 BlocksUnit.dcu
30474: 16/10/2022 17:25 59,939 neuralfit.dcu
30475: 16/10/2022 17:30 6,566 PXLTiming.dcu
30476: 16/10/2022 17:30 7,427 PythonAction.dcu
30477: 16/10/2022 17:30 278,951 PythonEngine.dcu
30478: 16/10/2022 17:30 6,446 SeSHA256.dcu
30479: 16/10/2022 17:30 66,636 uPSI_ALHttpClient2.dcu
30480: 16/10/2022 17:30 20,495 uPSI_BlocksUnit.dcu

```

```

30481: 16/10/2022 17:30      8,645 uPSI_PXLTiming.dcu
30482: 16/10/2022 17:30    175,131 uPSI_PythonEngine.dcu
30483: 16/10/2022 17:30    46,237 VarPyth.dcu
30484:
30485: 17/10/2022 11:11    49,666 JclNTFS2.dcu
30486: 17/10/2022 11:27    30,846 MathsLib.dcu
30487: 17/10/2022 11:26    56,533 uPSI_AzuliaUtils.dcu
30488: 17/10/2022 11:16    47,572 uPSI_JclNTFS2.dcu
30489:
30490: C:\maxXbox>dir /s /o N *.dcu | find "23/09/2022" > C:\maxXbox\maxboxunitalldisk_sort.txt
30491: 23/09/2022 16:15    483,806 fMain.dcu
30492: 23/09/2022 10:00    194,750 IFSI_WinForm1puzzle.dcu
30493: 23/09/2022 12:43    81,932 uPSI_Chart.dcu
30494: 23/09/2022 11:39    97,061 uPSI_TeCanvas.dcu
30495: 23/09/2022 11:39    177,404 uPSI_TeEngine.dcu
30496: 23/09/2022 11:20    73,888 uPSI_TeeProcs.dcu
30497: 23/09/2022 16:16    71,782 ALHttpClient2.dcu
30498: 23/09/2022 16:16    13,522 BlocksUnit.dcu
30499: 23/09/2022 17:45    30,862 MathsLib.dcu
30500: 23/09/2022 16:16    6,568 PXLTiming.dcu
30501: 23/09/2022 16:16    7,434 PythonAction.dcu
30502: 23/09/2022 16:16   278,867 PythonEngine.dcu
30503: 23/09/2022 16:06    10,517 RestUtils.dcu
30504: 23/09/2022 16:16    6,446 SeSHA256.dcu
30505: 23/09/2022 16:16    66,632 uPSI_ALHttpClient2.dcu
30506: 23/09/2022 17:45    31,846 uPSI_AzuliaUtils.dcu
30507: 23/09/2022 16:16    20,494 uPSI_BlocksUnit.dcu
30508: 23/09/2022 16:16    8,642 uPSI_PXLTiming.dcu
30509: 23/09/2022 16:16   175,123 uPSI_PythonEngine.dcu
30510: 23/09/2022 16:15    11,240 uPSI_RestUtils.dcu
30511: 23/09/2022 16:16    46,235 VarPyth.dcu
30512: 23/09/2022 15:11    63,422 uPSI_Series.dcu
30513:
30514: *****
30515: Release Notes maxXbox 4.7.6.50 May 2023 mX476
30516: *****
30517: Add 8 Units + 12 Tutorials
30518:
30519: 1527 unit uPSI_dprocess; TProcess2
30520: 1528 unit uPSI_uXmlStorage.pas
30521: 1229 unit uPSI_AsphyreTimer.pas
30522: 1230 unit uPSI_Pas2JSUtils.pas
30523: 1231 unit uPSI_pacMain; (Form1pac: TForm1pac;)
30524: 1232 unit dwsWebUtils.pas; DWS
30525: 1233 unit uPSI_dwsWebUtils.pas; DWS
30526: 1234 unit uPSI_RestUtils2.pas;
30527:
30528: Total of Function Calls: 36449
30529: SHA1: 4.7.6.50 C56F0A26DFBE706E6A1A9D0E23B6114AFCC09CDC
30530: CRC32: 29BCAE4D 33.1 MB (34,724,120 bytes)
30531: Compilation Timestamp 2023-05-21 13:42:16 UTC Signtime 21 May 2023 15:45:22
30532: Entry Point 25102976 - Contained Sections 10
30533: http://www.softwareschule.ch/maxbox_functions.txt
30534:
30535: 2023-05 Cumulative Update for Windows 10 Version 22H2 for x64-based Systems (KB5026361)
30536: Feature update to Windows 10, version 22H2
30537:
30538: *****
30539: Release Notes maxXbox 4.7.6.20 January 2023 mX476
30540: *****
30541: Add 81 Units + 25 Tutorials
30542:
30543: 1441 unit uPSI_neuralgeneric.pas; CAI
30544: 1442 unit uPSI_neuralthread.pas; CAI
30545: 1443 unit uPSI_uSysTools; TuO
30546: 1444 unit upsi_neuralsets; mX4
30547: 1445 unit uPSI_uWinNT.pas mX4
30548: 1446 unit uPSI_URungeKutta4.pas ICS
30549: 1447 unit uPSI_UrlConIcs.pas ICS
30550: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
30551: 1449 unit uPSI_Numedit2 mX4
30552: 1450 unit uPSI_PsAPI 3.pas mX4
30553: 1451 unit uPSI_SeSHA256.pas
30554: 1452 unit IdHashMessageDigest_max2;
30555: 1453 unit uPSI_BlocksUnit.pas
30556: 1454 unit uPSI_DelticsCommandLine.pas
30557: 1455 unit uPSI_DelticsStrUtils;
30558: 1456 unit uPSI_DelticsBitField;
30559: 1457 unit uPSI_DelticsSysUtils;
30560: 1458 unit uPSI_ALIniFiles2.pas
30561: 1459 unit uPSI_StarCalc2.pas
30562: 1460 unit uPSI_IdHashMessageDigest2.pas
30563: 1461 unit uPSI_U_Splines;
30564: 1462 unit uPSI_U_CoasterB.pas;
30565: 1463 unit U_SpringMass2.pas
30566: 1464 unit uPSI_MARSCoreUtils;
30567: 1465 unit uPSI_clJsonParser.pas
30568: 1466 unit uPSI_SynHighlighterPython.pas
30569: 1467 unit uPSI_DudsCommonDelphi;

```



```

30570: 1468 unit uPSI_AINNNeuron;
30571: 1469 unit uPSI_PJConsoleApp2;
30572: 1470 unit uPSI_PJPipeFilters2;
30573: 1471 unit uPSI_uHTMLBuilder;
30574: 1472 unit uPSI_PJPipe2;
30575: 1473 uPSI_WinApiDownload,
30576: 1474 uPSI_pxQRcode, //beta
30577: 1475 unit uPSI_neuralplanbuilder2
30578: 1476 unit uPSI_DelphiZXingQRcode;
30579: 1477 unit uPSI_RestJsonUtils;
30580: 1478 unit UtilsTimeCode;
30581: 1479 unit uPSC_classes2.pas; //TList
30582: 1480 unit uPSC_std2.pas
30583: 1481 unit uPSI_maxIniFiles.pas
30584: 1482 unit uROPSImports.pas
30585: 1483 unit uROPServerLink.pas
30586: 1484 unit uPSI_KLibUtils;
30587: 1485 unit uPSI_PathFunc2; //inno setup
30588: 1486 unit KLibVC_Redist.pas;
30589: 1487 unit HTTPApp2.pas;
30590: 1488 unit uPSI_XCollection2;
30591: 1489 unit uPSI_KLibWindows;
30592: 1490 unit KLibConstants;
30593: 1491 unit uPSI_AzuliaUtils.pas
30594: 1492 unit uPSI_ALHttpClient2;
30595: 1493 unit uPSI_ALWininetHttpClient2;
30596: 1494 unit uPSI_UtilsMax41.pas
30597: 1495 unit uPSI_JclSysUtils1;
30598: 1496 unit uPSI_RestUtils;
30599: 1497 unit uPSI_TeEngine2.pas
30600: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
30601: 1499 unit uPSI_PSResources.pas
30602: 1500 unit uPSI_TeCanvas2_1.pas
30603: 1501 unit uPSI_DataSetConverter4DUtil;
30604: 1502 unit uPSI_neuralfit2.pas;
30605: 1503 unit uPSI_SynCrtSock.pas
30606: 1504 uPSI_RunElevatedSupport.pas
30607: 1505 unit synTHHttpRequest.pas;
30608: 1506 unit uPSI_VelthuisFloatUtils.pas
30609: 1507 unit HttpConnection.pas
30610: 1508 unit uPSI_HttpConnectionWinInet.pas
30611: 1509 unit UHexUtils.pas
30612: 1510 unit UExeFileType.pas
30613: 1511 unit uPSI_UConsoleApp.pas
30614: 1512 unit uPSI_CompilersURunner.pas
30615: 1513 unit uPSI_HttpConnection.pas
30616: 1514 unit uPSI_DataSetUtils.pas
30617: 1515 unit uPSI_HTTPSender.pas
30618: 1516 unit AES_Cryptobox4.pas
30619: 1517 unit uPSI_JsonConverter.pas
30620: 1518 unit uPSI_RestClient.pas
30621: 1519 unit JsonToDataSetConverter;
30622: 1520 unit JsonListAdapter; (superobject)
30623: 1521 unit uPSI_OpenApiUtils.pas;
30624: 1522 unit uPSI_WinHttp_TLB.pas;
30625:
30626: Total of Function Calls: 36317
30627: SHA1: 4.7.6.20 D90222BDC9D9648E75A3A749D4C8EE1DF4655C30
30628: CRC32: 4A7231F8 32.7 MB (34,309,912 bytes)
30629: Compilation Timestamp 2023-01-24 21:15:08 UTC Signing time 24 Jan2023 22:18:14
30630: Entry Point 25025064 - Contained Sections 10
30631:
30632: Amount of Functions: 21538
30633: Amount of Procedures: 12755
30634: Amount of Constructors: 2010
30635: Amount of Destructors: 14
30636: Totals of Calls: 36317
30637: SHA1: of 4.7.6.20 D90222BDC9D9648E75A3A749D4C8EE1DF4655C30
30638: CRC32: 6CB990C1: 34309912 bytes
30639: mX4 executed: 24/01/2023 23:23:35 Runtime: 0:15:16.661 Memload: 44% use
30640:
30641: Total of Function Calls: 36317
30642: SHA1: 4.7.6.20 D90222BDC9D9648E75A3A749D4C8EE1DF4655C30
30643: CRC32: 6CB990C1 32.7 MB (34,309,912 bytes)
30644: Compilation Timestamp 2023-01-24 21:15:08 UTC Signing time 24 Jan2023 22:18:14
30645: Entry Point 25025064 - Contained Sections 10
30646: -----
30647:
30648: Total of Function Calls: 35958
30649: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
30650: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
30651: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
30652: Entry Point 24783132 - Contained Sections 10
30653: Amount of Functions: 21314
30654: Amount of Procedures: 12650
30655: Amount of Constructors: 1980
30656: Amount of Destructors: 14
30657: Totals of Calls: 35958
30658: SHA1: of 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7

```

```

30659: CRC32: 9E995FA2: 33805592 bytes
30660: mX4 executed: 17/10/2022 12:23:45 Runtime: 0:14:16.719 Memload: 36% use
30661:
30662: *****
30663: Release Notes maxBox 4.7.6.10 II November 2021 mX476
30664: *****
30665: Add 10 Units + 3 Tutorials
30666:
30667: 1441 unit uPSI_neuralgeneric.pas; CAI
30668: 1442 unit uPSI_neuralthread.pas; CAI
30669: 1443 unit uPSI_uSysTools; TuO
30670: 1444 unit upsi_neuralsets; mX4
30671: 1445 unit uPSI_uWinNT.pas mX4
30672: 1446 unit uPSI_URungeKutta4.pas ICS
30673: 1447 unit uPSI_UrlConIcs.pas ICS
30674: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
30675: 1449 unit uPSI_Numedit2 mX4
30676: 1450 unit uPSI_PsAPI_3.pas mX4
30677:
30678: Total of Function Calls: 35078
30679: SHA1: of 4.7.6.10 D4B0A36E42E9E89642A140CCEE2B7CCDDE3D041A
30680: CRC32: B8F2450F 30.6 MB (32,101,704 bytes)
30681:
30682: Amount of Functions: 20591
30683: Amount of Procedures: 12332
30684: Amount of Constructors: 1943
30685: Amount of Destructors: 14
30686: Totals of Calls: 34880
30687: SHA1: of 4.7.6.10 CF939E3A8D4723DB1DEF383C5FC961E06728C58F
30688: CRC32: 38F88218: 32022344 bytes
30689: mX4 executed: 14/11/2021 11:02:18 Runtime: 0:12:36.968 Memload: 42% use
30690:
30691: *****
30692: Tech Notes maxBox 4.7.5.90 V October 2021 mX47
30693: *****
30694: add uPSI_flcFloats.pas to complete Fundamentals5
30695: RegisterMethod: count, capacity in TNNetList
30696: with CL.Add(TNNetList) do RegisterMethod(@TNNetList.Count, 'Count');
30697: alias flots() - floattostr()
30698: Function flots( const A : Float) : String';
30699: plus resources MOON_FULL, maxbox4logo.jpg, EARTH.bmp
30700: Bugfix TSize Type
30701: CL.AddClassN(CL.FindClass('TObject'), 'TInteger'); Bigints for BigFloat
30702: - RegisterMethod(FindClass('TObject'), 'TBigFloat'), 'Procedure Assign4( A : TInteger);
30703: fix type TSize, add type TIntegerArray
30704: CL.AddTypes('TSize', record cx: Longint; cy: Longint; end); //not anymore in JvVCLUtils
30705: asize:= MakeSize(2,34);
30706: wGetTextExtentPoint32(adc, 'f', 4, asize)
30707: writeln(botostr(GetViewportExtEx( adc, asize)));
30708: Function AddLayer( pLayer : TNNetLayer) : TNNetLayer;
30709: Function AddLayer1( strData : string) : TNNetLayer;
30710: Function AddLayer2( pLayers : array of TNNetLayer) : TNNetLayer;
30711: TestConvolutionAPI Routine internal
30712:
30713: Docu of main Neural Unit for maxbox: http://www.softwareschule.ch/examples/uPSI_NeuralNetworkCAI.txt
30714: http://www.softwareschule.ch/examples/uPSI_neuralnetworkcai.txt
30715: http://www.softwareschule.ch/examples/uPSI_neuralvolume.txt
30716: http://www.softwareschule.ch/examples/uPSI_neuraldatasets.txt
30717: http://www.softwareschule.ch/examples/uPSI_neuralfit.txt
30718: Source https://github.com/joaopauloschuler/neural-api
30719:
30720: *****
30721: Procedure
30722: Proc SIZE 12456
30723: Proc *****Now the Proc list*****
30724: Procedure( ACol, ARow : Int; Items : TStrings)
30725: Procedure( Agg : TAggregate)
30726: Procedure( ASender : TComponent; const AReplyStatus : TReplyStatus)
30727: Procedure( ASender : TComponent; const AString :Str; var AMsg : TIdMessage)
30728: Procedure( ASender : TComponent; var AMsg : TIdMessage)
30729: Procedure( ASender : TObject; const ABytes : Int)
30730: Procedure( ASender : TObject; VStream : TStream)
30731: Procedure( AThread : TIdThread)
30732: Procedure( AWebModule : TComponent)
30733: Procedure( Column : TColumn)
30734: Procedure(const AUsername:Str; const APassword:Str; AAuthenticationResult: Bool)
30735: Procedure( const iStart : Int; const sText :Str)
30736: Procedure(Control:TCustomTabControl; TabIndex:Int; const Rect : TRect; Active :Bool)
30737: Procedure( Database : TDatabase; LoginParams : TStrings)
30738: Proc DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:TReconcileAction)
30739: Proc ( DATASET : TDATASET)
30740: Proc ( DataSet:TDataSet;E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
30741: Procedure( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
30742: Procedure( DataSet : TDataSet; UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction)
30743: Procedure( DATASET : TDATASET; var ACCEPT :Bool)
30744: Procedure( DBCtrlGrid : TDBCtrlGrid; Index : Int)
30745: Procedure( Done : Int)
30746: Procedure( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
30747: Procedure(HeaderControl:TCustomHeaderControl;Sect:THeaderSection;const Rect:TRect;Pressed:Bool)

```

```

30748: Procedure (HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
30749: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
30750: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;const Rect:TRect;Pressed:Bool)
30751: Procedure (HeaderControl:THeaderControl;Sectin:THeaderSection;Width:Int;State:TSectionTrackState)
30752: Procedure (Sender:TCustomListView;const ARect:TRect;Stage:TCustomDrawStage;var DefaultDraw: Bool)
30753: Procedure (Sender: TCustomListView; const ARect : TRect; var DefaultDraw :Bool)
30754: Procedure (Sender: TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
30755: Procedure (Sender: TCustomTreeView; const ARect : TRect; var DefaultDraw :Bool)
30756: Procedure (SENDER: TFIELD; const TEXT :Str)
30757: Procedure (SENDER: TFIELD; var TEXT :Str; DISPLAYTEXT :Bool)
30758: Procedure (Sender: TIdTelnet; const Buffer :Str)
30759: Procedure (Sender: TIdTelnet; Status : TIdTelnetCommand)
30760: Procedure (SENDER: TObject; ACANVAS : TCanvas; ARECT : TRect; SELECTED :Bool)
30761: Procedure (SENDER: TObject; ACANVAS : TCanvas; ARECT : TRect; STATE : TOWNERDRAWSTATE)
30762: Procedure (SENDER: TObject; ACANVAS : TCanvas; var WIDTH, HEIGHT : Int)
30763: Procedure (Sender: TObject; ACol, ARow : Longint; const Value :Str)
30764: Procedure (Sender: TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
30765: Procedure (Sender: TObject; ACol, ARow : Longint; var CanSelect :Bool)
30766: Procedure (Sender: TObject; ACol, ARow : Longint; var Value :Str)
30767: Procedure (Sender: TObject; Button : TMPBtnType)
30768: Procedure (Sender: TObject; Button : TMPBtnType; var DoDefault :Bool)
30769: Procedure (Sender: TObject; Button : TUDBtnType)
30770: Procedure (Sender: TObject; Canvas: TCanvas; PageRect:TRect; var DoneDrawing:Bool)
30771: Procedure (Sender:TObject;ClientSocket:TServerClientWinSocket;var SocketThread: TServerClientThread)
30772: Procedure (Sender: TObject; Column : TListColumn)
30773: Procedure (Sender: TObject; Column : TListColumn; Point : TPoint)
30774: Procedure (Sender: TObject; Connecting :Bool)
30775: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
DoneDraw:Bool)
30776: Procedure (Sendr:TObject;const Rect:TRect;DataCol:Int;Column:TColumn;State:TGridDrawState)
30777: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
30778: Procedure (Sender:TObject;const UserStr:str;var DateAndTime:TDateTime;var AllowChange:Bool)
30779: Procedure (Sender:TObject; E : Exception; var Handled :Bool)
30780: Procedure (Sender:TObject; FromIndex, ToIndex : Longint)
30781: Procedure (Sender:TObject; FromSection, ToSection: THeaderSection; var AllowDrag:Bool)
30782: Procedure (Sender:TObject; Index : LongInt)
30783: Procedure (Sender:TObject; Item : TListItem)
30784: Procedure (Sender:TObject; Item : TListItem; Change : TItemChange)
30785: Procedure (Sender:TObject;Item: TListItem; Change:TItemChange;var AllowChange:Boolean)
30786: Procedure (Sender:TObject; Item : TListItem; Selected :Bool)
30787: Procedure (Sender:TObject; Item : TListItem; var AllowEdit :Bool)
30788: Procedure (Sender:TObject; Item : TListItem; var S :Str)
30789: Procedure (Sender:TObject; Item1, Item2 : TListItem; Data : Int; var Compare : Int)
30790: Procedure (Sender:TObject; ModalResult : TModalResult; var CanClose :Bool)
30791: Procedure (Sender:TObject; Month : LongWord; var MonthBoldInfo : LongWord)
30792: Procedure (Sender:TObject; NewTab : Int; var AllowChange :Bool)
30793: Procedure (Sender:TObject; Node : TTreeNode)
30794: Procedure (Sender:TObject; Node : TTreeNode; var AllowChange :Bool)
30795: Procedure (Sender:TObject; Node : TTreeNode; var AllowCollapse :Bool)
30796: Procedure (Sender:TObject; Node : TTreeNode; var AllowEdit :Bool)
30797: Procedure (Sender:TObject; Node : TTreeNode; var AllowExpansion :Bool)
30798: Procedure (Sender:TObject; Node : TTreeNode; var S :Str)
30799: Procedure (Sender:TObject; Node1, Node2 : TTreeNode; Data : Int; var Compare : Int)
30800: Procedure (Sender: TObject; NumObjects, NumChars : Int; var SaveClipboard :Bool)
30801: Procedure (Sender: TObject; Rect : TRect)
30802: Procedure (Sender:TObject; Request:TWebRequest;Response:TWebResponse; var Handled:Bool)
30803: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Int;Orient:TPageScrollerOrientation;var Delta:Int)
30804: Procedure (Sender TObject; Socket : TCustomWinSocket)
30805: Procedure (Sender:TObject;Socket:TCustomWinSocket;ErrorEvent:TErrorEvent;var ErrorCode: Int)
30806: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
30807: Procedure (Sender: TObject; Socket : TSocket; var ClientSocket: TServerClientWinSocket)
30808: Procedure (SENDER: TObject; SOURCE : TMENUITEM; REBUILD :Bool)
30809: Procedure (Sender: TObject; StartPos, EndPos : Int; var AllowChange :Bool)
30810: Procedure (Sender: TObject; TabCanvas: TCanvas;R:TRect; Index: Int; Selected :Bool)
30811: Procedure (Sender: TObject; TabIndex : Int; var ImageIndex : Int)
30812: Procedure (Sender: TObject; Thread : TServerClientThread)
30813: Procedure (Sender: TObject; TickCount :Card; var Reset :Bool)
30814: Procedure (Sender: TObject; Username, Password :Str)
30815: Procedure (Sender: TObject; var AllowChange :Bool)
30816: Procedure (Sender: TObject; var AllowChange:Boolean; NewValue:SmallInt; Direction:TUpDownDirection)
30817: Procedure (Sender: TObject; var Caption :Str; var Alignment : THTMLCaptionAlignment)
30818: Procedure ( Sender : TObject; var Continue :Bool)
30819: Procedure (Sender:TObject;var dest:str;var NumRedirect:Int;var Handled:bool;var VMethod:TIdHTTPMethod)
30820: Procedure (Sender : TObject; var Username :Str)
30821: Procedure (Sender: TObject; Wnd : HWND)
30822: Procedure (Sender : TToolBar; Button : TToolButton)
30823: Procedure (Sender:TToolBar;const ARect:TRect;Stage:TCustomDrawStage;var DefaultDraw:Bool)
30824: Procedure (Sender : TToolBar; const ARect : TRect; var DefaultDraw :Bool)
30825: Procedure (Sender : TToolBar; Index : Int; var Allow :Bool)
30826: Procedure (Sender : TToolBar; Index : Int; var Button : TToolButton)
30827: Procedure (StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
30828: Procedure (StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
30829: Procedure (var FieldNames:TWideStrings;SQL:WideStr;var BindAllFields:Bool;var TableName:WideStr)
30830: Procedure (var FieldNames: TWideStrings; SQL : WideString; var TableName : WideString)
30831: procedure (Sender: TObject)
30832: procedure (Sender: TObject; var Done:Bool)
30833: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
30834: Proc _T (Name: tbtString; v: Variant);
30835: Proc AbandonSignalHandler ( RtlSigNum : Int)

```

```

30836: Proc Abort
30837: Proc About1Click( Sender : TObject)
30838: Proc Accept( Socket : TSocket)
30839: Proc AESSymmetricExecute(const plaintext, ciphertext, password:Str)
30840: Proc AESEncryptFile(const plaintext, ciphertext, password:Str)
30841: Proc AESDecryptFile(const replaintext, ciphertext, password:Str)
30842: Proc AESEncryptString(const plaintext:str;var ciphertext:str;password:Str)
30843: Proc AESDecryptString(var plaintext:str;const ciphertext:str;password:Str)
30844: Proc Add(Addend1, Addend2 : TMyBigInt)
30845: Proc ADD(const AKEY, AVALUE : VARIANT)
30846: Proc Add(const Key :Str; Value : Int)
30847: Proc ADD(const NAME, FIELDS :Str; OPTIONS : TINDEXTIONS)
30848: Proc ADD(FIELD : TFIELD)
30849: Proc ADD(ITEM : TMENUITEM)
30850: Proc ADD(POPUP : TPOPUPMENU)
30851: Proc AddCharacters( xCharacters : TCharSet)
30852: Proc AddDriver( const Name :Str; List : TStringList)
30853: Proc AddImages( Value : TCustomImageList)
30854: Proc AddIndex(const Name,Fields:str;Options:TIndexOptions;const DescFields:str)
30855: Proc AddLambdaTransitionTo( oState : TniRegularExpressionState)
30856: Proc AddLoader( Loader : TBitmapLoader)
30857: Proc ADDPARAM( VALUE : TPARAM)
30858: Proc AddPassword( const Password :Str)
30859: Proc AddStandardAlias( const Name, Path, DefaultDriver :Str)
30860: Proc AddState( oState : TniRegularExpressionState)
30861: Proc AddStrings( Strings : TStringList)
30862: Proc AddStrings(Strings: TStringList)
30863: Proc AddStrings1( Strings : TWideStrings)
30864: Proc AddStringTerm(var sString:str;const sTerm:str;const sSeparator:str)
30865: Proc AddToRecentDocs( const Filename:Str)
30866: Proc AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharSet)
30867: Proc AllFunctionsList1Click( Sender : TObject)
30868: Proc AllObjectsList1Click(Sender: TObject);
30869: Proc Allocate( AAllocateBytes : Int)
30870: Proc AllResourceList1Click(Sender: TObject);
30871: Proc AnsiAppend( var dst : Ansistr; const src : Ansistr)
30872: Proc AnsiAssign( var dst : Ansistr; var src : Ansistr)
30873: Proc AnsiDelete( var dst : Ansistr; index, count : Int)
30874: Proc AnsiFree( var s : Ansistr)
30875: Proc AnsiFromWide( var dst : Ansistr; const src : WideString)
30876: Proc AnsiInsert( var dst : Ansistr; const src : Ansistr; index : Int)
30877: Proc AnsiSetLength( var dst : Ansistr; len : Int)
30878: Proc Ansistr_to_stream( const Value : Ansistr; Destin : TStream)
30879: Proc AntiFreeze;
30880: Proc APPEND
30881: Proc Append( const S : WideString)
30882: Proc Append(S:Str);
30883: Proc AppendByte( var VBytes : TIdBytes; AByte : byte)
30884: Proc AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
30885: Proc AppendChunk( Val : OleVariant)
30886: Proc AppendData( const Data : OleVariant; HitEOF :Bool)
30887: Proc AppendStr( var Dest :Str; S :Str)
30888: Proc AppendString( var VBytes : TIdBytes; const AStr :Str; ALength : Int)
30889: Proc ApplyRange
30890: Proc Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
30891: Proc Arrange( Code : TListArrangement)
30892: Proc Assert(expr :Bool; const msg:Str);
30893: Proc Assert2(expr :Bool; const msg:Str);
30894: Proc Assign( Alist : TCustomBucketList)
30895: Proc Assign( Other : TObject)
30896: Proc Assign( Source : TDragObject)
30897: Proc Assign( Source : TPersistent)
30898: Proc Assign(Source: TPersistent)
30899: Proc Assign2(mystring, mypath:Str);
30900: Proc AssignCurValues( Source : TDataSet);
30901: Proc AssignCurValues1( const CurValues : Variant);
30902: Proc ASSIGNFIELD( FIELD : TFIELD)
30903: Proc ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
30904: Proc AssignFile(var F: Text; FileName:Str)
30905: Proc AssignFile(var F: TextFile; FileName:Str)
30906: Proc AssignFileRead(var mystring, myfilename:Str);
30907: Proc AssignFileWrite(mystring, myfilename:Str);
30908: Proc AssignTo( Other : TObject)
30909: Proc AssignValues( Value : TParameters)
30910: Proc ASSIGNVALUES( VALUE : TPARAMS)
30911: Proc AssociateExtension( IconPath, ProgramName, Path, Extension :Str)
30912: Proc Base64_to_stream( const Base64 : Ansistr; Destin : TStream)
30913: Proc Base64ToVar( NatData : Pointer; const SoapData : WideString);
30914: Proc Base64ToVar1( var V : Variant; const SoapData : WideString);
30915: Proc BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
30916: Proc BcdDivide( Dividend, Divisor :Str; var bcdOut : TBcd);
30917: Proc BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
30918: Proc BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
30919: Proc BcdDivide3( const Dividend : TBcd; const Divisor :Str; var bcdOut : TBcd);
30920: Proc BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
30921: Proc BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
30922: Proc BcdMultiply2( const bcdIn : TBcd; const StringIn :Str; var bcdOut : TBcd);
30923: Proc BcdMultiply3( StringIn1, StringIn2 :Str; var bcdOut : TBcd);
30924: Proc BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)

```

```

30925: Proc BcdToBytes( Value : TBcd; Bytes : array of byte)
30926: Proc Beep
30927: Proc BeepOk
30928: Proc BeepQuestion
30929: Proc BeepHand
30930: Proc BeepExclamation
30931: Proc BeepAsterisk
30932: Proc BeepInformation
30933: Proc BEGINDRAG(IMMEDIATE:BOOLEAN)
30934: Proc BeginLayout
30935: Proc BeginTimer( const Delay, Resolution :Card)
30936: Proc BeginUpdate
30937: Proc BeginUpdate;
30938: Proc BigScreen1Click(Sender: TObject);
30939: Proc BinToHex(Buffer: PChar; Text: PChar; BufSize: Int);
30940: Proc BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits :Bool);
30941: Proc BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits :Bool);
30942: Proc BitsToBooleans2( const Bits : Int; var B : TBooleanArray; AllBits :Bool);
30943: Proc BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits:Bool);
30944: Proc BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
30945: Proc BooleansToBits( var Dest : Byte; const B : TBooleanArray);
30946: Proc BooleansToBits1( var Dest : Word; const B : TBooleanArray);
30947: Proc BooleansToBits2( var Dest : Int; const B : TBooleanArray);
30948: Proc BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
30949: Proc BreakPointMenuClick( Sender : TObject)
30950: Proc BRINGTOFRONT
30951: Proc BringToFront;
30952: Proc btnBackClick( Sender : TObject)
30953: Proc btnBrowseClick( Sender : TObject)
30954: Proc BtnClick( Index : TNavigateBtn)
30955: Proc btnLargeIconsClick( Sender : TObject)
30956: Proc BuildAndSendRequest( AURI : TIdURI)
30957: Proc BuildCache
30958: Proc BurnMemory( var Buff, BuffLen : Int)
30959: Proc BurnMemoryStream( Destructo : TMemoryStream)
30960: Proc CalculateFirstSet
30961: Proc Cancel
30962: Proc CancelDrag;
30963: Proc CancelEdit
30964: Proc CANCELHINT
30965: Proc CancelRange
30966: Proc CancelUpdates
30967: Proc CancelWriteBuffer
30968: Proc Capture1(ADest:TStream;out VLineCount:Int;const ADelim:str;const AIsRFCMessage:Bool;
30969: Proc Capture2(ADest: TStrings; const ADelim:Str; const AIsRFCMessage:Boolean);
30970: Proc Capture3(ADest:TStrings;out VLineCount:Int;const ADelim:str;const AIsRFCMessage:Bool
30971: Proc CaptureScreenFormat(vname:Str; vextension:Str);
30972: Proc CaptureScreenPNG(vname:Str);
30973: Proc CardinalsToI64(var I: Int64; const LowPart, HighPart:Card);
30974: Proc CASCADE
30975: Proc CastNativeToSoap(Info:PTypeInfo;var SoapData:WideStr;NatData:Pointer;var IsNull:Bool)
30976: Proc CastSoapToVariant(SoapInfo:PTypeInfo;const SoapData:WideString;NatData:Pointer);
30977: Proc cbPathClick( Sender : TObject)
30978: Proc cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
30979: Proc cedebugAfterExecute( Sender : TPSScript)
30980: Proc cedebugBreakpoint(Sender: TObject;const FileName:str;Position,Row,Col:Card)
30981: Proc cedebugCompile( Sender : TPSScript)
30982: Proc cedebugExecute( Sender : TPSScript)
30983: Proc cedebugIdle( Sender : TObject)
30984: Proc cedebugLineInfo(Sender:TObject; const FileName:Str; Position,Row,Col:Card)
30985: Proc CenterHeight( const pc, pcParent : TControl)
30986: Proc CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
30987: Proc CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
30988: Proc Change
30989: Proc ChangeBiDiModeAlignment(var Alignment: TAlignment);
30990: Proc Changed
30991: Proc ChangeDir( const ADirName :Str)
30992: Proc ChangeDirUp
30993: Proc ChangeEntryTransitions( oNewState : TniRegularExpressionState)
30994: Proc ChangeLevelBy( Value : TChangeRange)
30995: Proc ChDir(const s:Str)
30996: Proc Check(Status: Int)
30997: Proc CheckCommonControl( CC : Int)
30998: Proc CHECKFIELDNAME( const FIELDNAME :Str)
30999: Proc CHECKFIELDNAMES( const FIELDNAMES :Str)
31000: Proc CheckForDisconnect(const ARaiseExceptionIfDisconnected:bool;const AIgnoreBuffer:bool)
31001: Proc CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected :Bool)
31002: Proc CheckToken( T : Char)
31003: Proc CheckToken(t:char)
31004: Proc CheckTokenSymbol( const S :Str)
31005: Proc CheckTokenSymbol(s:str)
31006: Proc CheckToolMenuDropdown( ToolButton : TToolButton)
31007: Proc Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
31008: Proc CIED65ToCIED50( var X, Y, Z : Extended)
31009: Proc CIELABToBGR( const Source, Target : Pointer; const Count :Card);
31010: Proc CipherFile1Click(Sender: TObject);
31011: Proc Clear;
31012: Proc Clear1Click( Sender : TObject)
31013: Proc ClearColor( Color : TColor)

```



```

31014: Proc CLEARITEM( AITEM : TMENUITEM)
31015: Proc ClearMapping
31016: Proc ClearSelection( KeepPrimary :Bool)
31017: Proc ClearWriteBuffer
31018: Proc Click
31019: Proc Close
31020: Proc closeMP3;
31021: Proc CloseClick( Sender : TObject)
31022: Proc CloseDatabase( Database : TDatabase)
31023: Proc CloseDataSets
31024: Proc CloseDialog
31025: Proc CloseFile( var F: Text);
31026: Proc Closure
31027: Proc CMYKToBGR( const Source,Target:Pointer; const BitsPerSample:Byte;Count:Card);
31028: Proc CMYKToBGR1( const C,M,Y,K, Target:Pointer; const BitsPerSample: Byte; Count:Card);
31029: Proc CodeCompletionList1Click( Sender : TObject)
31030: Proc ColEnter
31031: Proc Collapse
31032: Proc Collapse( Recurse :Bool)
31033: Proc ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
31034: Proc CommaSeparatedToStringList( AList : TStrings; const Value :Str)
31035: Proc CommitFreeAndNil( var Transaction : TDBXTransaction)
31036: Proc Compile1Click( Sender : TObject)
31037: Proc ComponentCount1Click(Sender: TObject);
31038: Proc Compress(azipfolder, azipfile:Str)
31039: Proc DeCompress(azipfolder, azipfile:Str)
31040: Proc XZip(azipfolder, azipfile:Str)
31041: Proc XUnZip(azipfolder, azipfile:Str)
31042: Proc Connect( const ATimeout : Int)
31043: Proc Connect( Socket : TSocket)
31044: Proc Console1Click(Sender: TObject);
31045: Proc Continue
31046: Proc ContinueCount( var Counter : TJclCounter)
31047: Proc CONTROLDESTROYED(CONTROL:TCONTROL)
31048: Proc ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : Int)
31049: Proc ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : Int)
31050: Proc ConvertImage(vsource, vdestination:Str);
31051: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
31052: Proc ConvertBitmap(vsource, vdestination:Str);
31053: Proc ConvertToGray(Cnv: TCanvas);
31054: Proc Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Int; Length : Int)
31055: Proc Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Int; Count : Int);
31056: Proc Copy1( Source : TBytes; Offset : Int; Buffer : TValueBuffer; Count : Int);
31057: Proc CopyBytesToHostLongWord( const Source:TIdBytes; const ASourceIdx: Int; var VDest:LongWord)
31058: Proc CopyBytesToHostWord( const ASource:TIdBytes; const ASourceIndex: Int; var VDest: Word)
31059: Proc CopyFrom( mbCopy : TMyBigInt)
31060: Proc CopyMemoryStream( Source, Destination : TMemoryStream)
31061: Proc CopyRect( const Dest: TRect; Canvas: TCanvas; const Source: TRect);
31062: Proc CopyTIdByteArray( const ASource: array of Byte; const ASourceIndex: Int; var VDest: array of Byte; const
ADestIndex: Int; const ALength: Int)
31063: Proc CopyTIdBytes( const ASrc:TIdBytes; const ASrcIdx: Int; var VDest:TIdBytes; const ADestIdx: Int; const
ALen: Int)
31064: Proc CopyTIdCardinal( const ASource: Card; var VDest:TIdBytes; const ADestIndex: Int)
31065: Proc CopyTIdInt64( const ASource: Int64; var VDest: TIdBytes; const ADestIndex : Int)
31066: Proc CopyTIdIPv6Address( const ASource:TIdIPv6Address; var VDest:TIdBytes; const ADestIdx: Int)
31067: Proc CopyTIdLongWord( const ASource:LongWord; var VDest:TIdBytes; const ADestIndex: Int)
31068: Proc CopyTIdNetworkLongWord( const ASource:LongWord; var VDest:TIdBytes; const ADestIndex: Int)
31069: Proc CopyTIdNetworkWord( const ASource:Word; var VDest:TIdBytes; const ADestIndex: Int)
31070: Proc CopyTIdString( const ASource:Str; var VDest:TIdBytes; const ADestIndex: Int; ALength: Int)
31071: Proc CopyTIdWord( const ASource: Word; var VDest: TIdBytes; const ADestIndex : Int)
31072: Proc CopyToClipboard
31073: Proc CountParts
31074: Proc CreateDataSet
31075: Proc CreateEmptyFile( const FileName :Str)
31076: Proc CreateFileFromString( const FileName, Data :Str)
31077: Proc CreateFromDelta( Source : TPacketDataSet)
31078: Proc CREATEHANDLE
31079: Proc CreatePipeStreams( var InPipe:TInputPipeStream; var
OutPipe:TOutputPipeStream; SecAttr:PSecurityAttributes; BufSize:Longint);
31080: Proc CreateProcAsUser( const UserDomain, UserName, Password, CommandLine :Str)
31081: Proc CreateProcAsUserEx( const UserDomain, UserNme, Passwo, CommandLine: str; const Environ:PChar)
31082: Proc CreateTable
31083: Proc CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
31084: Proc CSyntax1Click(Sender: TObject);
31085: Proc CurrencyToComp( Value : Currency; var Result : Comp)
31086: Proc CURSORPOSCHANGED
31087: Proc CutFirstDirectory( var S:Str)
31088: Proc DataBaseError( const Message:Str)
31089: Proc DateTimeToString( var Result :Str; Format :Str; DateTime : TDateTime);
31090: Proc DateTimeToString( var Result:Str; const Format:Str; DateTime: TDateTime)
31091: Proc DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
31092: Proc DateTimeToSystemTime( const DateTime: TDateTime; var SystemTime: TSystemTime);
31093: Proc DBIError(errorCode: Int)
31094: Proc DebugOutput( const AText :Str)
31095: Proc DebugLn( DebugLOGFILE:Str; Event_message:Str);
31096: Proc DebugRun1Click( Sender : TObject)
31097: Proc Dec;
31098: Proc DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
31099: Proc DecodeDate( const DateTime: TDateTime; var Year, Month, Day: Word);

```

```

31100: Proc DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
31101: Proc DecodeDateMonthWeek(const AVal:TDateTime;out AYear,AMonth,AWeekOfMonth,ADayOfWeek :Word)
31102: Proc DecodeDateTime(const AVal:TDateTime;out AYear,AMonth,ADay,AMin,AMSec,AMSec:Word)
31103: Proc DecodeDateWeek(const AValue: TDateTime;out AYear,AWeekOfYear,ADayOfWeek: Word)
31104: Proc DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
31105: Proc DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
31106: Proc DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
31107: Proc DecompileClick( Sender : TObject)
31108: Proc DefaultDrawColumnCell(const Rect:TRect;DataCol:Int;Column:TColumn;State:TGridDrawState)
31109: Proc DefaultDrawDataCell(const Rect: TRect; Field: TField; State: TGridDrawState)
31110: Proc DeferLayout
31111: Proc defFileread
31112: Proc DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
31113: Proc DelayMicroseconds( const MicroSeconds : Int)
31114: Proc Delete
31115: Proc Delete(const AFilename :Str)
31116: Proc Delete(const Index : Int)
31117: Proc DELETE(INDEX : Int)
31118: Proc Delete(Index : LongInt)
31119: Proc Delete(Node : TTreeNode)
31120: Proc Delete(var s: AnyString; ifrom, icount: Longint);
31121: Proc DeleteAlias( const Name :Str)
31122: Proc DeleteDriver( const Name :Str)
31123: Proc DeleteIndex( const Name :Str)
31124: Proc DeleteKey( const Section, Ident :Str)
31125: Proc DeleteRecords
31126: Proc DeleteRecords( AffectRecords : TAffectRecords)
31127: Proc DeleteString( var pStr :Str; const pDelStr :Str)
31128: Proc DeleteTable
31129: Proc DelphiSiteClick(Sender: TObject);
31130: Proc Deselect
31131: Proc Deselect( Node : TTreeNode)
31132: Proc DestroyComponents
31133: Proc DestroyHandle
31134: Proc Diff( var X : array of Double)
31135: Proc Diff(var X: array of Double);
31136: Proc DirCreate( const DirectoryName :Str);
31137: Proc DISABLEALIGN
31138: Proc DisableConstraints
31139: Proc Disconnect
31140: Proc Disconnect( Socket : TSocket)
31141: Proc Dispose
31142: Proc Dispose(P: PChar)
31143: Proc DivMod( Dividend : Int; Divisor : Word; var Result, Remainder : Word)
31144: Proc DoKey( Key : TDBCtrlGridKey)
31145: Proc DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
31146: Proc DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
31147: Proc Dormant
31148: Proc DoubleToBcd1( const AValue : Double; var bcd : TBcd);
31149: Proc DoubleToBytes( Value : Double; Bytes : array of byte)
31150: Proc DoubleToComp( Value : Double; var Result : Comp)
31151: Proc doWebCamPic(picname:Str); //eg: c:\mypic.png
31152: Proc Draw( Canvas : TCanvas; X, Y, Index : Int; Enabled :Bool);
31153: Proc Draw(X, Y: Int; Graphic: TGraphic);
31154: Proc Draw1(Canvas:TCanvas;X,Y,Indx:Int;ADrawingStyle:TDrawingStyl;AImageType:TImageType;Enabled:Bool);
31155: Proc DrawArrow(ACanvas:TCanvas; Direction:TScrollDirection;Location:TPoint;Size:Int)
31156: Proc DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Int; Shadow :Bool)
31157: Proc DrawChevron(ACanvas:TCanvas; Direction: TScrollDirection; Location:TPoint; Size : Int)
31158: Proc DrawColumnCell( const Rect : TRect; DataCol : Int; Column : TColumn; State : TGridDrawState)
31159: Proc DrawFocusRect(const Rect: TRect);
31160: Proc DrawHDIbToTBitmap( HDIB : THandle; Bitmap : TBitmap)
31161: Proc DRAWMENUITEM(MENUITEM: TMENUITEM; ACANVAS : TCanvas; ARECT : TRect; STATE: TOWNERDRAWSTATE)
31162: Proc DrawOverlay(Canvas:TCanvas;X,Y:Int;ImageIndex:Int;Overlay:TOverlay;Enabled:Bool);
31163: Proc DrawOverlay1(Canvas:TCanvas;X,Y:Int;ImageIndex:Int;Overlay:TOverlay;ADrawingStyle:
TDrawingStyle;AImageType:TImageType;Enabled:Bool);
31164: Proc drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: Int);
31165: Proc DrawPolyLine(const Canvas:TCanvas;var Points:TPointArray;const ClipRect: TRect)
31166: Proc DropConnections
31167: Proc DropDown
31168: Proc DumpDescription( oStrings : TStringList)
31169: Proc DumpStateTable( oStrings : TStringList)
31170: Proc EDIT
31171: Proc EditButtonClick
31172: Proc EditFont1Click( Sender : TObject)
31173: Proc Ellipse(X1, Y1, X2, Y2: Int);
31174: Proc Ellipse1( const Rect : TRect);
31175: Proc EMMS
31176: Proc Encode( ADest : TStream)
31177: Proc ENDDRAG(DROP:BOOLEAN)
31178: Proc EndEdit( Cancel :Bool)
31179: Proc EndTimer
31180: Proc EndUpdate
31181: Proc EraseSection( const Section :Str)
31182: Proc Error( const Ident :Str)
31183: Proc Error(Ident:Int)
31184: Proc ErrorFmt( const Ident :Str; const Args : array of const)
31185: Proc ErrorMessage( const Message :Str)
31186: Proc ErrorMessage(Message:str)
31187: Proc Exchange( Index1, Index2 : Int)

```

```

31188: Proc Exchange(Index1, Index2: Int);
31189: Proc Exec( FileName, Parameters, Directory :Str)
31190: Proc ExecProc
31191: Proc ExecSQL( UpdateKind : TUpdateKind)
31192: Proc Execute
31193: Proc Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
31194: Proc ExecuteAndWait( FileName :Str; Visibility : Int)
31195: Proc ExecuteCommand(executeFile, paramString:Str)
31196: Proc ExecuteShell(executeFile, paramString:Str)
31197: Proc ShellExecuteAndWait(executeFile, paramString:Str);
31198: Proc ExitThread(ExitCode: Int); stdcall;
31199: Proc ExitProcess(ExitCode: Int); stdcall;
31200: Proc Expand( AUserName :Str; AResults : TStrings)
31201: Proc Expand( Recurse :Bool)
31202: Proc ExportClipboard1Click( Sender : TObject)
31203: Proc ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
31204: Proc ExtractContentFields( Strings : TStrings)
31205: Proc ExtractCookieFields( Strings : TStrings)
31206: Proc ExtractFields(Separators,WhiteSpace:TSysCharSet; Content:PChar; Strings : TStrings)
31207: Proc ExtractHeaderFields(Separ,
    WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuotes:Bool)
31208: Proc ExtractHTTPFields(Separators,WhiteSpace: TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
31209: Proc ExtractQueryFields( Strings : TStrings)
31210: Proc FastDegToGrad
31211: Proc FastDegToRad
31212: Proc FastGradToDeg
31213: Proc FastGradToRad
31214: Proc FastRadToDeg
31215: Proc FastRadToGrad
31216: Proc FileClose( Handle : Int)
31217: Proc FileClose(handle: Int)
31218: Proc FilesFromWildcard(Dir,Mask:str;var Files:TStringList;Subdirs,ShowDirs,Multitask:Bool)
31219: Proc FileStructure( AStructure : TIdFTPDataStructure)
31220: Proc FillByte2(var X: Byte ; count: Int; value: byte)
31221: Proc FillBytes( var VBytes : TIdBytes; const ACount : Int; const AValue : Byte)
31222: Proc FillChar( Buffer : TRecordBuffer; Length : Int; value : Byte)
31223: Proc FillChar2(var X: PChar ; count: Int; value: char)
31224: Proc FillChars(var p:Str; count: Int; value: char); //fix3.8
31225: Proc FillIPList
31226: Proc FillRect(const Rect: TRect);
31227: Proc FillTStrings( AStrings : TStrings)
31228: Proc FilterOnBookmarks( Bookmarks : array of const)
31229: Proc FinalizePackage(Module: HMODULE)
31230: Proc FindClose;
31231: Proc FindClose2(var F: TSearchRec)
31232: Proc FindMatches( const sString:str; xNotify:TniRegularExpressionMatchFoundEvent);
31233: Proc FindMatches1(const sString:str;iStart:Int;xNotify:TniRegularExpressionMatchFoundEvent);
31234: Proc FindNearest( const KeyValues : array of const)
31235: Proc FinishContext
31236: Proc FIRST
31237: Proc FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
31238: Proc FloatToDecimal(var Result:TFloatRec;const Val:extend;ValueType:TFloatValue;Precis,Decs:Int);
31239: Proc FloodFill( X, Y : Int; Color : TColor; FillStyle : TFillStyle)
31240: Proc FlushSchemaCache( const TableName :Str)
31241: Proc FmtStr(var Result:Str; const Format:Str; const Args: array of const)
31242: Proc FOCUSCONTROL(CONTROL:TWINCONTROL)
31243: Proc Form1Close( Sender : TObject; var Action : TCloseAction)
31244: Proc FormActivate( Sender : TObject)
31245: Proc FormatLn(const format:Str; const args: array of const); //alias
31246: Proc FormClose( Sender : TObject; var Action : TCloseAction)
31247: Proc FormCreate( Sender : TObject)
31248: Proc FormDestroy( Sender : TObject)
31249: Proc FormKeyPress( Sender : TObject; var Key : Char)
31250: Proc FormOutput1Click(Sender: TObject);
31251: Proc FormToHtml( Form : TForm; Path :Str)
31252: Proc FrameRect(const Rect: TRect);
31253: Proc Frame3D(Canvas:TCanvas;var Rect:TRect;TopColor,BottomColor:TColor; Width : Int)
31254: Proc NotebookHandlesNeeded( Notebook : TNotebook)
31255: Proc Free( Buffer : TRecordBuffer)
31256: Proc Free( Buffer : TValueBuffer)
31257: Proc Free;
31258: Proc FreeAndNil(var Obj:TObject)
31259: Proc FreeImage
31260: Proc FreeMem(P: PChar; Size: Int)
31261: Proc FreeTreeData( Tree : TUpdateTree)
31262: Proc Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Int)
31263: Proc FullCollapse
31264: Proc FullExpand
31265: Proc GenerateDPB(sl: TStrings; var DPB:Str; var DPBLength: Short); //InterBase
31266: Proc GenerateTPB(sl: TStrings; var TPB:Str; var TPBLength: Short);
31267: Proc OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
31268: Proc Get1( AURL :Str; const AResponseContent : TStream);
31269: Proc Get1( const ASourceFile :Str; ADest : TStream; AResume :Bool);
31270: Proc Get2(const ASourceFile,ADestFile:Str;const ACanOverwrite: bool; AResume: Bool);
31271: Proc GetAliasNames( List : TStrings)
31272: Proc GetAliasParams( const AliasName :Str; List : TStrings)
31273: Proc GetApplicationsRunning( Strings : TStrings)
31274: Proc getBox(aURL, extension:Str);
31275: Proc GetCommandTypes( List : TWideStrings)

```

```

31276: Proc GetConfigParams( const Path, Section :Str; List : TStrings)
31277: Proc GetConnectionNames( List : TStrings; Driver :Str; DesignMode :Bool)
31278: Proc GetConvFamilies( out AFamilies : TConvFamilyArray)
31279: Proc GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
31280: Proc GetDatabaseNames( List : TStrings)
31281: Proc GetDataPacket( DataSet : TDataSet; var RecsOut : Int; out Data : OleVariant)
31282: Proc GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize:longWORD; var ImageSize : longWORD)
31283: Proc GetDir(d: byte; var s:Str)
31284: Proc GetDirList( const Search :Str; List : TStrings; Recursive :Bool)
31285: Proc GetDriverNames( List : TStrings)
31286: Proc GetDriverNames( List : TStrings; DesignMode :Bool)
31287: Proc GetDriverParams( const DriverName :Str; List : TStrings)
31288: Proc GetEmailsClick( Sender : TObject)
31289: Proc getEnvironmentInfo;
31290: Func getEnvironmentString:Str;
31291: Proc GetFieldNames(const DatabaseName, TableName :Str; List : TStrings)
31292: Proc GetFieldNames(const TableName :Str; List : TStrings)
31293: Proc GetFieldNames(const TableName :Str; List : TStrings);
31294: Proc GetFieldNames(const TableName : WideString; List : TWideStrings);
31295: Proc GETFIELDNAMES(LIST : TSTRINGS)
31296: Proc GetFieldNames1(const TableName :Str; List : TStrings);
31297: Proc GetFieldNames1(const TableName :Str; SchemaName :Str; List : TStrings);
31298: Proc GetFieldNames2(const TableName : WideString;SchemaName : WideString; List : TWideStrings);
31299: Proc GetFieldNames3(const TableName : WideString; List : TWideStrings);
31300: Proc GetFileAttributeList( const Items : TStrings; const Attr : Int)
31301: Proc GetFileAttributeListEx( const Items : TStrings; const Attr : Int)
31302: Proc GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
31303: Proc GetFormatSettings
31304: Proc GetFromDIB( var DIB : TBitmapInfo)
31305: Proc GetFromHDIB( HDIB : HBitmap)
31306: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
31307: Proc GetGEOMap(C_form,apath:Str; const Data:Str); //c_form: [html/json/xml]
31308: Proc GetIcon( Index : Int; Image : TIcon);
31309: Proc GetIcon1(Index:Int; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
31310: Proc GetIndexInfo( IndexName :Str)
31311: Proc GetIndexNames( const TableName, SchemaName :Str; List : TStrings);
31312: Proc GetIndexNames( List : TStrings)
31313: Proc GetIndexNames1(const TableName : WideString; List : TWideStrings);
31314: Proc GetIndexNames2(const TableName, SchemaName : WideString; List : TWideStrings);
31315: Proc GetIndexNames4(const TableName :Str; List : TStrings);
31316: Proc GetInternalResponse
31317: Proc GETITEMNAMES( LIST : TSTRINGS)
31318: Proc GetMem(P: PChar; Size: Int)
31319: Proc GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:Int;GROUPS:array of Int)
31320: Proc GetPackageDescription(ModuleName: PChar:Str)
31321: Proc GetPackageNames( List : TStrings);
31322: Proc GetPackageNames1( List : TWideStrings);
31323: Proc GetParamList( List : TList; const ParamNames : WideString)
31324: Proc GetProcedureNames( List : TStrings);
31325: Proc GetProcedureNames( List : TWideStrings);
31326: Proc GetProcedureNames1(const PackageName :Str; List : TStrings);
31327: Proc GetProcedureNames1(List : TStrings);
31328: Proc GetProcedureNames2(const PackageName, SchemaName :Str; List : TStrings);
31329: Proc GetProcedureNames3(List : TWideStrings);
31330: Proc GetProcedureNames4(const PackageName : WideString; List : TWideStrings);
31331: Proc GetProcedureNames5(const PackageName,SchemaName: WideString; List : TWideStrings);
31332: Proc GetProcedureParams(ProcedureName : WideString; List : TList);
31333: Proc GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
31334: Proc GetProcedureParams2(ProcedureName,PackageName,WideString;List:TList);
31335: Proc GetProviderNames( Names : TWideStrings);
31336: Proc GetProviderNames( Proc : TGetStrProc)
31337: Proc GetProviderNames1( Names : TStrings);
31338: Proc GetQRCode2(Width,Height:Word;Correct_Level:str;const Data:str;apath:Str);
31339: Proc GetQRCode3(Width,Height:Word;Correct_Level:str;const Data:str;apath:str);//no open image
31340: Func GetQRCode4(Width,Height:Word;Correct_Level:str;const Data:str;aformat:str):TLinearBitmap;
31341: Proc GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
31342: Proc GetSchemaNames( List : TStrings);
31343: Proc GetSchemaNames1( List : TWideStrings);
31344: Proc getScriptandRunAsk;
31345: Proc getScriptandRun(ascript:Str);
31346: Proc getScript(ascript:Str); //alias
31347: Proc getWebScript(ascript:Str); //alias
31348: Proc GetSessionNames( List : TStrings)
31349: Proc GetStoredProcNames( const DatabaseName :Str; List : TStrings)
31350: Proc GetStrings( List : TStrings)
31351: Proc GetSystemTime; stdcall;
31352: Proc GetTableNames(const DatabseName,Pattern:str;Extensins,SystemTables:Bool;List:TStrings)
31353: Proc GetTableNames( List : TStrings; SystemTables :Bool)
31354: Proc GetTableNames( List : TStrings; SystemTables :Bool);
31355: Proc GetTableNames( List : TWideStrings; SystemTables :Bool);
31356: Proc GetTableNames1(List: TStrings; SchemaName : WideString; SystemTables Boolean);
31357: Proc GetTableNames1( List : TStrings; SystemTables :Bool);
31358: Proc GetTableNames2(List: TWideStrings; SchemaName:WideString;SystemTables:Boolean);
31359: Proc GetTransitionsOn( cChar : char; oStateList : TList)
31360: Proc GetVisibleWindows( List : Tstrings)
31361: Proc GoBegin
31362: Proc GotoCurrent( DataSet : TCustomClientDataSet)
31363: Proc GotoCurrent( Table : TTable)
31364: Proc GotoEndClick(Sender: TObject);

```

```

31365: Proc GotoNearest
31366: Proc GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
    Direction: TGradientDirection)
31367: Proc HandleException( E : Exception; var Handled :Bool)
31368: Proc HANDLEMESSAGE
31369: Proc HandleNeeded;
31370: Proc Head( AURL :Str)
31371: Proc Help( var AHelpContents : TStringList; ACommand :Str)
31372: Proc HexToBinary( Stream : TStream)
31373: Proc HexToBinary(Stream:TStream)
31374: Proc HideDragImage
31375: Proc HideFormCaption( FormHandle : THandle; Hide :Bool)
31376: Proc HideTraybar
31377: Proc HideWindowForSeconds(secs: Int); ///3 seconds
31378: Proc HideWindowForSeconds2(secs: Int; apphandle, aself: TForm); ///3 seconds
31379: Proc HookOSExceptions
31380: Proc HookSignal( RtlSigNum : Int)
31381: Proc HSLToRGB( const H, S, L : Single; out R, G, B : Single);
31382: Proc HTMLSyntax1Click( Sender : TObject)
31383: Proc IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
31384: Proc IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
31385: Proc ImportfromClipboard1Click( Sender : TObject)
31386: Proc ImportfromClipboard2Click( Sender : TObject)
31387: Proc IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Int)
31388: Proc Incb(var x: byte);
31389: Proc Incb2(var x: byte; inc: integer);
31390: Proc Include1Click( Sender : TObject)
31391: Proc IncludeOFF; ///preprocessing
31392: Proc IncludeON;
31393: Proc Info1Click(Sender: TObject);
31394: Proc InitAltRecBuffers( CheckModified :Bool)
31395: Proc InitContext( Request : TWebRequest; Response : TWebResponse)
31396: Proc InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
31397: Proc InitData( ASource : TDataSet)
31398: Proc InitDelta( ADelta : TPacketDataSet);
31399: Proc InitDeltal( const ADelta : OleVariant);
31400: Proc InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
31401: Proc Initialize
31402: Proc InitializePackage(Module: HMODULE)
31403: Proc INITIATEACTION
31404: Proc initHexArray(var hexxn: THexArray); ///THexArray', 'array[0..15] of char;'
31405: Proc InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
31406: Proc InitModule( AModule : TComponent)
31407: Proc InitStdConvs
31408: Proc InitTreeData( Tree : TUpdateTree)
31409: Proc INSERT
31410: Proc Insert(Index : Int; AClass : TClass)
31411: Proc Insert(Index : Int; AComponent : TComponent)
31412: Proc Insert(Index : Int; AObject : TObject)
31413: Proc Insert(Index : Int; const S : WideString)
31414: Proc Insert(Index : Int; Image, Mask : TBitmap)
31415: Proc Insert(Index: Int; const S:Str);
31416: Proc Insert(Index: Int; S:Str);
31417: Proc Insert(s: AnyString; var s2: AnyString; iPos: Longint);
31418: Proc InsertComponent(AComponent:TComponent)
31419: Proc InsertControl(AControl: TControl);
31420: Proc InsertIcon( Index : Int; Image : TIcon)
31421: Proc InsertMasked( Index : Int; Image : TBitmap; MaskColor : TColor)
31422: Proc InsertObject( Index : Int; const S : WideString; AObject : TObject)
31423: Proc InsertObject(Index:Int;S:str;AObject:TObject)
31424: Proc Int16ToBytes( Value : SmallInt; Bytes : array of byte)
31425: Proc Int32ToBytes( Value : Int; Bytes : array of byte)
31426: Proc Int64ToBytes( Value : Int64; Bytes : array of byte)
31427: Proc I64ToCardinals(I: Int64; var LowPart, HighPart:Card);
31428: Proc InternalBeforeResolve( Tree : TUpdateTree)
31429: Proc InvalidateModuleCache
31430: Proc InvalidateTitles
31431: Proc InvalidateDateDayError( const AYear, ADayOfYear : Word)
31432: Proc InvalidateDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
31433: Proc InvalidateTimeError(const AYear, AMth, ADay, AHour, AMin, ASec, AMilSec:Word;const ABaseDate:TDateTime)
31434: Proc InvalidateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
31435: Proc InvalidateDayOfWeekInMonthError(const AYear, AMonth, ANthDayOfWeek, ADayOfWeek: Word)
31436: Proc JavaSyntax1Click(Sender: TObject);
31437: Proc JclLocalesInfoList( const Strings : TStringList; InfoType : Int)
31438: Proc KillDataChannel
31439: Proc Largefont1Click( Sender : TObject)
31440: Proc LAST
31441: Proc LaunchCpl( FileName :Str)
31442: Proc Launch( const AFile :Str)
31443: Proc LaunchFile( const AFile :Str)
31444: Proc LetFileList(FileList: TStringlist; apath:Str);
31445: Proc lineToNumber( xmemo :Str; met :Bool)
31446: Proc ListViewCustomDrawItem(Sender:TCustomListView;Item:TListItem;State:TCustomDrawState;var DefaultDraw:Bool)
31447: Proc
    ListViewCustomDrawSubItem(Sender:TCustomListView;Item:TListItem;SubItem:Int;State:TCustomDrawState;var DefaultDraw:Bool)
31448: Proc ListViewData( Sender : TObject; Item : TListItem)
31449: Proc ListViewDataFind(Sender:TObject;Find:TItemFind;const FindString:Str;const
    FindPosition:TPoint;FindData:Pointer;StartIndex:Int;Direction:TSearchDirection;Wrap:Boolean;var Index:Int)
31450: Proc ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Int)

```



```

31451: Proc ListViewDbClick( Sender : TObject)
31452: Proc ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
31453: Proc ListDLLEExports(const FileName:Str; List: TStrings);
31454: Proc Load( const WSDLFileName : WideString; Stream : TMemoryStream)
31455: Proc LoadBytecodeClick(Sender: TObject);
31456: Proc LoadFileFromResource(const FileName:Str; ms: TMemoryStream);
31457: Proc LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
31458: Proc LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
31459: Proc LoadFromFile( AFileName :Str)
31460: Proc LoadFromFile( const AFileName :Str; const AHeadersOnly :Bool)
31461: Proc LoadFromFile( const FileName :Str)
31462: Proc LOADFROMFILE( const FILENAME :Str; BLOBTYPE : TBLOBTYPE)
31463: Proc LoadFromFile( const FileName :Str; DataType : TDataType)
31464: Proc LoadFromFile( const FileName : WideString)
31465: Proc LoadFromFile( const FileName, FileType :Str; Bitmap : TLinearBitmap)
31466: Proc LoadFromFile(const AFileName:Str)
31467: Proc LoadFromFile(FileName:Str);
31468: Proc LoadFromFile(FileName:str)
31469: Proc LoadFromResourceID( Instance : THandle; ResID : Int)
31470: Proc LoadFromResourceName( Instance : THandle; const ResName :Str)
31471: Proc LoadFromStream( AStream : TStream; const AHeadersOnly :Bool)
31472: Proc LoadFromStream( const Stream : TStream)
31473: Proc LoadFromStream( S : TStream)
31474: Proc LoadFromStream(Stream: TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
31475: Proc LoadFromStream(Stream: TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
31476: Proc LoadFromStream( Stream : TStream)
31477: Proc LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
31478: Proc LoadFromStream( Stream : TStream; DataType : TDataType)
31479: Proc LoadFromStream(Stream: TStream);
31480: Proc LoadFromStream1( Stream : TSeekableStream; const FormatExt :Str);
31481: Proc LoadFromStream2( Stream : TStream; const FormatExt :Str);
31482: Proc LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
31483: Proc LoadLastFileClick( Sender : TObject)
31484: { LoadIcoToImage loads two icons from resource named NameRes,
31485:   into two image lists ALarge and ASmall}
31486: Proc LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes:Str);
31487: Proc LoadMemo
31488: Proc LoadParamsFromIniFile( FFileName : WideString)
31489: Proc Lock
31490: Proc Login
31491: Proc MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
31492: Proc MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
31493: Proc MakeCaseInsensitive
31494: Proc MakeDeterministic( var bChanged :Bool)
31495: Proc MakeGrayPal( var Palette, ColorCount : Int)
31496: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
31497: Proc MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
31498: Proc MakeComplexSound(N:Int;freqlist:TStrgs;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
31499: type MSArray = array[0..1] of TMemoryStream;
31500: Func MakeComplexSound2(N:integer {stream # to use};
  freqlist:TStrings;Duration{mSec}:Integer;pinknoise:Bool;shape:integer;Volume:TVolumeLevel):MSArray;
31501: Proc SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGp:Int);
31502: Proc SetRectComplexFormatStr( const S :Str)
31503: Proc SetPolarComplexFormatStr( const S :Str)
31504: Proc AddComplexSoundObjectToList(newf,newp,newa,news:Int; freqlist: TStrings);
31505: Proc MakeVisible
31506: Proc MakeVisible( PartialOK :Bool)
31507: Proc ManualClick( Sender : TObject)
31508: Proc MarkReachable
31509: Proc maxXbox; //shows the exe version data in a win box
31510: Proc MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
31511: Proc Memo1Change( Sender : TObject)
31512: Proc Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:str;Line,Column:Int;var
  Action:TSynReplaceAction)
31513: Proc Memo1SpecialLineColors(Sender:TObject;Line:Int;var Special:Boolean;var FG,BG:TColor)
31514: Proc Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
31515: Proc Memory1Click( Sender : TObject);
31516: Proc MERGE( MENU : TMAINMENU)
31517: Proc MergeChangeLog
31518: Proc MINIMIZE
31519: Proc MinimizeMaxbox;
31520: Proc MyCopyFile(Name1,Name2:str);
31521: Proc MkDir(const s:Str)
31522: Proc MakeDir(const s:Str);
31523: Proc ChangeDir(const s:Str);
31524: Func makeFile(const FileName:Str): Int);
31525: Proc mnuPrintFont1Click( Sender : TObject)
31526: Proc ModalStarted
31527: Proc Modified
31528: Proc ModifyAlias( Name :Str; List : TStrings)
31529: Proc ModifyDriver( Name :Str; List : TStrings)
31530: Proc MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4,Skew,Kurtosis:Extended)
31531: Proc MouseToCell( X, Y : Int; var ACol, ARow : Longint)
31532: Proc Move( CurIndex, NewIndex : Int)
31533: Proc Move(CurIndex, NewIndex: Int);
31534: Proc Move2(const Source: TByteArray; var Dest: TByteArray; Count: Int)
31535: Proc MoveChars(const ASource:str;ASourceStart:int;var ADest:str;ADestStart,ALen:int)
31536: Proc moveCube( o : TMyLabel)
31537: Proc MoveTo( Destination : LongInt; AttachMode : TAttachMode)

```

```

31538: Proc MoveTo(X, Y: Int);
31539: Proc MoveWindowOrg(DC: HDC; DX, DY: Int);
31540: Proc MovePoint(var x,y:Extended; const angle:Extended);
31541: Proc Multiply( Multiplier1, Multiplier2 : TMyBigInt);
31542: Proc Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Int);
31543: Proc MsgAbout(Handle: Int; const Msg, Caption: str; const IcoName: str='MAINICON'; Flags: DWORD=MB_OK);
31544: Proc mxButton(x,y,width,height,top,left,ahandle: Int);
31545: Proc New( Width, Height : Int; PixFormat : TPixelFormat)
31546: Proc New(P: PChar)
31547: Proc NewlClick(Sender: TObject);
31548: Proc NewInstanceClick(Sender: TObject);
31549: Proc NEXT
31550: Proc NextMonth
31551: Proc Noop
31552: Proc NormalizePath( var APath :Str)
31553: Proc ObjectBinaryToText(Input, Output: TStream)
31554: Proc ObjectBinaryToText1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
31555: Proc ObjectResourceToText(Input, Output: TStream)
31556: Proc ObjectResourceToText1(Input,Output:TStream;var OriginalFormat:TStreamOriginalFormat)
31557: Proc ObjectTextToBinary(Input, Output: TStream)
31558: Proc ObjectTextToBinary1(Input,Output: TStream;var OriginalFormat: TStreamOriginalFormat)
31559: Proc ObjectTextToResource(Input, Output: TStream)
31560: Proc ObjectTextToResource1(Input,Output:TStream;var OriginalFormat:TStreamOriginalFormat)
31561: Proc Open( const Name, Address, Service :Str; Port : Word; Block :Bool)
31562: Proc Open( const UserID : WideString; const Password : WideString);
31563: Proc Open;
31564: Proc openlClick( Sender : TObject)
31565: Proc OpenCdDrive
31566: Proc OpenCloseCdDrive( OpenMode :Bool; Drive : Char)
31567: Proc OpenCurrent
31568: Proc OpenFile(vfilenamepath:Str)
31569: Proc OpenDirectory1Click( Sender : TObject)
31570: Proc OpenDir(adir:Str);
31571: Proc OpenIndexFile( const IndexName :Str)
31572: Proc OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
    SchemID:OleVariant;DataSet:TADODataSet)
31573: Proc OpenWriteBuffer( const AThreshold : Int)
31574: Proc OptimizeMem
31575: Proc Options1( AURL :Str);
31576: Proc OutputDebugString(lpOutputString : PChar)
31577: Proc PackBuffer
31578: Proc Paint
31579: Proc PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch :Bool)
31580: Proc PaintToTBitmap( Target : TBitmap)
31581: Proc PaletteChanged
31582: Proc ParentBiDiModeChanged
31583: Proc PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
31584: Proc PasteFromClipboard;
31585: Proc PasteImage( Source : TLinearBitmap; X, Y : Int)
31586: Proc PathExtractElements(const Source:str;var Drive,Path,FileName,Ext:Str)
31587: Proc PerformEraseBackground(Control: TControl; DC: HDC);
31588: Proc PError( Text :Str)
31589: Proc Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Int);
31590: Proc Pie(X1: Int;Y1: Int;X2: Int;Y2: Int;X3: Int;Y3: Int;X4: Int;Y4: Int);
31591: Proc Play( FromFrame, ToFrame : Word; Count : Int)
31592: Proc playmp3(mpath:Str);
31593: Proc closeMP3;
31594: Proc PlayMP3lClick( Sender : TObject)
31595: Proc PointCopy( var Dest : TPoint; const Source : TPoint)
31596: Proc PointMove( var P : TPoint; const DeltaX, DeltaY : Int)
31597: Proc PolyBezier(const Points: array of TPoint);
31598: Proc PolyBezierTo(const Points: array of TPoint);
31599: Proc Polygon(const Points: array of TPoint);
31600: Proc Polyline(const Points: array of TPoint);
31601: Proc Pop
31602: Proc POPULATEOLE2MENU(SHAREDMENU:HMENU;GROUPS:array of INT;var WIDTHS:array of LONGINT)
31603: Proc PopulationVarianceAndMean(const X: TDynFloatArray;var Variance, Mean: Float)
31604: Proc POPUP( X, Y : Int)
31605: Proc PopupURL(URL : WideString);
31606: Proc POST
31607: Proc Post4(AURL:Str;const ASource: TStrings; const AResponseContent : TStream);
31608: Proc Post5( AURL :Str; const ASource, AResponseContent : TStream);
31609: Proc Post6(AURL:str;const ASource:TidMultiPartFormDataStream;AResponseContent:TStream);
31610: Proc PostUser( const Email, FirstName, LastName : WideString)
31611: Proc PostKeyEx32(key: Word; const shift: TShiftState; specialkey:Bool);
31612: Proc Pred(X: int64);
31613: Proc Prepare
31614: Proc PrepareStatement
31615: Proc PreProcessXML( AList : TStrings)
31616: Proc PreventDestruction
31617: Proc Print( const Caption :Str)
31618: Proc PrintBitmap(aGraphic: TGraphic; Title:Str);
31619: Proc printf(const format:Str; const args: array of const);
31620: Proc PrintList(Value: TStringList);
31621: Proc PrintImage(aValue:TBitmap;Style:TBitmapStyle);
31622: //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
31623: Proc PrintoutlClick( Sender : TObject)
31624: Proc ProcessHeaders
31625: Proc PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)

```

```

31626: Proc ProcessMessage( AMsg : TIdMessage; AHeaderOnly :Bool);
31627: Proc ProcessMessage1(AMsg:TIdMessage;const AStream TStream; AHeaderOnly:Bool);
31628: Proc ProcessMessage2(AMsg: TIdMessage; const AFilename :Str; AHeaderOnly :Bool);
31629: Proc ProcessMessagesOFF; //application.processmessages
31630: Proc ProcessMessagesON;
31631: Proc ProcessPath(const EditText:str;var Drive:Char;var DirPart:str;var FilePart:str)
31632: Proc ProcessPath1(const EditText:str;var Drive:Char;var DirPart:str;var FilePart:str);
31633: Proc Proclst Size is: 3797 /1415
31634: Proc procMessClick( Sender : TObject)
31635: Proc PSScriptCompile( Sender : TPSScript)
31636: Proc PSScriptExecute( Sender : TPSScript)
31637: Proc PSScriptLine( Sender : TObject)
31638: Proc Push( ABoundary :Str)
31639: Proc PushItem(Altem: Pointer)
31640: Proc Put2( AURL :Str; const ASource, AResponseContent : TStream);
31641: Proc Put2(const ASourceFile:Str; const ADestFile:Str;const AAppend:boolean);
31642: Proc PutLinuxLines(const Value:Str)
31643: Proc Quit
31644: Proc RaiseConversionError( const AText :Str);
31645: Proc RaiseConversionError1( const AText :Str; const AArgs : array of const);
31646: Proc RaiseConversionRegError( AFamily : TConvFamily; const ADescription :Str)
31647: Proc RaiseException(Ex: TIFException; Param:Str);
31648: Proc RaiseExceptionForLastCmdResult;
31649: Proc RaiseLastException;
31650: Proc RaiseException2;
31651: Proc RaiseException3(const Msg:Str);
31652: Proc RaiseExcept(const Msg:Str);
31653: Proc RaiseLastOSError
31654: Proc RaiseLastWin32;
31655: Proc RaiseLastWin32Error)
31656: Proc RaiseListError( const ATemplate :Str; const AData : array of const)
31657: Proc RandomFillStream( Stream : TMemoryStream)
31658: Proc randomize;
31659: Proc Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
31660: Proc RCS
31661: Proc Read( Socket : TSocket)
31662: Proc Readln1(var ast:Str); //of inputquery
31663: Proc ReadBlobData
31664: Proc ReadBuffer(Buffer:str;Count:LongInt)
31665: Proc ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
31666: Proc ReadSection( const Section :Str; Strings : TStrings)
31667: Proc ReadSections( Strings : TStrings)
31668: Proc ReadSections( Strings : TStrings);
31669: Proc ReadSections1( const Section :Str; Strings : TStrings);
31670: Proc ReadSectionValues( const Section :Str; Strings : TStrings)
31671: Proc ReadStream( AStream:TStream;AByteCount:LongInt;const AReadUntilDisconnect:Bool)
31672: Proc ReadStrings( ADest : TStrings; AReadLinesCount : Int)
31673: Proc ReadVersion2(aFileName:Str; aVersion : TStrings);
31674: Func ReadVersion(aFileName:Str; aVersion : TStrings):Bool;
31675: Proc Realign;
31676: Proc Rectangle(X1, Y1, X2, Y2: Int);
31677: Proc Rectangle1( const Rect : TRect);
31678: Proc RectCopy( var Dest : TRect; const Source : TRect)
31679: Proc RectFitToScreen( var R : TRect)
31680: Proc RectGrow( var R : TRect; const Delta : Int)
31681: Proc RectGrowX( var R : TRect; const Delta : Int)
31682: Proc RectGrowY( var R : TRect; const Delta : Int)
31683: Proc RectMove( var R : TRect; const DeltaX, DeltaY : Int)
31684: Proc RectMoveTo( var R : TRect; const X, Y : Int)
31685: Proc RectNormalize( var R : TRect)
31686: // TFileCallbackProc = procedure(filename:str);
31687: Proc RecurseDirectory(Dir:Str;IncludeSubs:Bool;callback: TFileCallbackProcedure);
31688: Proc RecurseDirectory2(Dir:Str; IncludeSubs :Bool);
31689: Proc RedirectTransition(oOldState:TniRegularExpressionState;oNewState:TniRegularExpressionState)
31690: Proc Refresh;
31691: Proc RefreshData( Options : TFetchOptions)
31692: Proc REFRESHLOOKUPLIST
31693: Proc regExPathfinder(Pathin, fileout, firstp, aregex, ext:Str; asort:Bool);
31694: Proc RegExPathfinder2(Pathin,fileout,firstp, aregex, ext:Str; asort, acopy:Bool);
31695: Proc RegisterAuthenticationMethod(MethodName:Str; AuthClass : TIdAuthenticationClass)
31696: Proc RegisterChanges( Value : TChangeLink)
31697: Proc RegisterConversionFormat(const AExtension:str;AConversionClass:TConversionClass)
31698: Proc RegisterFileFormat(const AExtension,ADescription:str;AGraphicClass:TGraphicClass)
31699: Proc RegisterFileFormat(Extension,AppID:str;Description:str;Executable:str;IconIndex:Int)
31700: Proc ReInitialize( ADelay :Card)
31701: Proc RELEASE
31702: Proc Remove( const AByteCount : Int)
31703: Proc REMOVE( FIELD : TFIELD)
31704: Proc REMOVE( ITEM : TMENUITEM)
31705: Proc REMOVE( POPUP : TPOPUPMENU)
31706: Proc RemoveAllPasswords
31707: Proc RemoveComponent(AComponent:TComponent)
31708: Proc RemoveDir( const ADirName :Str)
31709: Proc RemoveLambdaTransitions( var bChanged :Bool)
31710: Proc REMOVEPARAM( VALUE : TPARAM)
31711: Proc RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
31712: Proc RemoveTransitionTo1( oState : TniRegularExpressionState);
31713: Proc Rename( const ASourceFile, ADestFile :Str)
31714: Proc Rename( const FileName :Str; Reload :Bool)

```

```

31715: Proc RenameTable( const NewTableName :Str)
31716: Proc Replace( Index : Int; Image, Mask : TBitmap)
31717: Proc ReplaceClick( Sender : TObject)
31718: Proc ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
31719: Proc ReplaceDate( var DateTime: TDateTime; const NewDate: TDateTime))
31720: Proc ReplaceIcon( Index : Int; Image : TIcon)
31721: Proc ReplaceMasked( Index : Int; NewImage : TBitmap; MaskColor : TColor)
31722: Proc ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
31723: Proc ReplaceTime( var DateTime: TDateTime; const NewTime: TDateTime);
31724: Proc Requery( Options : TExecuteOptions)
31725: Proc Reset
31726: Proc ResetClick( Sender : TObject)
31727: Proc ResizeCanvas( XSiz, YSiz, XPos, YPos : Int; Color : TColor)
31728: Proc ResourceExploreClick(Sender: TObject);
31729: Proc RestoreContents
31730: Proc RestoreDefaults
31731: Proc RestoreOtherInstance( MainFormClassName, MainFormCaption :Str)
31732: Proc RetrieveHeaders
31733: Proc RevertRecord
31734: Proc RGBAToBGR( const Source, Target: Pointer; const BitsPerSample:Byte;Count :Card)
31735: Proc RGBToBGR( const Source, Target:Pointer; const BitsPerSample:Byte; Count :Card);
31736: Proc RGBToBGR1( const R,G,B,Target: Pointer; const BitsPerSample : Byte; Count :Card);
31737: Proc RGBToHSL( const R, G, B : Single; out H, S, L : Single);
31738: Proc RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
31739: Proc RGBToHSV( r, g, b : Int; var h, s, v : Int)
31740: Proc RleCompress2( Stream : TStream)
31741: Proc RleDecompress2( Stream : TStream)
31742: Proc Rmdir(const S:Str)
31743: Proc Rollback
31744: Proc Rollback( TransDesc : TTransactionDesc)
31745: Proc RollbackFreeAndNil( var Transaction : TDBXTransaction)
31746: Proc RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
31747: Proc RollbackTrans
31748: Proc RoundRect(X1, Y1, X2, Y2, X3, Y3: Int);
31749: Proc RoundToAllocGranularity64( var Value : Int64; Up :Bool)
31750: Proc RoundToAllocGranularityPtr( var Value : Pointer; Up :Bool)
31751: Proc RunDll32Internal(Wnd:HWND; const DLLName,FuncName,CmdLine:str;CmdShow : Int)
31752: procedure RunDosInMemo(DosApp: string; AMemo:TMemo););
31753: Proc S_AddMessageToStrings( AMessages : TStrings; AMsg :Str)
31754: Proc S_EBox( const AText :Str)
31755: Proc S_GetEncryptionKeys(DatTime1,DTime2:TDateTime;var StartKey:int;var MultKey:int;var AddKey:int
31756: Proc S_IBox( const AText :Str)
31757: Proc S_ReplaceChar( var cStr :Str; cOldChr, cNewChr : char)
31758: Proc S_ReplaceStringInFile(AFileName:str; ASearchString, AReplaceString :Str)
31759: Proc S_TokenInit( cBuffer : PChar; const cDelimiters :Str)
31760: Proc SampleVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
31761: Proc Save2Click( Sender : TObject)
31762: Proc Saveas3Click( Sender : TObject)
31763: Proc SavebeforeClick( Sender : TObject)
31764: Proc SaveBytesToFile(const Data: TBytes; const FileName:Str);
31765: Proc SaveCanvas2(vCanvas: TCanvas; FileName:Str);
31766: Proc SaveConfigFile
31767: Proc SaveOutputClick( Sender : TObject)
31768: Proc SaveScreenshotClick(Sender: TObject);
31769: Proc SaveLn(pathname,content:Str); //SaveLn(exepath+'mysaveIntest.txt', memo2.text);
31770: Proc SaveToClipboardFormat( var AFormat:Word; var AData: THandle; var APalette: HPALETTE)
31771: Proc SaveToClipboardFormat( var Format: Word; var Data : THandle; var APalette: HPALETTE)
31772: Proc SaveToFile( AFileName :Str)
31773: Proc SAVETOFILE( const FILENAME :Str)
31774: Proc SaveToFile( const FileName : WideString)
31775: Proc SaveToFile( const FileName : WideString; Format : TPersistFormat)
31776: Proc SaveToFile( const FileName, FileType :Str; Bitmap : TLinearBitmap)
31777: Proc SaveToFile(FileName:Str);
31778: Proc SaveToFile(FileName:str)
31779: Proc SaveToStream( AStream : TStream; const AHeadersOnly :Bool)
31780: Proc SaveToStream(OutStream TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
31781: Proc SaveToStream( S : TStream)
31782: Proc SaveToStream( Stream : TSeekableStream; const Ext :Str; Bitmap : TLinearBitmap)
31783: Proc SaveToStream( Stream : TStream)
31784: Proc SaveToStream( Stream : TStream; Format : TDataPacketFormat)
31785: Proc SaveToStream(Stream: TStream);
31786: Proc SaveToStream(Stream:TStream)
31787: Proc SaveToStream1( Stream : TSeekableStream; const FormatExt :Str);
31788: Proc SaveToStream2( Stream : TStream; const FormatExt :Str);
31789: Proc SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
31790: Proc Say(const sText:Str)
31791: Proc SBytecodeClick( Sender : TObject)
31792: Proc ScaleImage(const SourceBitmap, ResizedBitmap:TBitmap;const ScaleAmount:Double)
31793: Proc ScriptExplorerClick(Sender: TObject);
31794: Proc Scroll( Distance : Int)
31795: Proc Scroll( DX, DY : Int)
31796: Proc ScrollBy(DeltaX, DeltaY: Int);
31797: Proc SCROLLINVIEW(ACONTROL:TCONTROL)
31798: Proc ScrollTabs( Delta : Int)
31799: Proc SearchClick( Sender : TObject)
31800: Proc SearchAndOpenDoc(vfilenamepath:Str)
31801: Proc SearchAndOpenFile(vfilenamepath:Str)
31802: Proc SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr:Str)
31803: Proc SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr:Str; offset: Int);

```

```

31804: Proc SearchNext1Click( Sender : TObject)
31805: Proc Select( Node : TTreeNode; ShiftState : TShiftState);
31806: Proc Select1( const Nodes : array of TTreeNode);
31807: Proc Select2( Nodes : TList);
31808: Proc SelectNext( Direction :Bool)
31809: Proc SelectNextPage( GoForward :Bool; CheckTabVisible :Bool)
31810: Proc SelfTestPEM //unit uPSI_cPEM
31811: Proc Send( AMsg : TIdMessage)
31812: //config forst in const MAILINIFILE = 'maildef.ini';
31813: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox',
31814: Proc SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
31815: Proc SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
31816: Proc SendMsg( AMsg : TIdMessage; const AHeadersOnly :Bool)
31817: Proc SendMsg(AMsg: TIdMessage; const AHeadersOnly:Bool = False)
31818: Proc SendResponse
31819: Proc SendStream( AStream : TStream)
31820: Proc SendMCICmd(Cmd:Str); !
31821: Proc Set8087CW( NewCW : Word)
31822: Proc SetAll( One, Two, Three, Four : Byte)
31823: Proc SetAltRecBuffers( Old, New, Cur : PChar)
31824: Proc SetAppDispatcher( const ADispatcher : TComponent)
31825: Proc SetArrayLength;
31826: Proc SetArrayLength2String(arr: T2StringArray; asize1, asize2: Int); //2 dimension
31827: Proc SetArrayLength2Int(arr: T2IntArray; asize1, asize2: Int);
31828: Proc SetArrayLength2String2(arr: T2StringArray; asize1, asize2: Int); //all init
31829: Proc SetArrayLength2Int2(arr: T2IntArray; asize1, asize2: Int);
31830: Proc SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: Int);
31831: Proc Set2DimStrArray(var arr: T2StringArray; asize1, asize2: Int);
31832: Proc Set2DimIntArray(var arr: T2IntArray; asize1, asize2: Int);
31833: Proc Set3DimIntArray(var arr: T3IntArray; asize1, asize2, asize3: Int);
31834: Proc Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: Int);
31835: Proc SetAsHandle( Format : Word; Value : THandle)
31836: Proc SetBounds(ALeft, ATop, AWidth, AHeight: Int)
31837: Proc SetCaptureControl(Control: TControl);
31838: Proc SetColumnAttributes
31839: Proc SetCookieField(Values:TStrings;const ADoman,APath:str;AExpires:TDateTime;ASecure:Bool)
31840: Proc SetCustomHeader( const Name, Value :Str)
31841: Proc SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
FieldNames:Widestring)
31842: Proc SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
31843: Proc SetFocus
31844: Proc SetFocus; virtual;
31845: Proc SetInitialState
31846: Proc SetKey
31847: Proc SetLastError(ErrorCode: Int)
31848: Proc SetLength;
31849: Proc SetLength2(var S:Str; NewLength: Int);
31850: Proc SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
31851: Proc SETOLE2MENUHANDLE( HANDLE : HMENU)
31852: Proc SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
31853: Proc SETPARAMS(APOSITION,AMIN,AMAX:Int)
31854: Proc SetParams1( UpdateKind : TUpdateKind);
31855: Proc SetPassword( const Password :Str)
31856: Proc SetPointer( Ptr : Pointer; Size : Longint)
31857: Proc SetPrimalityTest( const Method : TPrimalityTestMethod)
31858: Proc SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
31859: Proc SetProvider( Provider : TComponent)
31860: Proc SetProxy( const Proxy :Str)
31861: Proc SetPSResult( var PSResult : TPSResult; Value : TObject)
31862: Proc SetRange( const StartValues, EndValues : array of const)
31863: Proc SetRangeEnd
31864: Proc SetRate( const aPercent, aYear : Int)
31865: Proc SetRate(const aPercent, aYear: Int)
31866: Proc Set_ReportMemoryLeaksOnShutdown(abo:Bool)
31867: Proc SetSafeCallExceptionMsg( Msg :Str)
31868: Proc SETSELTEXTBUF(BUFFER:PCHAR)
31869: Proc SetSize( AWidth, AHeight : Int)
31870: Proc SetSize(NewSize:LongInt)
31871: Proc SetString(var s:Str; buffer: PChar; len: Int)
31872: Proc SetStrings( List : TStrings)
31873: Proc SetText( Text : PwideChar)
31874: Proc SetText(Text: PChar);
31875: Proc SetTextBuf( Buffer : PChar)
31876: Proc SETTEXTBUF(BUFFER:PCHAR)
31877: Proc SetTick( Value : Int)
31878: Proc SetTimeout( ATimeOut : Int)
31879: Proc SetTraceEvent( Event : TDBXTraceEvent)
31880: Proc SetUserName( const UserName :Str)
31881: Proc SetWallpaper( Path :Str);
31882: Proc ShellStyle1Click(Sender: TObject);
31883: Proc SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
31884: Proc ShowFileProperties( const FileName :Str)
31885: Proc ShowInclude1Click( Sender : TObject)
31886: Proc ShowInterfaces1Click( Sender : TObject)
31887: Proc ShowLastException1Click( Sender : TObject)
31888: Proc ShowMessage( const Msg :Str)
31889: Proc ShowMessageBig(const aText :Str);
31890: Proc ShowMessageBig2(const aText :Str; autosize:Bool);
31891: Proc ShowMessageBig3(const aText :Str; fsize: byte; autosize:Bool);

```



```

31892: Proc MsgBig(const aText :Str);           //alias
31893: Proc showMessage(mytext:Str);
31894: Proc ShowMessageFmt( const Msg :Str; Params : array of const)
31895: Proc ShowMessageFmt(const Msg:Str; Params: array of const))
31896: Proc ShowMessagePos( const Msg :Str; X, Y : Int)
31897: Proc ShowMessagePos(const Msg:Str; X: Int; Y: Int))
31898: Proc ShowSearchDialog( const Directory :Str)
31899: Proc ShowSpecChars1Click( Sender : TObject)
31900: Proc ShowBitmap(bmap: TBitmap); //draw in a form!
31901: Proc ShredFile( const FileName :Str; Times : Int)
31902: Proc Shuffle(vQ: TStringList);
31903: Proc ShuffleList( var List : array of Int; Count : Int)
31904: Proc SimulateKeystroke( Key : byte; Shift : TShiftState)
31905: Proc SinCos( const Theta : Extended; var Sin, Cos : Extended)
31906: Proc SinCosE( X : Extended; out Sin, Cos : Extended)
31907: Proc Site( const ACommand :Str)
31908: Proc SkipEOL
31909: Proc Sleep( ATime :Card)
31910: Proc Sleep( milliseconds :Card)
31911: Func SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
31912: Proc Slinenumbers1Click( Sender : TObject)
31913: Proc Sort
31914: Proc SortColorArray(ColorArray:TColorArr;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
31915: Proc SoundAlarm; //beep seq
31916: Proc Speak(const sText:Str) //async like voice
31917: Proc Speak2(const sText:Str) //sync
31918: Proc Split(Str:Str; SubStr:Str; List: TStringList);
31919: Proc SplitNameValue( const Line :Str; var Name, Value :Str)
31920: Proc SplitColumns( const AData :Str; AStrings : TStringList; const ADelim :Str)
31921: Proc SplitColumnsNoTrim(const AData:str; AStrings:TStringList;const ADelim :Str)
31922: Proc SplitLines( AData : PChar; ADataSize : Int; AStrings : TStringList)
31923: Proc SplitString( const AStr, AToken :Str; var VLeft, VRight :Str)
31924: Proc SQLSyntax1Click(Sender: TObject);
31925: Proc SRand( Seed : RNG_IntType)
31926: Proc Start
31927: Proc StartCount( var Counter : TJclCounter; const Compensate :Bool)
31928: Proc StartFileFinder3(spath,aext,searchstr:Str; arecursiv:Bool; reslist: TStringList);
31929: //Ex. StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
31930: Proc StartTransaction( TransDesc : TTransactionDesc)
31931: Proc Status( var AStatusList : TStringList)
31932: Proc StatusBar1Db1Click( Sender : TObject)
31933: Proc StepIntolClick( Sender : TObject)
31934: Proc StepIt
31935: Proc StepOutlClick( Sender : TObject)
31936: Proc Stop
31937: Proc stopmp3;
31938: Proc StartWeb(aurl:Str);
31939: Proc Str(aint: Int; astr:Str); //of system
31940: Proc StrDispose( Str : PChar)
31941: Proc StrDispose(Str: PChar)
31942: Proc StrReplace(var Str:Str; Old, New:Str);
31943: Proc StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch: Bool)
31944: Proc StretchDraw(const Rect: TRect; Graphic: TGraphic);
31945: Proc StringToBytes( Value :Str; Bytes : array of byte)
31946: Proc StrSet(c : Char; I : Int; var s :Str);
31947: Proc StrSplitP(const Delimiter: Char; Input:Str; const Strings: TStringList);
31948: Proc StructureMount( APath :Str)
31949: Proc STYLECHANGED(SENDER:TObJECT)
31950: Proc Subselect( Node : TTreeNode; Validate :Bool)
31951: Proc Succ(X: int64);
31952: Proc SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares:Extended)
31953: Proc SwapChar(var X,Y: char); //swapX follows
31954: Proc SwapFloats( var X, Y : Float)
31955: Proc SwapGrid(grd: TStringGrid);
31956: Proc SwapOrd( var I, J : Byte);
31957: Proc SwapOrd( var X, Y : Int)
31958: Proc SwapOrd1( var I, J : Shortint);
31959: Proc SwapOrd2( var I, J : Smallint);
31960: Proc SwapOrd3( var I, J : Word);
31961: Proc SwapOrd4( var I, J : Int);
31962: Proc SwapOrd5( var I, J :Card);
31963: Proc SwapOrd6( var I, J : Int64);
31964: Proc SymetricCompareFiles(const plaintext, replaintext:Str)
31965: Proc Synchronizel( Method : TMethod);
31966: Proc SyntaxCheck1Click(Sender: TObject);
31967: Proc SysFreeString(const S: WideString); stdcall;
31968: Proc TakeOver( Other : TLinearBitmap)
31969: Proc Talkln(const sText:Str) //async voice
31970: Proc tbtn6resClick(Sender: TObject);
31971: Proc tbtnUseCaseClick( Sender : TObject)
31972: Proc TerminalStyle1Click(Sender: TObject);
31973: Proc Terminate
31974: Proc texSyntax1Click( Sender : TObject)
31975: Proc TextOut(X, Y: Int; Text:Str);
31976: Proc TextRect( Rect : TRect; X, Y : Int; const Text :Str);
31977: Proc TextRect(Rect: TRect; X: Int; Y: Int; const Text:Str);
31978: Proc TextRect1( var Rect : TRect; var Text :Str; TextFormat:TTextFormat);
31979: Proc TextStart
31980: Proc TILE

```

```

31981: Proc TimeStampToBytes( Value : TBcd; Bytes : array of byte)
31982: Proc TitleClick( Column : TColumn)
31983: Proc ToDo
31984: Proc Tone(500 hz, 10000 ms length);
31985: Proc toolbtnTutorialClick(Sender: TObject);
31986: Proc Tracel( AURL :Str; const AResponseContent : TStream);
31987: Proc TransferMode( ATransferMode : TIdFTPTransferMode)
31988: Proc Truncate
31989: Proc Tutorial101Click(Sender: TObject);
31990: Proc Tutorial10Statistics1Click(Sender: TObject);
31991: Proc Tutorial11Forms1Click(Sender: TObject);
31992: Proc Tutorial12SQL1Click(Sender: TObject);
31993: Proc tutorial1Click( Sender : TObject)
31994: Proc tutorial21Click( Sender : TObject)
31995: Proc tutorial31Click( Sender : TObject)
31996: Proc tutorial4Click( Sender : TObject)
31997: Proc Tutorial5Click( Sender : TObject)
31998: Proc Tutorial6Click(Sender: TObject);
31999: Proc Tutorial91Click(Sender: TObject);
32000: Proc UnhookSignal( RtlSigNum : Int; OnlyIfHooked :Bool)
32001: Proc UniqueString(var str: Ansistr)
32002: Proc UnloadLoadPackage(Module: HMODULE)
32003: Proc Unlock
32004: Proc UNMERGE( MENU : TMAINMENU)
32005: Proc UnRegisterChanges( Value : TChangeLink)
32006: Proc UnregisterConversionFamily( const AFamily : TConvFamily)
32007: Proc UnregisterConversionType( const AType : TConvType)
32008: Proc UnRegisterProvider( Prov : TCustomProvider)
32009: Proc UPDATE
32010: Proc UpdateBatch( AffectRecords : TAffectRecords)
32011: Proc UPDATECURSORPOS
32012: Proc UpdateFile
32013: Proc UpdateItems( FirstIndex, LastIndex : Int)
32014: Proc UpdateResponse( AResponse : TWebResponse)
32015: Proc UpdateScrollBar
32016: Proc UpdateView1Click( Sender : TObject)
32017: Proc UpdateExeResource(Const Source, Dest:str); ///
32018: Proc Val(const s:Str; var n, z: Int)
32019: Proc VarArraySet(c : Variant; I : Int; var s : Variant);
32020: Proc VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
32021: Proc VariantAdd( const src : Variant; var dst : Variant)
32022: Proc VariantAnd( const src : Variant; var dst : Variant)
32023: Proc VariantArrayRedim( var V : Variant; High : Int)
32024: Proc VariantCast( const src : Variant; var dst : Variant; vt : Int)
32025: Proc VariantClear( var V : Variant)
32026: Proc VariantCpy( const src : Variant; var dst : Variant)
32027: Proc VariantDiv( const src : Variant; var dst : Variant)
32028: Proc VariantMod( const src : Variant; var dst : Variant)
32029: Proc VariantMul( const src : Variant; var dst : Variant)
32030: Proc VariantOr( const src : Variant; var dst : Variant)
32031: Proc VariantPutElement( var V : Variant; const data : Variant; i1 : Int);
32032: Proc VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : Int);
32033: Proc VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : Int);
32034: Proc VariantPutElement3(var V: Variant; const data : Variant; i1, i2, i3, i4 : Int);
32035: Proc VariantPutElement4(var V:Variant;const data: Variant; i1,i2,i3, i4, i5 : Int);
32036: Proc VariantShl( const src : Variant; var dst : Variant)
32037: Proc VariantShr( const src : Variant; var dst : Variant)
32038: Proc VariantSub( const src : Variant; var dst : Variant)
32039: Proc VariantXor( const src : Variant; var dst : Variant)
32040: Proc VarCastError;
32041: Proc VarCastError1( const ASourceType, ADestType : TVarType);
32042: Proc VarInvalidOp
32043: Proc VarInvalidNullOp
32044: Proc VarOverflowError( const ASourceType, ADestType : TVarType)
32045: Proc VarRangeCheckError( const ASourceType, ADestType : TVarType)
32046: Proc VarArrayCreateError
32047: Proc VarResultCheck( AResult : HRESULT);
32048: Proc VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
32049: Proc HandleConversionException( const ASourceType, ADestType : TVarType)
32050: Func VarTypeAsText( const AType : TVarType) :Str
32051: Proc Voice(const sText:Str) //async
32052: Proc Voice2(const sText:Str) //sync
32053: Proc WaitMiliSeconds( AMSec : word)
32054: Proc WaitMS( AMSec : word);
32055: Proc WebCamPic(picname:Str); //eg: c:\mypic.png
32056: Proc WideAppend( var dst : WideString; const src : WideString)
32057: Proc WideAssign( var dst : WideString; var src : WideString)
32058: Proc WideDelete( var dst : WideString; index, count : Int)
32059: Proc WideFree( var s : WideString)
32060: Proc WideFromAnsi( var dst : WideString; const src : Ansistr)
32061: Proc WideFromPChar( var dst : WideString; src : PChar)
32062: Proc WideInsert( var dst : WideString; const src : WideString; index : Int)
32063: Proc WideSetLength( var dst : WideString; len : Int)
32064: Proc WideString2Stream( aWideString : WideString; oStream : TStream)
32065: Proc WideStringToBytes( Value : WideString; Bytes : array of byte)
32066: Proc WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
32067: Proc WinInet_HttpGet(const Url:Str; Stream:TStream);
32068: Proc HttpGet(const Url:Str; Stream:TStream);
32069: Proc WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : Int)

```

```

32070: Proc WordWrap1Click( Sender : TObject)
32071: Proc Write( const AOut :Str)
32072: Proc Write( Socket : TSocket)
32073: Proc Write(S:Str);
32074: Proc WriteBinaryStream( const Section, Name :Str; Value : TStream)
32075: Proc WriteBool( const Section, Ident :Str; Value :Bool)
32076: Proc WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow :Bool)
32077: Proc WriteBuffer(Buffer:str;Count:LongInt)
32078: Proc WriteCardinal( AValue :Card; const AConvert :Bool)
32079: Proc WriteChar( AValue : Char)
32080: Proc WriteDate( const Section, Name :Str; Value : TDateTime)
32081: Proc WriteDateTime( const Section, Name :Str; Value : TDateTime)
32082: Proc WriteFloat( const Section, Name :Str; Value : Double)
32083: Proc WriteHeader( AHeader : TStrings)
32084: Proc WriteInt( AValue : Int; const AConvert :Bool)
32085: Proc WriteInt( const Section, Ident :Str; Value : Longint)
32086: Proc WriteLn( const AOut :Str)
32087: Proc PrintLn( AOut :Str)
32088: Proc Writeln(s:Str);
32089: Proc WriteLog( const FileName, LogLine :Str)
32090: Proc WriteRFCReply( AReply : TIdRFCReply)
32091: Proc WriteRFCStrings( AStrings : TStrings)
32092: Proc WriteSmallInt( AValue : SmallInt; const AConvert :Bool)
32093: Proc WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCnt:Bool;const ASize:Int)
32094: Proc WriteString( const Section, Ident, Value :Str)
32095: Proc WriteStrings( AValue : TStrings; const AWriteLinesCount :Bool)
32096: Proc WriteTime( const Section, Name :Str; Value : TDateTime)
32097: Proc WriteObjectResourceHeader( ObjStream, Output : TStream)
32098: Proc Write16bitResourceHeader(const AName: TBytes; DataSize : Int; Output:TStream)
32099: Proc Write32bitResourceHeader(const AName: TBytes; DataSize : Int; Output:TStream)
32100: Proc WriteDataSetToCSV(DataSet: TDataSet; FileName:Str);
32101: Proc WStrSet(c : AnyString; I : Int; var s : AnyString);
32102: Proc XMLSyntax1Click(Sender: TObject);
32103: Proc XOR_Streams2( Dest, Srce : TMemoryStream)
32104: Proc XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
32105: Proc ZeroFillStream( Stream : TMemoryStream)
32106: Proc XMLSyntax1Click(Sender: TObject);
32107: Proc ZeroMemory( Ptr : Pointer; Length : Longint)
32108: procedure(Control: TWinControl; Index: Int; Rect: TRect; State: Byte)
32109: procedure(Control: TWinControl; Index: Int; var Height: Int)
32110: procedure(Sender,Source: TObject; X, Y: Int; State: TDragState; var Accept:Bool)
32111: procedure(Sender,Source: TObject; X, Y: Int)
32112: procedure(Sender,Target: TObject; X, Y: Int)
32113: procedure(Sender:TObject; ASection, AWidth: Int)
32114: procedure(Sender:TObject; ScrollCode: TScrollCode;var ScrollPos: Int)
32115: procedure(Sender:TObject; Shift: TShiftState; X, Y: Int);
32116: procedure(Sender:TObject; var Action: TCloseAction)
32117: procedure(Sender:TObject; var CanClose:Bool)
32118: procedure(Sender:TObject; var Key: Char);
32119: ProcedureName ProcedureNames ProcedureParametersCursor @
32120:
32121: *****Now Constructors constructor *****
32122: Size is: 1727 1640 1588
32123: Attach( VersionInfoData : Pointer; Size : Int)
32124: constructor Create( ABuckets : TBucketListSizes)
32125: Create(ACallBackWnd : HWND)
32126: Create(AClient : TCustomTaskDialog)
32127: Create(AClient : TIdTelnet)
32128: Create(ACollection : TCollection)
32129: Create(ACollection : TFavoriteLinkItems)
32130: Create(ACollection : TTaskDialogButtons)
32131: Create(AConnection : TIdCustomHTTP)
32132: Create(ACreateSuspended :Bool)
32133: Create(ADataset : TCustomSQLDataSet)
32134: CREATE(ADATASET : TDATASET)
32135: Create(Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
32136: Create(AGrid : TCustomDBGrid)
32137: Create(AGrid : TStringGrid; AIndex : Longint)
32138: Create(AHTTP : TIdCustomHTTP)
32139: Create(AListItems : TListItems)
32140: Create(AOnBytesRemoved : TIdBufferBytesRemoved)
32141: Create(AOnBytesRemoved : TIdBufferBytesRemoved)
32142: Create(AOwner : TCommonCalendar)
32143: Create(AOwner : TComponent)
32144: CREATE(AOWNER : TCOMPONENT)
32145: Create(AOwner : TCustomListView)
32146: Create(AOwner : TCustomOutline)
32147: Create(AOwner : TCustomRichEdit)
32148: Create(AOwner : TCustomRichEdit; AttributeType : TAttributeType)
32149: Create(AOwner : TCustomTreeView)
32150: Create(AOwner : TIdUserManager)
32151: Create(AOwner : TListItems)
32152: Create(AOwner:TObj;Handl:hDBICur;CBType:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
32153: CREATE(AOWNER : TPERSISTENT)
32154: Create(AOwner : TPersistent)
32155: Create(AOwner : TTable)
32156: Create(AOwner : TTreeNode)
32157: Create(AOwner : TWinControl; const ClassName :Str)
32158: Create(AParent : TIdCustomHTTP)

```

```

32159: Create(AParent : TUpdateTree; AResolver : TCustomResolver)
32160: Create(AProvider : TBaseProvider)
32161: Create(AProvider : TCustomProvider);
32162: Create(AProvider : TDataSetProvider)
32163: Create(ASocket : TCustomWinSocket; TimeOut : Longint)
32164: Create(ASocket : TSocket)
32165: Create(AStrings : TWideStrings)
32166: Create(AToolBar : TToolBar)
32167: Create(ATreeNodes : TTreeNodes)
32168: Create(AutoFill : Bool)
32169: Create(AWebPageInfo : TAbstractWebPageInfo)
32170: Create(AWebRequest : TWebRequest)
32171: Create(Collection : TCollection)
32172: Create(Collection : TIdMessageParts; ABody : TStrings)
32173: Create(Collection : TIdMessageParts; const AFileName : TFileName)
32174: Create(Column : TColumn)
32175: Create(const AConvFamily : TConvFamily; const ADescription : Str)
32176: Create(const AConvFamily : TConvFamily; const ADescription: str; const AFactor: Double)
32177: Create(const AConvFamily: TConvFamily; const ADescription: str; const AToCommonProc,
AFromComProc: TConversionProc)
32178: Create(const AInitialState : Bool; const AManualReset : Bool)
32179: Create(const ATabSet : TTabSet)
32180: Create(const Compensate : Bool)
32181: Create(const FileMap: TJclCustomFileMapping; Access, Size : Card; ViewOffset : Int64)
32182: Create(const FileName : Str)
32183: Create(const FileName : Str; FileMode: Card; const Name: str; Protect: Card; const MaximumSize : Int64; const
SecAttr : PSecurityAttributes);
32184: Create(const FileName : Str; FileMode : Word; WordShareDenyWrite)
32185: Create(const MaskValue : Str)
32186: Create(const Name: str; Protect: Card; const MaximumSize: Int64; const SecAttr: PSecurityAttributes)
32187: Create(const Prefix : Str)
32188: Create(const sRegularExpression : Str; xFlags : TniRegularExpressionMatchFlags)
32189: Create(const sRule : Str; xFlags : TniRegularExpressionMatchFlags)
32190: Create(const sRule : Str; xFlags : TniRegularExpressionMatchFlags)
32191: Create(CoolBar : TCoolBar)
32192: Create(CreateSuspended : Bool; ASocket : TServerClientWinSocket)
32193: Create(CreateSuspended : Bool; ASocket : TServerWinSocket)
32194: Create(DataSet: TDataSet; const Text: WideString; Options: TFilterOptions; ParserOptions: TParserOptions; const
FieldName: WideString; DepFields: TBits; FieldMap: TFieldMap)
32195: Create(DBCtrlGrid : TDBCtrlGrid)
32196: Create(DSTableProducer : TDSTableProducer)
32197: Create(DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
32198: Create(ErrorCode : DBIResult)
32199: Create(Field : TBlobField; Mode : TBlobStreamMode)
32200: Create(Grid : TCustomDBGrid; ColumnClass : TColumnClass)
32201: Create(HeaderControl : TCustomHeaderControl)
32202: Create(HTTPRequest : TWebRequest)
32203: Create(iStart : Int; sText : Str)
32204: Create(iValue : Int)
32205: Create(Kind : TMmTimerKind; Notification : TMmNotificationKind)
32206: Create(MciErrNo : MCIERROR; const Msg : Str)
32207: Create(MemoryStream: TCustomMemStream; FreeStream: Boolean; const AIndexOption: TJclMappedTextReaderIndex);
32208: Create(Message : Str; ErrorCode : DBIResult)
32209: Create(Msg : Str)
32210: Create(NativeError, Context : Str; ErrCode, PrevError : Int; E : Exception)
32211: Create(NativeError, Context : Str; ErrorCode, PreviousError : DBIResult)
32212: Create(oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
32213: Create(oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
32214: Create(oSource: TniRegularExpressionState; oDestinat: TniRegularExpressionState; xCharacts: TCharSet; bLambda: bool)
32215: Create(Owner: EDBEngineError; ErrorCode: DBIResult; NativeError: Longint; Message: PChar)
32216: Create(Owner: TCustomComboBoxEx)
32217: CREATE OWNER TINDEXDEFS; const NAME, FIELDS: Str; OPTIONS: TINDEXOPTIONS)
32218: Create(Owner: TPersistent)
32219: Create(Params: TStrings) Create( Size : Card)
32220: Create(Socket: TSocket; ServerWinSocket : TServerWinSocket)
32221: Create(StatusBar: TCustomStatusBar)
32222: Create(WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
32223: Create(WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
32224: Create(AHandle: Int)
32225: Create(AOwner: TComponent); virtual;
32226: Create(const AURI : Str)
32227: Create(FileName: str; Mode: Word)
32228: Create(Instance: THandle; ResName: str; ResType: PChar)
32229: Create(Stream : TStream)
32230: Create1( ADataset : TDataset);
32231: Create1(const FileHandle: THandle; const Name: str; Protect: Card; const MaximumSize: Int64; const
SecAttr: PSecurityAttributes);
32232: Create1( const FileName : Str; const AIndexOption : TJclMappedTextReaderIndex);
32233: Create2( Other : TObj);
32234: CreateAt( FileMap: TJclCustomFileMapping; Access, Size: Card; ViewOffset: Int64; Address: Pointer)
32235: CreateError(const anErrCode: Int; const asReplyMessage: Str; const asErrorMessage: Str)
32236: CreateFmt( MciErrNo : MCIERROR; const Msg : Str; const Args : array of const)
32237: CreateFromId( Instance: THandle; ResId: Int; ResType: PChar)
32238: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
32239: CREATENEW(AOWNER: TCOMPONENT; Dummy: Int)
32240: CreateRes(Ident : Int);
32241: CreateRes(MciErrNo : MCIERROR; Ident : Int)
32242: CreateRes(ResStringRec : PResStringRec);
32243: CreateResHelp( Ident : Int; AHelpContext : Int);

```

```

32244: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Int);
32245: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
32246: CreateSize( AWidth, AHeight : Int)
32247: Open(const Name:Str; const InheritHandle:Bool; const DesiredAccess :Card)
32248:
32249: -----
32250: unit uPSI_MathMax;
32251: -----
32252: CONSTANTS
32253: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
32254: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
32255: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
32256: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
32257: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
32258: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(Pi)
32259: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
32260: PiJ: Float = 3.1415926535897932384626433832795; // PI
32261: PI: Extended = 3.1415926535897932384626433832795;
32262: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
32263: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
32264: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
32265: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
32266: Sqrt3: Float = 1.7320508075688772935274463415059; // Sqrt(3)
32267: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
32268: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
32269: SqrtPi: Float = 1.7724538509055160272981674833411; // Sqrt(Pi)
32270: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
32271: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
32272: ThreePi: Float = 9.424779607693797153879301498385; // 3 * PI
32273: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
32274: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
32275: LnPi: Float = 1.1447298858494001741434273513531; // Ln(Pi)
32276: Log2J: Float = 0.30102999566398119521373889472449; // Log10(2)
32277: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
32278: LogPi: Float = 0.4971498726941338543512682882909; // Log10(Pi)
32279: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
32280: E: Float = 2.7182818284590452353602874713527; // Natural constant
32281: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
32282: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
32283: TwoToPower63: Float = 9223372036854775808.0; // 2^63
32284: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
32285: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
32286: RadCor : Double = 57.29577951308232; {number of degrees in a radian}
32287: StDelta : Extended = 0.00001; {delta for difference equations}
32288: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
32289: StMaxIterations : Int = 100; {max attempts for convergence}
32290:
32291: Proc SIRegister_StdConvs(CL: TPSPascalCompiler);
32292: begin
32293: MetersPerInch = 0.0254; // [1]
32294: MetersPerFoot = MetersPerInch * 12;
32295: MetersPerYard = MetersPerFoot * 3;
32296: MetersPerMile = MetersPerFoot * 5280;
32297: MetersPerNauticalMiles = 1852;
32298: MetersPerAstronomicalUnit = 1.49598E11; // [4]
32299: MetersPerLightSecond = 2.99792458E8; // [5]
32300: MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
32301: MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
32302: MetersPerCubit = 0.4572; // [6][7]
32303: MetersPerFathom = MetersPerFoot * 6;
32304: MetersPerFurlong = MetersPerYard * 220;
32305: MetersPerHand = MetersPerInch * 4;
32306: MetersPerPace = MetersPerInch * 30;
32307: MetersPerRod = MetersPerFoot * 16.5;
32308: MetersPerChain = MetersPerRod * 4;
32309: MetersPerLink = MetersPerChain / 100;
32310: MetersPerPica = MetersPerInch * 0.013837; // [7]
32311: MetersPerPica = MetersPerInch * 12;
32312:
32313: SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
32314: SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
32315: SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
32316: SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
32317: SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
32318: SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
32319: CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
32320: CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
32321: CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
32322: CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
32323: CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
32324: CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
32325: CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
32326: CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
32327:
32328: CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
32329: CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
32330: CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
32331: CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
32332: CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;

```



```

32333: CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
32334: CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
32335: CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
32336: CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
32337: CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
32338: CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
32339: CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
32340: CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
32341: CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
32342: CubicMetersPerUKGallon = 0.00454609; // [2][7]
32343: CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
32344: CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
32345: CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
32346: CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
32347: CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
32348: CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
32349: CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
32350: CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
32351:
32352: GramsPerPound = 453.59237; // [1][7]
32353: GramsPerDrams = GramsPerPound / 256;
32354: GramsPerGrains = GramsPerPound / 7000;
32355: GramsPerTons = GramsPerPound * 2000;
32356: GramsPerLongTons = GramsPerPound * 2240;
32357: GramsPerOunces = GramsPerPound / 16;
32358: GramsPerStones = GramsPerPound * 14;
32359:
32360: MaxAngle 9223372036854775808.0;
32361: MaxTanH 5678.2617031470719747459655389854;
32362: MaxFactorial( 1754);
32363: MaxFloatingPoint(1.189731495357231765085759326628E+4932);
32364: MinFloatingPoint', (3.3621031431120935062626778173218E-4932);
32365: MaxTanH( 354.89135644669199842162284618659);
32366: MaxFactorial'LongInt'( 170);
32367: MaxFloatingPointD(1.797693134862315907729305190789E+308);
32368: MinFloatingPointD(2.2250738585072013830902327173324E-308);
32369: MaxTanH( 44.361419555836499802702855773323);
32370: MaxFactorial'LongInt'( 33);
32371: MaxFloatingPointS( 3.4028236692093846346337460743177E+38);
32372: MinFloatingPointS( 1.1754943508222875079687365372222E-38);
32373: PiExt( 3.1415926535897932384626433832795);
32374: RatioDegToRad( PiExt / 180.0);
32375: RatioGradToRad( PiExt / 200.0);
32376: RatioDegToGrad( 200.0 / 180.0);
32377: RatioGradToDeg( 180.0 / 200.0);
32378: Crc16PolynomCCITT'LongWord $1021);
32379: Crc16PolynomIBM'LongWord $8005);
32380: Crc16Bits'LongInt'( 16);
32381: Crc16Bytes'LongInt'( 2);
32382: Crc16HighBit'LongWord $8000);
32383: NotCrc16HighBit', 'LongWord $7FFF);
32384: Crc32PolynomIEEE', 'LongWord $04C11DB7);
32385: Crc32PolynomCastagnoli', 'LongWord $1EDC6F41);
32386: Crc32Koopman', 'LongWord $741B8CD7);
32387: Crc32Bits', 'LongInt'( 32);
32388: Crc32Bytes', 'LongInt'( 4);
32389: Crc32HighBit', 'LongWord $80000000);
32390: NotCrc32HighBit', 'LongWord $7FFFFFFF);
32391:
32392: MinByte = Low(Byte);
32393: MaxByte = High(Byte);
32394: MinWord = Low(Word);
32395: //MaxWord = High(Word);
32396: MaxWord2 = High(Word);
32397: MinShortInt = Low(ShortInt);
32398: MaxShortInt = High(ShortInt);
32399: MinSmallInt = Low(SmallInt);
32400: MaxSmallInt = High(SmallInt);
32401: MinLongWord = LongWord(Low(LongWord));
32402: MaxLongWord = LongWord(High(LongWord));
32403: MinLongInt = LongInt(Low(LongInt));
32404: MaxLongInt = LongInt(High(LongInt));
32405: MinInt64 = Int64(Low(Int64));
32406: MaxInt64 = Int64(High(Int64));
32407: MinInt = Int(Low(Int));
32408: MaxInt = Int(High(Int));
32409: MinCardinal = Cardinal(Low(Cardinal));
32410: MaxCardinal = Cardinal(High(Cardinal));
32411: MinNativeUInt = NativeUInt(Low(NativeUInt));
32412: MaxNativeUInt = NativeUInt(High(NativeUInt));
32413: MinNativeInt = NativeInt(Low(NativeInt));
32414: MaxNativeInt = NativeInt(High(NativeInt));
32415: Func CosH( const Z : Float ) : Float;
32416: Func SinH( const Z : Float ) : Float;
32417: Func TanH( const Z : Float ) : Float;
32418:
32419: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
32420: InvLn2 = 1.44269504088896340736; { 1/Ln(2) }
32421: InvLn10 = 0.43429448190325182765; { 1/Ln(10) }

```

```

32422: TwoPi      = 6.28318530717958647693; { 2*Pi }
32423: PiDiv2     = 1.57079632679489661923; { Pi/2 }
32424: SqrtPi     = 1.77245385090551602730; { Sqrt(Pi) }
32425: Sqrt2Pi    = 2.50662827463100050242; { Sqrt(2*Pi) }
32426: InvSqrt2Pi = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
32427: LnSqrt2Pi  = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
32428: Ln2PiDiv2  = 0.91893853320467274178; { Ln(2*Pi)/2 }
32429: Sqrt2      = 1.41421356237309504880; { Sqrt(2) }
32430: Sqrt2Div2  = 0.70710678118654752440; { Sqrt(2)/2 }
32431: Gold       = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
32432: CGold      = 0.38196601125010515179; { 2 - GOLD }
32433: MachEp     = 2.220446049250313E-16; { 2^(-52) }
32434: MaxNum     = 1.797693134862315E+308; { 2^1024 }
32435: MinNum     = 2.225073858507202E-308; { 2^(-1022) }
32436: MaxLog     = 709.7827128933840;
32437: MinLog     = -708.3964185322641;
32438: MaxFac     = 170;
32439: MaxGam     = 171.624376956302;
32440: MaxLgm     = 2.556348E+305;
32441: SingleCompareDelta = 1.0E-34;
32442: DoubleCompareDelta = 1.0E-280;
32443: { $IFDEF CLR }
32444: ExtendedCompareDelta = DoubleCompareDelta;
32445: { $ELSE }
32446: ExtendedCompareDelta = 1.0E-4400;
32447: { $ENDIF }
32448: Bytes1KB   = 1024;
32449: Bytes1MB   = 1024 * Bytes1KB;
32450: Bytes1GB   = 1024 * Bytes1MB;
32451: Bytes64KB  = 64 * Bytes1KB;
32452: Bytes64MB  = 64 * Bytes1MB;
32453: Bytes2GB   = 2 * LongWord(Bytes1GB);
32454: clBlack32' = $FF000000 );
32455: clDimGray32' = $FF3F3F3F );
32456: clGray32' = $FF7F7F7F );
32457: clLightGray32' = $FFBFBFBF );
32458: clWhite32' = $FFFFFFF );
32459: clMaroon32' = $FF7F0000 );
32460: clGreen32' = $FF007F00 );
32461: clOlive32' = $FF7F7F00 );
32462: clNavy32' = $FF00007F );
32463: clPurple32' = $FF7F007F );
32464: clTeal32' = $FF007F7F );
32465: clRed32' = $FFFF0000 );
32466: clLime32' = $FF00FF00 );
32467: clYellow32' = $FFFFFF00 );
32468: clBlue32' = $FF0000FF );
32469: clFuchsia32' = $FFFF00FF );
32470: clAqua32' = $FF00FFFF );
32471: clAliceBlue32' = $FFF0F8FF );
32472: clAntiqueWhite32' = $FFFAEBD7 );
32473: clAquamarine32' = $FFF7FFD4 );
32474: clAzure32' = $FFF0FFFF );
32475: clBeige32' = $FFF5F5DC );
32476: clBisque32' = $FFFE4C4C );
32477: clBlancheDalmont32' = $FFFFEBCD );
32478: clBlueViolet32' = $FF8A2BE2 );
32479: clBrown32' = $FFA52A2A );
32480: clBurlyWood32' = $FFDEB887 );
32481: clCadetblue32' = $FF5F9EA0 );
32482: clChartreuse32' = $FFF7FF00 );
32483: clChocolate32' = $FFD2691E );
32484: clCoral32' = $FFF7F50 );
32485: clCornFlowerBlue32' = $FF6495ED );
32486: clCornsilk32' = $FFFFFF8DC );
32487: clCrimson32' = $FFDC143C );
32488: clDarkBlue32' = $FF00008B );
32489: clDarkCyan32' = $FF008B8B );
32490: clDarkGoldenRod32' = $FFB8860B );
32491: clDarkGray32' = $FFA9A9A9 );
32492: clDarkGreen32' = $FF006400 );
32493: clDarkGrey32' = $FFA9A9A9 );
32494: clDarkKhaki32' = $FFBDB76B );
32495: clDarkMagenta32' = $FF8B008B );
32496: clDarkOliveGreen32' = $FF556B2F );
32497: clDarkOrange32' = $FFFF8C00 );
32498: clDarkOrchid32' = $FF9932CC );
32499: clDarkRed32' = $FF8B0000 );
32500: clDarkSalmon32' = $FFE9967A );
32501: clDarkSeaGreen32' = $FF8FBC8F );
32502: clDarkSlateBlue32' = $FF483D8B );
32503: clDarkSlateGray32' = $FF2F4F4F );
32504: clDarkSlateGrey32' = $FF2F4F4F );
32505: clDarkTurquoise32' = $FF00CED1 );
32506: clDarkViolet32' = $FF9400D3 );
32507: clDeepPink32' = $FFFF1493 );
32508: clDeepSkyBlue32' = $FF00BFFF );
32509: clDodgerBlue32' = $FF1E90FF );
32510: clFireBrick32' = $FFB22222 );

```

```
32511: clFloralWhite32', $FFFFFFA0 );
32512: clGainsBoro32', $FFDCDCDC );
32513: clGhostWhite32', $FFF8F8FF );
32514: clGold32', $FFFFD700 );
32515: clGoldenRod32', $FFDAA520 );
32516: clGreenYellow32', $FFADFF2F );
32517: clGrey32', $FF808080 );
32518: clHoneyDew32', $FFF0FFF0 );
32519: clHotPink32', $FFFF69B4 );
32520: clIndianRed32', $FFCD5C5C );
32521: clIndigo32', $FF4B0082 );
32522: clIvory32', $FFFFFFF0 );
32523: clKhaki32', $FF0E68C );
32524: clLavender32', $FFE6E6FA );
32525: clLavenderBlush32', $FFFFFF0F );
32526: clLawnGreen32', $FF7CFC00 );
32527: clLemonChiffon32', $FFFFFACD );
32528: clLightBlue32', $FFADD8E6 );
32529: clLightCoral32', $FFF08080 );
32530: clLightCyan32', $FFE0FFFF );
32531: clLightGoldenRodYellow32', $FFFAFAD2 );
32532: clLightGreen32', $FF90EE90 );
32533: clLightGrey32', $FFD3D3D3 );
32534: clLightPink32', $FFFFB6C1 );
32535: clLightSalmon32', $FFFA07A );
32536: clLightSeagreen32', $FF20B2AA );
32537: clLightSkyblue32', $FF87CEFA );
32538: clLightSlategray32', $FF778899 );
32539: clLightSlategrey32', $FF778899 );
32540: clLightSteelblue32', $FFB0C4DE );
32541: clLightYellow32', $FFFFFFE0 );
32542: clLtGray32', $FFC0C0C0 );
32543: clMedGray32', $FFA0A0A4 );
32544: clDkGray32', $FF808080 );
32545: clMoneyGreen32', $FFC0DDC0 );
32546: clLegacySkyBlue32', $FFA6CAFA );
32547: clCream32', $FFFFF0 );
32548: clLimeGreen32', $FF32CD32 );
32549: clLinen32', $FFFAF0E6 );
32550: clMediumAquamarine32', $FF66CDAA );
32551: clMediumBlue32', $FF0000CD );
32552: clMediumOrchid32', $FFBA55D3 );
32553: clMediumPurple32', $FF9370DB );
32554: clMediumSeaGreen32', $FF3CB371 );
32555: clMediumSlateBlue32', $FF7B68EE );
32556: clMediumSpringGreen32', $FF00FA9A );
32557: clMediumTurquoise32', $FF48D1CC );
32558: clMediumVioletRed32', $FFC71585 );
32559: clMidnightBlue32', $FF191970 );
32560: clMintCream32', $FFF5FFFA );
32561: clMistyRose32', $FFFE4E1 );
32562: clMoccasin32', $FFFE4B5 );
32563: clNavajoWhite32', $FFFFDEAD );
32564: clOldLace32', $FFFD5E6 );
32565: clOliveDrab32', $FF6B8E23 );
32566: clOrange32', $FFFA500 );
32567: clOrangeRed32', $FFF4500 );
32568: clOrchid32', $FFDA70D6 );
32569: clPaleGoldenRod32', $FFEE8AA );
32570: clPaleGreen32', $FF98FB98 );
32571: clPaleTurquoise32', $FFAFEEEE );
32572: clPaleVioletred32', $FFDB7093 );
32573: clPapayaWhip32', $FFFFEFD5 );
32574: clPeachPuff32', $FFFFDAB9 );
32575: clPeru32', $FFCD853F );
32576: clPlum32', $FFDDA0DD );
32577: clPowderBlue32', $FFB0E0E6 );
32578: clRosyBrown32', $FFBC8F8F );
32579: clRoyalBlue32', $FF4169E1 );
32580: clSaddleBrown32', $FF8B4513 );
32581: clSalmon32', $FFFA8072 );
32582: clSandyBrown32', $FFFA4A60 );
32583: clSeaGreen32', $FF2E8B57 );
32584: clSeaShell32', $FFFFFF5EE );
32585: clSienna32', $FFA0522D );
32586: clSilver32', $FFC0C0C0 );
32587: clSkyblue32', $FF87CEEB );
32588: clSlateBlue32', $FF6A5ACD );
32589: clSlateGray32', $FF708090 );
32590: clSlateGrey32', $FF708090 );
32591: clSnow32', $FFFFFFAFA );
32592: clSpringgreen32', $FF00FF7F );
32593: clSteelblue32', $FF4682B4 );
32594: clTan32', $FFD2B48C );
32595: clThistle32', $FFD8BFD8 );
32596: clTomato32', $FFFF6347 );
32597: clTurquoise32', $FF40E0D0 );
32598: clViolet32', $FFEE82EE );
32599: clWheat32', $FFF5DEB3 );
```

```

32600: clWhitesmoke32', $FFF5F5F5 ));
32601: clYellowgreen32', $FF9ACD32 ));
32602: clTrWhite32', $FFFFFFF ));
32603: clTrBlack32', $F000000 ));
32604: clTrRed32', $FFFF0000 ));
32605: clTrGreen32', $F00FF00 ));
32606: clTrBlue32', $F0000FF ));
32607: // Fixed point math constants
32608: FixedOne = $10000; FixedHalf = $7FFF;
32609: FixedPI = Round(PI * FixedOne);
32610: FixedToFloat = 1/FixedOne;
32611:
32612: Special Types
32613: *****
32614: type Complex = record //for complex numbers
32615:   X, Y : Float; end;
32616: type TComplex', 'record Form : ComplexForm; X : Float; Y : Float; R : '
32617:   + Float; Theta : Float; end;
32618: type TVector = array of Float;
32619: TIntVector = array of Int;
32620: TCompVector = array of Complex;
32621: TBoolVector = array of Boolean;
32622: TStrVector = array of String;
32623: TMatrix = array of TVector;
32624: TIntMatrix = array of TIntVector;
32625: TCompMatrix = array of TCompVector;
32626: TBoolMatrix = array of TBoolVector;
32627: TStrMatrix = array of TStrVector;
32628: TByteArray = array[0..32767] of byte; !
32629: THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
32630: TBitmapStyle = (bsNormal, bsCentered, bsStretched);
32631: T2StringArray = array of array of string;
32632: T2IntArray = array of array of Int;
32633: AddTypeS('INT_PTR', 'Int
32634: AddTypeS('LONG_PTR', 'Int
32635: AddTypeS('UINT_PTR', 'Cardinal
32636: AddTypeS('ULONG_PTR', 'Cardinal
32637: AddTypeS('DWORD_PTR', 'ULONG_PTR
32638: TIntDynArray', 'array of Int
32639: TCardinalDynArray', 'array of Cardinal
32640: TWordDynArray', 'array of Word
32641: TSmallIntDynArray', 'array of SmallInt
32642: TByteDynArray', 'array of Byte
32643: TShortIntDynArray', 'array of ShortInt
32644: TInt64DynArray', 'array of Int64
32645: TLongWordDynArray', 'array of LongWord
32646: TSingleDynArray', 'array of Single
32647: TDoubleDynArray', 'array of Double
32648: TBooleanDynArray', 'array of Boolean
32649: TStringDynArray', 'array of string
32650: TWideStringDynArray', 'array of WideString
32651: TDynByteArray = array of Byte;
32652: TDynShortintArray = array of Shortint;
32653: TDynSmallintArray = array of Smallint;
32654: TDynWordArray = array of Word;
32655: TDynIntArray = array of Int;
32656: TDynLongintArray = array of Longint;
32657: TDynCardinalArray = array of Cardinal;
32658: TDynInt64Array = array of Int64;
32659: TDynExtendedArray = array of Extended;
32660: TDynDoubleArray = array of Double;
32661: TDynSingleArray = array of Single;
32662: TDynFloatArray = array of Float;
32663: TDynPointerArray = array of Pointer;
32664: TDynStringArray = array of string;
32665: TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
32666:   ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
32667: TSynSearchOptions = set of TSynSearchOption;
32668: TFloat = single
32669: Float = double
32670:
32671: /* Project:IFSI_WinFormIppuzzle.pas BaseInclude RunTimeLib for maXbox *: pas_includebox.inc
32672: -----
32673: Proc drawPolygon(vPoints: TXVector; cFrm: TForm);
32674: Proc drawPlot(vPoints: TXVector; cFrm: TForm; vcolor: Int);
32675: Proc SaveCanvas(vCanvas: TCanvas; FileName:Str);
32676: Proc SaveCanvas2(vCanvas: TCanvas; FileName:Str);
32677: Func CheckStringSum(vstring:Str): Int;
32678: Func HexToInt(HexNum:Str): LongInt;
32679: Func IntToBin(Int: Int):Str;
32680: Func BinToInt(Binary:Str): Int;
32681: Func HexToBin(HexNum:Str):Str; external2
32682: Func BinToHex(Binary:Str):Str;
32683: Func IntToFloat(i: Int): double;
32684: Func AddThousandSeparator(S:Str; myChr: Char):Str;
32685: Func Max3(const X,Y,Z: Int): Int;
32686: Proc Swap(var X,Y: char); // faster without inline
32687: Proc ReverseString(var S:Str);
32688: Func CharToHexStr(Value: Char):Str;

```

```

32689: Func CharToUnicode(Value: Char):Str;
32690: Func Hex2Dec(Value: Str002): Byte;
32691: Func HexStrCodeToStr(Value:Str):Str;
32692: Func HexToStr(i: Int; value:Str):Str;
32693: Func UnicodeToStr(Value:Str):Str;
32694: Func CRC16(statement:Str):Str;
32695: Func SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2:Str):Str;
32696: Proc SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr:Str);
32697: Proc SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr:Str; offset: Int);
32698: Proc ExecuteCommand(executeFile, paramString:Str);
32699: Proc ShellExecuteAndWait(executeFile, paramString:Str);
32700: Func ExecConsoleApp(const AppName, Parameters:Str; AppOutput: TStrings): DWORD;
32701: Proc SearchAndOpenDoc(vfilenamepath:Str);
32702: Proc ShowInterfaces(myFile:Str);
32703: Func Fact2(av: Int): extended;
32704: Func BoolToStr(B:Bool):Str;
32705: Func GCD(x, y : LongInt) : LongInt;
32706: Func LCM(m,n: longint): longint;
32707: Func GetASCII:Str;
32708: Func GetItemHeight(Font: TFont): Int;
32709: Func myPlaySound(s: pchar; flag,syncflag: Int):Bool;
32710: Func myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
32711: Func getHINSTANCE: longword;
32712: Func getHMODULE: longword;
32713: Func GetASCII:Str;
32714: Func ByteIsOk(const AByte:Str; var VB: Byte):Bool;
32715: Func WordIsOk(const AWord:Str; var VW: Word):Bool;
32716: Func TwentyFourBitValueIsOk(const AValue:Str; var VI: Int):Bool;
32717: Func LongIsOk(const ALong:Str; var VC:Card):Bool;
32718: Func SafeStr(const s:Str):Str;
32719: Func ExtractUrlPath(const FileName:Str):Str;
32720: Func ExtractUrlName(const FileName:Str):Str;
32721: Func IsInternet:Bool;
32722: Func RotateLeft1Bit_u32( Value: uint32): uint32;
32723: Proc LinearRegression(const KnownY:array of Double;const KnownX:array of Double;NData:Int;var
LF:TStLinEst;ErrorStats:Bool);
32724: Proc getEnvironmentInfo;
32725: Proc AntiFreeze;
32726: Func GetCPUSpeed: Double;
32727: Func IsVirtualPcGuest :Bool;
32728: Func IsVmWareGuest :Bool;
32729: Proc StartSerialDialog;
32730: Func IsWoW64:Bool;
32731: Func IsWow64String(var s:Str):Bool;
32732: Proc StartThreadDemo;
32733: Func RGB(R,G,B: Byte): TColor;
32734: Func Sendln(amess:Str):Bool;
32735: Proc maxbox;
32736: Func AspectRatio(aWidth, aHeight: Int):Str;
32737: Func wget(aURL, afile:Str):Bool;
32738: Proc PrintList(Value: TStringList);
32739: Proc PrintImage(aValue: TBitmap; Style: TBitmapStyle);
32740: Proc getEnvironmentInfo;
32741: Proc AntiFreeze;
32742: Func getBitmap(apat:Str): TBitmap;
32743: Proc ShowMessageBig(const aText :Str);
32744: Func YesNoDialog(const ACaption, AMsg:Str):Bool;
32745: Proc SetArrayLength2String(arr: T2StringArray; asize1, asize2: Int);
32746: Proc SetArrayLength2Int(arr: T2IntArray; asize1, asize2: Int);
32747: //Func myStrToBytes(const Value:Str): TBytes;
32748: //Func myBytesToStr(const Value: TBytes):Str;
32749: Func SaveAsExcelFile(AGrid:TStringGrid;ASheetName,AFileName:Str;open:boolean):Bool;
32750: Func getBitmap(apat:Str): TBitmap;
32751: Proc ShowMessageBig(const aText :Str);
32752: Func StrToBytes(const Value:Str): TBytes;
32753: Func BytesToStr(const Value: TBytes):Str;
32754: Func SaveAsExcelFile(AGrid: TStringGrid;ASheetName,AFileName:Str;open:boolean):Bool;
32755: Func ReverseDNSLookup(const IPAdrs:Str;const DNSServer:Str;Timeout,Retris:Int;var HostName:Str):Bool;
32756: Func FindInPaths(const fileName, paths :Str) :Str;
32757: Proc initHexArray(var hexn: THexArray);
32758: Func josephusG(n,k: Int; var graphout:Str): Int;
32759: Func isPowerof2(num: int64):Bool;
32760: Func powerOf2(exponent: Int): int64;
32761: Func getBigPI:Str;
32762: Proc MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:Str);
32763: Func GetASCIILine:Str;
32764: Proc MakeComplexSound(N:Int{stream # to use};freqlist:TStrings;Duration{mSec}: Int;
pinknoise:Bool; shape: Int; Volume: TVolumeLevel);
32766: Proc SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:Int);
32767: Proc AddComplexSoundObjectToList(newf,newp,newa,news:Int; freqlist: TStrings);
32768: Func mapFunc(ax, in_min, in_max, out_min, out_max: Int): Int;
32769: Func mapmax(ax, in_min, in_max, out_min, out_max: Int): Int;
32770: Func isKeyPressed:Bool;
32771: Func KeyPress:Bool;
32772: Proc StrSplitP(const Delimiter: Char; Input:Str; const Strings: TStrings);
32773: Func ReadReg(Base: HKEY; KeyName, ValueName:Str):Str;
32774: Func ReadRegistry(Base: HKEY; KeyName, ValueName:Str):Str;
32775: Func GetOSName:Str;
32776: Func GetOSVersion:Str;

```



```

32777: Func GetOSNumber:Str;
32778: Func getEnvironmentString:Str;
32779: Proc StrReplace(var Str:Str; Old, New:Str);
32780: Proc SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
32781: Func getTeamViewerID:Str;
32782: Proc RecurseDirectory(Dir:Str; IncludeSubs:boolean; callback:TFileCallbackProcedure);
32783: Proc RecurseDirectory2(Dir:Str; IncludeSubs:Bool);
32784: Proc WinInet HttpGet(const Url:Str; Stream:TStream);
32785: Proc GetQRCode2(Width,Height: Word;Correct_Level:Str;const Data:Str;apath:Str);
32786: Func StartSocketService:Bool;
32787: Proc StartSocketServiceForm;
32788: Func GetFileList(FileList: TStringlist; apath:Str): TStringlist;
32789: Func GetFileList1(apath:Str): TStringlist;
32790: Proc LetFileList(FileList: TStringlist; apath:Str);
32791: Proc StartWeb(aurl:Str);
32792: Func GetTodayFiles(startDir, amask:Str): TStringlist;
32793: Func PortTCPisOpen(dwPort: Word; ipAddress:Str):Bool;
32794: Func JavaHashCode(val:Str): Int;
32795: Proc PostKeyEx32(key: Word; const shift: TShiftState; specialkey:Bool);
32796: Proc SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName:Str);
32797: Proc HideWindowForSeconds(secs: Int); { //3 seconds }
32798: Proc HideWindowForSeconds2(secs: Int; apphandle, aself: TForm); { //3 seconds }
32799: Proc ConvertToGray(Cnv: TCanvas);
32800: Func GetFileDate(aFile:Str; aWithTime:Boolean):Str;
32801: Proc ShowMemory;
32802: Func ShowMemory2:Str;
32803: Func getHostIP:Str;
32804: Proc ShowBitmap(bmap: TBitmap);
32805: Func GetOsVersionInfo: TOSVersionInfo; { //thx to wischniewski }
32806: Func CreateDBGridForm(dblist: TStringList): TListBox;
32807: Func isService:Bool;
32808: Func isApplication:Bool;
32809: Func isTerminalSession:Bool;
32810: Func SetPrivilege(privilegeName:Str; enable:Bool):Bool;
32811: Proc getScriptandRunAsk;
32812: Proc getScriptandRun(ascript:Str);
32813: Func VersionCheckAct:Str;
32814: Proc getBox(aurl, extension:Str);
32815: Func CheckBox:Str;
32816: Func isNTFS:Bool;
32817: { //Proc doWebCamPic; }
32818: Proc doWebCamPic(picname:Str);
32819: Func readm:Str;
32820: Proc getGEOmapandRunAsk; { //APIKey needed }
32821: Func GetMapX(C_form,apath:Str; const Data:Str):Bool;
32822: Proc GetGEOmap(C_form,apath:Str; const Data:Str);
32823: Func GetMapXGeoReverse(C_form:Str; const lat, long:Str):Str;
32824: { //Func RoundTo(const AValue:Extended; const ADigit: TRoundToEXRangeExtended):Extended; }
32825: Func GetGeoCodeCoord(C_form:Str; const data:Str; atxt:Bool):Str;
32826: Func GetGeoCoord(C_form:Str; const data:Str; atxt:Bool):Str;
32827: ex.: writeln(GetGeoCoord('xml','church cefalu sicily',true))
32828: Func DownloadFile(SourceFile, DestFile:Str):Bool;
32829: Func DownloadFileOpen(SourceFile, DestFile:Str):Bool;
32830: Func OpenMap(const Data:Str):Bool;
32831: Func GetGeoCode(C_form,apath:Str; const data:Str; sfile:Bool):Str;
32832: Func getFileCount(amask:Str): Int;
32833: Func CoordinateStr(Idx: Int; PosInSec: Double; PosLn: TNavPos):Str;
32834: Proc DebugLn(DebugLOGFILE:Str; E:Str);
32835: Func IntToFloat(i: Int): double;
32836: Func AddThousandSeparator(S:Str; myChr: Char):Str;
32837: Func mymciSendString(cmd: PChar; ret: PChar; len: Int; callback: Int):Card;
32838: Func RoundTime(ADate:Str; Rounding: Int; bRound:Bool):Str;
32839: Func DynamicDllCallName(Dll:Str;const Name:Str;HasResult:Bool;var Returned:Card;const Parameters:array of Int):Bool;
32840: Func DynamicDllCall(Dll:Str;const Name:Str;HasResult:Bool;var Returned:Card;const Parameters:array of Int):Bool;
32841: Func GetSpecialFolderID(const Path:Str): Int;
32842: Func GetSpecialFolderPath2(FolderID: Int):Str;
32843: End C:\maXbook\maxbox3\mX3999\maxbox3\source\IFSI_WinForm1puzzle.pas File loaded
32844:
32845: { // News of 3.9.8 up }
32846: Halt-Stop Program in Menu,WebServer2,Stop Event Recompile,
32847: Conversion Routines, Prebuild Forms, more RCDData, DebugOutString
32848: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
32849: JvChart - TjvChart Component - 2009 Public
32850: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
32851: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
32852: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
32853: DMath DLL included incl. Demos
32854: Interface Navigator menu/View/Intf Navigator
32855: Unit Explorer menu/Debug/Units Explorer
32856: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maxXcel
32857: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
32858: Script History to 9 Files WebServer light /Options/Addons/WebServer
32859: Full Text Finder, JVSimLogic Simulator Package
32860: Halt-Stop Program in Menu, WebServer2, Stop Event ,
32861: Conversion Routines, Prebuild Forms, CodeSearch
32862: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
32863: Conversion Routines, Prebuild Forms, more RCDData, DebugOutString

```

```

32864: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
32865: JvChart - TjvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TjvPaintFX
32866: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
32867: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
32868: IDE Reflection API, Session Service Shell S3
32869: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
32870: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
32871: arduino map() function, PMRandom Generator
32872: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
32873: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
32874: REST Test Lib, Multilang Component, Forth Interpreter
32875: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
32876: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
32877: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
32878: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
32879: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
32880: QRCode Service, add more CFunctions like CDateTime of Synapse
32881: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
32882: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
32883: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
32884: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
32885: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
32886: BOLD Package, Indy Package5, maTRix. MATHEMAX
32887: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
32888: emax layers: system-package-component-unit-class-function-block
32889: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
32890: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
32891: Tutorial 18.3 RGB LED, OpenGL Geometry, maxpix, statictext
32892: OpenGL Game Demo: ..Options/Add Ons/Reversi
32893: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
32894: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
32895: 7% performance gain (hot spot profiling)
32896: PEP -Pascal Education Program, GSM Module, CGI, PHP Runner
32897: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
32898: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
32899: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
32900:
32901: add routines in 3.9.7.5
32902: 097: Proc RIRegister_BarCodeScanner_Routines(S: TPSExec);
32903: 996: Proc RIRegister_DBCtrls_Routines(S: TPSExec);
32904: 069: Proc RIRegister_IdStrings_Routines(S: TPSExec);
32905: 516: Proc RIRegister_JclMultimedia_Routines(S: TPSExec);
32906: 215: Proc RIRegister_PNGLoader_Routines(S: TPSExec);
32907: 374: Proc RIRegister_SerDlgs_Routines(S: TPSExec);
32908: 777: Proc RIRegister_LinarBitmap_Routines(S: TPSExec);
32909: 1216 Proc RIRegister_uPSI_KDialogs, TKBrowseFolderDialog;
32910:
32911: ////////////////////////////////// TestUnits //////////////////////////////////
32912: Proc SelftestPEM;
32913:   SelfTestCFundamentUtils;
32914:   SelfTestCFileUtils;
32915:   SelfTestCDateTime;
32916:   SelfTestCTimer;
32917:   SelfTestCRandom;
32918:   SelftestAES;
32919:   SelfTestASN1;
32920:   SelfTestX509;
32921:   TestDes:Bool;
32922:   Test3Des:Bool;
32923:   TestAes:Bool;
32924:   SelfTestcTLSEUtils;
32925:   SelfTestCFundamentUtils;
32926:   SelfTestcHTTPEUtils;
32927:   SelfTestcXMLFunctions;
32928:   SelfTestHugeWord;
32929:   SelfTestRSA;
32930:
32931:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
32932:   Assert(WinPathToUnixPath('c\d.f') = '/c/d.f', 'WinPathToUnixPath
32933: //Note: There's no need for installing a client certificate in the
32934: // webbrowser. The server asks the webbrowser to send a certificate but
32935: // if nothing is installed the software will work because the server
32936: // doesn't check to see if a client certificate was supplied. If you want you can install: file:
32937: c_cacert.p12 password: c_cakey
32938:
32939: TKObject = class(TObject)
32940: private
32941:   FParent: TKObjectList;
32942:   Proc SetParent(const Value: TKObjectList);
32943: protected
32944:   FUpdateLock: Int;
32945:   Proc CallBeforeUpdate; virtual;
32946:   Proc CallAfterUpdate; virtual;
32947:   Proc ParentChanged; virtual;
32948: public
32949:   constructor Create; virtual;
32950:   Proc Assign(ASource: TKObject); virtual;
32951:   Func EqualProperties(AValue: TKObject):Bool; virtual;
32952:   Proc LockUpdate; virtual;

```

```

32952:     Proc UnLockUpdate; virtual;
32953:     Func UpdateUnlocked:Bool; virtual;
32954:     property Parent: TKObjectList read FParent write SetParent;
32955: end;
32956:
32957: TKObjectClass = class of TKObject;
32958: TKObjectList = class(TKObjectList)
32959: protected
32960:     FUpdateLock: Int;
32961:     Proc CallBeforeUpdate; virtual;
32962:     Proc CallAfterUpdate; virtual;
32963: public
32964:     constructor Create; virtual;
32965:     Func Add(AObject: TObject): Int;
32966:     Proc Assign(ASource: TKObjectList); virtual;
32967:     Func EqualProperties(AValue: TKObjectList):Bool; virtual;
32968:     Proc Insert(Index: Int; AObject: TObject);
32969:     Proc LockUpdate; virtual;
32970:     Proc UnLockUpdate; virtual;
32971:     Func UpdateUnlocked:Bool; virtual;
32972: end;
32973:
32974: TGraphicControl = class(TControl)
32975: private
32976:     FCanvas: TCanvas;
32977:     Proc WMPaint(var Message: TWMPaint); message WM_PAINT;
32978: protected
32979:     Proc Paint; virtual;
32980:     property Canvas: TCanvas read FCanvas;
32981: public
32982:     constructor Create(AOwner: TComponent); override;
32983:     destructor Destroy; override;
32984: end;
32985:
32986: TCustomControl = class(TWinControl)
32987: private
32988:     FCanvas: TCanvas;
32989:     Proc WMPaint(var Message: TWMPaint); message WM_PAINT;
32990: protected
32991:     Proc Paint; virtual;
32992:     Proc PaintWindow(DC: HDC); override;
32993:     property Canvas: TCanvas read FCanvas;
32994: public
32995:     constructor Create(AOwner: TComponent); override;
32996:     destructor Destroy; override;
32997: end;
32998: RegisterPublishedProperties;
32999: ('ONCHANGE', 'TNotifyEvent', iptrw);
33000: ('ONCLICK', 'TNotifyEvent', iptrw);
33001: ('ONDBLCLICK', 'TNotifyEvent', iptrw);
33002: ('ONENTER', 'TNotifyEvent', iptrw);
33003: ('ONEXIT', 'TNotifyEvent', iptrw);
33004: ('ONKEYDOWN', 'TKeyEvent', iptrw);
33005: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
33006: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
33007: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
33008: ('ONMOUSEUP', 'TMouseEvent', iptrw);
33009: //*****
33010: // To stop the while loop, click on Options/Show Include (boolean switch)!
33011: Control a loop in a script with a form event:
33012: IncludeON; //control the while loop
33013: while maxForm1.ShowInclude1.checked do begin //menu event Options/Show Include
33014: repeat {for it:= 1 to n do} until is keypressed //keypress in output window below (memo2)
33015: include example:
33016: {$I .\web\mX47520\flcMaths.inc}
33017:
33018: //-----
33019: //*****mX4 ini-file Configuration*****
33020: //-----
33021: using config file maxboxdef.ini      menu/Help/Config File
33022:
33023: /*** Definitions for maXbox mX47 ***
33024: [FORM]
33025: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
33026: FONTSIZE=14
33027: EXTENSION=txt
33028: SCREENX=1386
33029: SCREENY=1077
33030: MEMHEIGHT=350
33031: PRINTFONT=Courier New //GUI Settings
33032: FONTNAME= Courier New //GUI Settings for editor
33033: LINENUMBERS=Y //line numbers at gutter in editor at left side
33034: EXCEPTIONLOG=Y //store excepts+success in 2 log files see below! -menu Debug/Show Last Exceptions
33035: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
33036: BOOTSCRIPT=Y //enabling load a boot script
33037: MEMORYREPORT=Y //shows memory report on closing maXbox
33038: MACRO=Y //expand macros (see below) incode e.g. #path:C:\maXbox\works2021\maxbox4\docs\
33039: NAVIGATOR=N //shows Func list at the right side of editor
33040: NAVWIDTH=350 //width of the right side interface list <CTRL L> >=200

```

```

33041: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
33042: DEP=Y // activate Data Execution Prevention
33043: INDENT=Y //shows indent vertical lines! (prepare for code folding) stop it with=N
33044: [WEB]
33045: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
33046: IPHOST=192.168.1.53 //run as Administrator!
33047: ROOTCERT=filepathY
33048: SCERT=filepathY
33049: RSAKEY=filepathY
33050: VERSIONCHECK=Y //set to =N for better starttime and less web traffic!
33051: APP=C:\WINDOWS\System32\calc.exe //set path to an external app
33052: MYSCRIPT=E:\maxbox4\mXGit39991\examples\330_myclock.txt //start script of menu /View/MyScript
33053:
33054: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
33055: Also possible to set report memory in script to override ini setting
33056: Proc Set_ReportMemoryLeaksOnShutdown(abo:Bool)
33057:
33058: After Change the ini file you can reload the file with ../Help/Config Update
33059: //-----
33060: //*****mX4 maildef.ini ini-file Configuration*****
33061: //-----
33062: /*** Definitions for maXMail ***/
33063: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
33064: [MAXMAIL]
33065: HOST=getmail.softwareschule.ch
33066: USER=mailusername
33067: PASS=password
33068: PORT=110
33069: SSL=Y
33070: BODY=Y
33071: LAST=5
33072:
33073: Command Line Interface CLI -f-s-m-r-t as ParamStr(2); compile with -c
33074: -----
33075: Ex.: S_ShellExecute(ExtractFilePath(ParamStr(0))+ 'maXbox4.exe',
33076: myscript + ParamStr(2),seCmdOpen)
33077: Ex.: C:\maXbox\mx4\mx4_v38\mx4\source>maxbox4_7.exe -c examples\440_xml_tutor2.txt
33078:
33079: if (ParamStr(2) = '-f') then begin
33080: Application.BringToFront; //maximize?
33081: //maxform1.Show;
33082: end;
33083: if (ParamStr(2) = '-s') then begin
33084: maxform1.Show;
33085: end;
33086: Compile1Click(self);
33087: maxform1.memo2.lines.add('CLI Console Call Log at: ' +DateTimeToStr(Now));
33088: hlog.Add('>>>> Start Console Call Exe: {App_name} v{App_ver}{800}{now}');
33089: if (ParamStr(2) = 'm') then begin
33090: //Compile1Click(self);!
33091: Application.Minimize;
33092: end;
33093: if (ParamStr(2) = 'r') then begin
33094: Application.run;
33095: end;
33096: if (ParamStr(2) = '-t') then begin //new4
33097: Application.terminate;
33098: end;
33099: -----
33100: S_ShellExecute(ExePath+'maXbox4.exe',
33101: ExePath+'examples\773_streamfibonacci_first.txt t',secmdopen); -terminate
33102: ADO Connection String:
33103: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
33104: \452_dbtreeview2access.txt \452_dbtrv3accessUML2.txt
33105: program TestDbTreeViewMainForm2_ACCESS;
33106: ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
33107: +Exepath+'\examples\detail.mdb;Persist Security Info=False';
33108: 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;"
33109:
33110: Func CreateAccessDatabase(FileName:Str):Str;
33111: var cat: OLEVariant;
33112: begin
33113: Result:= '';
33114: try
33115: cat:= CreateOleObject('ADOX.Catalog');
33116: cat.Create('Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' + FileName + ');
33117: cat:= NULL;
33118: except
33119: writeln(ExceptionToString(ExceptionType, ExceptionParam));
33120: writeln('ADOX.Catalog create failed ');
33121: //on e: Exception do Result := e.message;
33122: end;
33123: end;
33124:
33125: ADOCommand1.CommandText:= 'CREATE TABLE TABLE1 ('
33126: + 'ID AUTOINCREMENT,'
33127: + 'FirstName varchar(255) NOT NULL,'

```

```

33128:                                     +'LastName varchar(255),'
33129:                                     +'Age int )';
33130:
33131: OpenSSL Lib:  unit ssl_openssl_lib;
33132: {$IFDEF CIL}
33133: const
33134:   {$IFDEF LINUX}
33135:     DLLSSLName = 'libssl.so';
33136:     DLLUtilName = 'libcrypto.so';
33137:   {$ELSE}
33138:     DLLSSLName = 'ssleay32.dll';
33139:     DLLUtilName = 'libeay32.dll';
33140:   {$ENDIF}
33141: {$ELSE}
33142: var
33143:   {$IFDEF MSWINDOWS}
33144:     {$IFDEF DARWIN}
33145:       DLLSSLName:Str = 'libssl.dylib';
33146:       DLLUtilName:Str = 'libcrypto.dylib';
33147:     {$ELSE}
33148:       DLLSSLName:Str = 'libssl.so';
33149:       DLLUtilName:Str = 'libcrypto.so';
33150:     {$ENDIF}
33151:   {$ELSE}
33152:     DLLSSLName:Str = 'ssleay32.dll';
33153:     DLLSSLName2:Str = 'libssl32.dll';
33154:     DLLUtilName:Str = 'libeay32.dll';
33155:   {$ENDIF}
33156: {$ENDIF}
33157:
33158: Proc getHTTP_PNG(vimage: TImage; vURL:Str);
33159: var pngStream: TMemoryStream;
33160: begin
33161:   with TLinearBitmap.Create do
33162:     try
33163:       pngStream:= TMemoryStream.Create;
33164:       try
33165:         HTTPGet(vURL, pngStream)
33166:       except
33167:         showmessage(E.message)
33168:       end
33169:       pngStream.Position:= 0;
33170:       LoadFromStream2(pngstream, 'JPG');
33171:       vimage.Picture:= NIL;
33172:       AssignTo(vimage.picture);
33173:       //SaveToFile(ExePath+'mX4_open.png');
33174:     finally
33175:       Dispose;
33176:       Free;
33177:       pngStream.Free;
33178:     end;
33179: end;
33180:
33181: Proc ByteArray2Stream(ASource: TBytes; ADest: TStream);
33182: var i: Integer;
33183: begin
33184:   for i:= 0 to Length(ASource)-1 do
33185:     ADest.Write(chr(ASource[i]), 1);
33186:   end;
33187:
33188:   buffstr:Str;
33189:   ADest:= TMemoryStream.create;
33190:   ByteArray2Stream([0,23,3,234,8,45,255], aDest);
33191:   SetLength(buffstr,1);
33192:   ADest.Position:= 0;
33193:   for it:= 0 to adest.size-1 do begin
33194:     adest.read(buffstr, 1);
33195:     writeln('decode: '+ittoa(ord(buffstr[1])));
33196:   end;
33197:   aDest.free;
33198:
33199: Proc SearchByteSeq(AHexString:Str; adata: RawByteString);
33200: var dataToSearch: RawByteString;
33201: nextValueStr:Str;
33202: nextValueByte: integer; //byte;
33203: errorPos, index: integer;
33204: begin
33205:   repeat
33206:     AHexString:= Trim(AHexString);
33207:     nextValueStr:= copy(AHexString,1,2);
33208:     delete(AHexString,1,2);
33209:     val('$'+nextValueStr, nextValueByte, errorPos);
33210:     //Func ValBinary( const S :Str; var code : Int) : longint;
33211:     writeln(ittoa(errorpos))
33212:     if errorPos = 0 then
33213:       dataToSearch:= datatoSearch + chr(nextValueByte);
33214:     until AHexString = '';
33215:   if length(dataToSearch) > 0 then begin
33216:     index:= pos(dataToSearch, adata)-1;

```



```

33217:   if index = 0 then
33218:     writeln('Data not found')
33219:   else
33220:     writeln('Data found at position '+itoa(index));
33221:   end;
33222: end;
33223:
33224: Pictures from:
33225: http://www.softwareschule.ch/images/atomimage1.png
33226: http://www.softwareschule.ch/images/atomimage2.png
33227: http://www.softwareschule.ch/images/maxboxlogo.bmp
33228: http://www.softwareschule.ch/images/maxlindau2.png
33229: http://www.softwareschule.ch/images/citymax.bmp
33230: http://www.softwareschule.ch/images/oposlogo.gif
33231:
33232: //-----
33233: //*****mX4 Macro Tags *****
33234: //-----
33235: asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\docs\maxbox_extract_funclist399.txt end
33236:
33237: //Tag Macros in ini-file configure
33238:
33239:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
33240:
33241: //Tag Macros
33242: 10188: SearchAndCopy(memo1.lines, '#name', getUsernameWin, 11);
33243: 10189: SearchAndCopy(memo1.lines, '#date', datetimetToStr(now), 11);
33244: 10190: SearchAndCopy(memo1.lines, '#host', getComputenameWin, 11);
33245: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
33246: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
33247: 10199: SearchAndCopy(memo1.lines, '#files', fname + ' +SHA1(Act_Filename), 11);
33248: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
33249: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
33250: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
33251: [getUserNameWin, getComputenameWin, datetimetToStr(now),
33252: 10196: SearchAndCopy(memo1.lines, '#head', Format('%s: %s: %s %s ',
33253: [getUserNameWin, getComputenameWin, datetimetToStr(now), Act_Filename]), 11);
33254: [getUserNameWin, getComputenameWin, datetimetToStr(now), Act_Filename]), 11);
33255: 10198: SearchAndCopy(memo1.lines, '#tech', Format('perf: %s threads: %d %s %s',
33256: [perftime numprocessthreads, getIPAddress(getComputerNameWin),
33257: timetoStr(time), mbversion]), 11);
33258: 10298: SearchAndCopy(memo1.lines, '#net', Format('DNS: %s; local IPs: %s; local IP: %s',
33259: [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)], 10);
33260:
33261: // #tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
33262: // #tech!perf: 0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
33263:
33264: //Replace Macros
33265: SearchAndCopy(memo1.lines, 'timet14:40:27oStr(time), 6);
33266: SearchAndCopy(memo1.lines, 'atetoS17/02/2017tr(date), 6);
33267: SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\docs\
33268: SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maxbox3\
33269: SearchAndCopy(memo1.lines, 'maxbox_functions.txt
33270: SearchAndCopy(memo1.lines, 'C:\maXbox\maxbox3\maxbox3\maxbox3\Source
33271: ref: netcologne.dl.sourceforge.net/project/maxbox/maxbox3.zip
33272: SearchAndCopy(memo1.lines, '#tech!perf:0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
33273: [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
33274: // #tech!perf: 0:0:2.116 threads: 15 192.168.56.1 14:40:27 4.2.5.10
33275: SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
33276: SearchAndCopy(memo1.lines, 'maxbox_extract_funclist476.txt
33277:
33278: //-----
33279: //*****mX4 ToDo List Tags ../Help/ToDo List*****
33280: //-----
33281: while I < sl.Count do begin
33282: // if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9#]*:*)') then
33283: if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*)' then
33284: BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
33285: else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*)' then
33286: BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
33287: else if MatchesMask(sl[I], '*/? TODO (#?#)*:*)' then
33288: BreakupToDo(Filename, sl, I, 'TODO', False, True) //only priority info TODO
33289: else if MatchesMask(sl[I], '*/? DONE (#?#)*:*)' then
33290: BreakupToDo(Filename, sl, I, 'DONE', False, True) //only priority info DONE
33291: else if MatchesMask(sl[I], '*/?*TODO*:*)' then
33292: BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
33293: else if MatchesMask(sl[I], '*/?*DONE*:*)' then
33294: BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
33295: Inc(I);
33296: end;
33297:
33298: //-----
33299: //*****mX4 Public Tools API *****
33300: //-----
33301: file : unit uPSI_fMain.pas; [SOTAP] Open Tools API Catalog
33302: // Those functions concern the editor and preprocessor, all of the IDE
33303: Example: Call it with maxform1.InfolClick(self)
33304: Note: Call all Methods with maxForm1., e.g.:

```

```

33305:           maxForm1.ShellStyle1Click(self);
33306:
33307: Proc SIRegister_fMain(CL: TPSPascalCompiler);
33308: begin
33309:   Const('BYTECODE','String 'bytecode.txt'
33310:   Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
33311:   Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
33312:   Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
33313:   Const('PSINC','String PS Includes (*.inc)|*.INC
33314:   Const('DEFFILENAME','String 'firstdemo.txt
33315:   Const('DEFINIFILE','String 'maxboxdef.ini
33316:   Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
33317:   Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
33318:   Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
33319:   Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
33320:   Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
33321:   Const('ALLUNITLIST','String 'docs\maxbox3_9.xml;
33322:   Const('INCLUDEBOX','String 'pas_includebox.inc
33323:   Const('BOOTSCRIPT','String 'maxbootscript.txt
33324:   Const('MBVERSION','String '4.7.6.20
33325:   Const('VERSION','String'4.7.6.20
33326:   Const('MBVER','String '476
33327:   Const('MBVERI','Int'(476);
33328:   Const('MBVERIALL','Int'(47620);
33329:   Const('EXENAME','String 'maXbox4.exe
33330:   Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
33331:   Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
33332:   Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
33333:   Const('MXINTERNETCHECK','String 'www.ask.com
33334:   Const('MXMAIL','String 'max@kleiner.com
33335:   Const('TAB',Char #09);
33336:   Const('CODECOMPLETION','String 'bds_delphi.dci
33337:   SIRegister_TMaxForm1(CL);
33338: end;
33339:   with FindClass('TForm'),'TMaxForm1') do begin
33340:     memo2', 'TMemo', iptrw);
33341:     memo1', 'TSynMemo', iptrw);
33342:     CB1SCList', 'TComboBox', iptrw);
33343:     mxNavigator', 'TComboBox', iptrw);
33344:     IPHost', 'string', iptrw);
33345:     IPPort', 'Int', iptrw);
33346:     COMPort', 'Int', iptrw); //3.9.6.4
33347:     Splitter1', 'TSplitter', iptrw);
33348:     PSScript', 'TPSScript', iptrw);
33349:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
33350:     MainMenu1', 'TMainMenu', iptrw);
33351:     Program1', 'TMenuItem', iptrw);
33352:     Compile1', 'TMenuItem', iptrw);
33353:     Files1', 'TMenuItem', iptrw);
33354:     open1', 'TMenuItem', iptrw);
33355:     Save2', 'TMenuItem', iptrw);
33356:     Options1', 'TMenuItem', iptrw);
33357:     Savebefore1', 'TMenuItem', iptrw);
33358:     Largefont1', 'TMenuItem', iptrw);
33359:     sBytecode1', 'TMenuItem', iptrw);
33360:     Saveas3', 'TMenuItem', iptrw);
33361:     Clear1', 'TMenuItem', iptrw);
33362:     Slinenumbers1', 'TMenuItem', iptrw);
33363:     About1', 'TMenuItem', iptrw);
33364:     Search1', 'TMenuItem', iptrw);
33365:     SynPasSyn1', 'TSynPasSyn', iptrw);
33366:     memo1', 'TSynMemo', iptrw);
33367:     SynEditSearch1', 'TSynEditSearch', iptrw);
33368:     WordWrap1', 'TMenuItem', iptrw);
33369:     XPManifest1', 'TXPManifest', iptrw);
33370:     SearchNext1', 'TMenuItem', iptrw);
33371:     Replace1', 'TMenuItem', iptrw);
33372:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
33373:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
33374:     ShowInclude1', 'TMenuItem', iptrw);
33375:     SynEditPrint1', 'TSynEditPrint', iptrw);
33376:     Printout1', 'TMenuItem', iptrw);
33377:     mnPrintColors1', 'TMenuItem', iptrw);
33378:     dlgFilePrint', 'TPrintDialog', iptrw);
33379:     dlgPrintFont1', 'TFontDialog', iptrw);
33380:     mnuPrintFont1', 'TMenuItem', iptrw);
33381:     Include1', 'TMenuItem', iptrw);
33382:     CodeCompletionList1', 'TMenuItem', iptrw);
33383:     IncludeList1', 'TMenuItem', iptrw);
33384:     ImageList1', 'TImageList', iptrw);
33385:     ImageList2', 'TImageList', iptrw);
33386:     CoolBar1', 'TCoolBar', iptrw);
33387:     ToolBar1', 'TToolBar', iptrw);
33388:     tbtnLoad', 'TToolButton', iptrw);
33389:     ToolButton2', 'TToolButton', iptrw);
33390:     tbtnFind', 'TToolButton', iptrw);
33391:     tbtnCompile', 'TToolButton', iptrw);
33392:     tbtnTrans', 'TToolButton', iptrw);
33393:     tbtnUseCase', 'TToolButton', iptrw); //3.8

```

```

33394:    toolbtnTutorial', 'TToolButton', iptrw);
33395:    tbtn6res', 'TToolButton', iptrw);
33396:    ToolButton5', 'TToolButton', iptrw); 'ToolButton1', 'TToolButton', iptrw);
33397:    ToolButton3', 'TToolButton', iptrw); 'statusBar1', 'TStatusBar', iptrw);
33398:    SaveOutput1', 'TMenuItem', iptrw); 'ExportClipboard1', 'TMenuItem', iptrw);
33399:    Close1', 'TMenuItem', iptrw); 'Manual1', 'TMenuItem', iptrw);
33400:    About2', 'TMenuItem', iptrw); 'loadLastfile1', 'TMenuItem', iptrw);
33401:    imgLogo', 'TImage', iptrw); 'cedebg', 'TPSScriptDebugger', iptrw);
33402:    debugPopupMenu', 'TPopupMenu', iptrw);
33403:    BreakPointMenu', 'TMenuItem', iptrw);
33404:    Decompile1', 'TMenuItem', iptrw);
33405:    StepInto1', 'TMenuItem', iptrw);
33406:    StepOut1', 'TMenuItem', iptrw);
33407:    Reset1', 'TMenuItem', iptrw);
33408:    DebugRun1', 'TMenuItem', iptrw);
33409:    PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
33410:    PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
33411:    PSImport_Forms1', 'TPSImport_Forms', iptrw);
33412:    PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
33413:    tutorial4', 'TMenuItem', iptrw);
33414:    ExporttoClipboard1', 'TMenuItem', iptrw);
33415:    ImportfromClipboard1', 'TMenuItem', iptrw);
33416:    N4', 'TMenuItem', iptrw); N5', 'TMenuItem', iptrw); N6', 'TMenuItem', iptrw);
33417:    ImportfromClipboard2', 'TMenuItem', iptrw);
33418:    tutorial11', 'TMenuItem', iptrw);
33419:    N7', 'TMenuItem', iptrw);
33420:    ShowSpecChars1', 'TMenuItem', iptrw);
33421:    OpenDirectory1', 'TMenuItem', iptrw);
33422:    procMess', 'TMenuItem', iptrw);
33423:    tbtnUseCase', 'TToolButton', iptrw);
33424:    ToolButton7', 'TToolButton', iptrw);
33425:    EditFont1', 'TMenuItem', iptrw);
33426:    UseCase1', 'TMenuItem', iptrw);
33427:    tutorial21', 'TMenuItem', iptrw);
33428:    OpenUseCase1', 'TMenuItem', iptrw);
33429:    PSImport_DB1', 'TPSImport_DB', iptrw);
33430:    tutorial31', 'TMenuItem', iptrw);
33431:    SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
33432:    HTMLSyntax1', 'TMenuItem', iptrw);
33433:    ShowInterfaces1', 'TMenuItem', iptrw);
33434:    Tutorial5', 'TMenuItem', iptrw);
33435:    AllFunctionsList1', 'TMenuItem', iptrw);
33436:    ShowLastException1', 'TMenuItem', iptrw);
33437:    PlayMP31', 'TMenuItem', iptrw);
33438:    SynTeXSyn1', 'TSynTeXSyn', iptrw);
33439:    texSyntax1', 'TMenuItem', iptrw);
33440:    N8', 'TMenuItem', iptrw);
33441:    GetEMails1', 'TMenuItem', iptrw);
33442:    SynCppSyn1', 'TSynCppSyn', iptrw);
33443:    CSyntax1', 'TMenuItem', iptrw);
33444:    Tutorial6', 'TMenuItem', iptrw);
33445:    New1', 'TMenuItem', iptrw);
33446:    AllObjectsList1', 'TMenuItem', iptrw);
33447:    LoadBytecode1', 'TMenuItem', iptrw);
33448:    CipherFile1', 'TMenuItem', iptrw);
33449:    N9', 'TMenuItem', iptrw); 'N10', 'TMenuItem', iptrw);
33450:    Tutorial11', 'TMenuItem', iptrw);
33451:    Tutorial71', 'TMenuItem', iptrw);
33452:    UpdateService1', 'TMenuItem', iptrw);
33453:    PascalSchool1', 'TMenuItem', iptrw);
33454:    Tutorial81', 'TMenuItem', iptrw);
33455:    DelphiSite1', 'TMenuItem', iptrw);
33456:    Output1', 'TMenuItem', iptrw);
33457:    TerminalStyle1', 'TMenuItem', iptrw);
33458:    ReadOnly1', 'TMenuItem', iptrw);
33459:    ShellStyle1', 'TMenuItem', iptrw);
33460:    BigScreen1', 'TMenuItem', iptrw);
33461:    Tutorial91', 'TMenuItem', iptrw);
33462:    SaveOutput2', 'TMenuItem', iptrw);
33463:    N11', 'TMenuItem', iptrw);
33464:    SaveScreenshot', 'TMenuItem', iptrw);
33465:    Tutorial101', 'TMenuItem', iptrw);
33466:    SQLSyntax1', 'TMenuItem', iptrw);
33467:    SynSQLSyn1', 'TSynSQLSyn', iptrw);
33468:    Console1', 'TMenuItem', iptrw);
33469:    SynXMLSyn1', 'TSynXMLSyn', iptrw);
33470:    XMLSyntax1', 'TMenuItem', iptrw);
33471:    ComponentCount1', 'TMenuItem', iptrw);
33472:    NewInstancel', 'TMenuItem', iptrw);
33473:    toolbtnTutorial', 'TToolButton', iptrw);
33474:    Memory1', 'TMenuItem', iptrw);
33475:    SynJavaSyn1', 'TSynJavaSyn', iptrw);
33476:    JavaSyntax1', 'TMenuItem', iptrw);
33477:    SyntaxCheck1', 'TMenuItem', iptrw);
33478:    Tutorial10Statistics1', 'TMenuItem', iptrw);
33479:    ScriptExplorer1', 'TMenuItem', iptrw);
33480:    FormOutput1', 'TMenuItem', iptrw);
33481:    ArduinoDump1', 'TMenuItem', iptrw);
33482:    AndroidDump1', 'TMenuItem', iptrw);

```

```
33483: GotoEnd1', 'TMenuItem', iptrw);
33484: AllResourceList1', 'TMenuItem', iptrw);
33485: ToolButton4', 'TToolButton', iptrw);
33486: tbtn6res', 'TToolButton', iptrw);
33487: Tutorial11Forms1', 'TMenuItem', iptrw);
33488: Tutorial12SQL1', 'TMenuItem', iptrw);
33489: ResourceExplore1', 'TMenuItem', iptrw);
33490: Inf1', 'TMenuItem', iptrw);
33491: N12', 'TMenuItem', iptrw);
33492: CryptoBox1', 'TMenuItem', iptrw);
33493: Tutorial13Ciphering1', 'TMenuItem', iptrw);
33494: CipherFile2', 'TMenuItem', iptrw);
33495: N13', 'TMenuItem', iptrw);
33496: ModulesCount1', 'TMenuItem', iptrw);
33497: AddOns2', 'TMenuItem', iptrw);
33498: N4GewinntGame1', 'TMenuItem', iptrw);
33499: DocuforAddOns1', 'TMenuItem', iptrw);
33500: Tutorial14Async1', 'TMenuItem', iptrw);
33501: Lessons15Review1', 'TMenuItem', iptrw);
33502: SynPHPSyn1', 'TSynPHPSyn', iptrw);
33503: PHPSyntax1', 'TMenuItem', iptrw);
33504: Breakpoint1', 'TMenuItem', iptrw);
33505: SerialRS2321', 'TMenuItem', iptrw);
33506: N14', 'TMenuItem', iptrw);
33507: SynCSSyn1', 'TSynCSSyn', iptrw);
33508: CSyntax2', 'TMenuItem', iptrw);
33509: Calculator1', 'TMenuItem', iptrw);
33510: tbtnSerial', 'TToolButton', iptrw);
33511: ToolButton8', 'TToolButton', iptrw);
33512: Tutorial151', 'TMenuItem', iptrw);
33513: N15', 'TMenuItem', iptrw);
33514: N16', 'TMenuItem', iptrw);
33515: ControlBar1', 'TControlBar', iptrw);
33516: ToolBar2', 'TToolBar', iptrw);
33517: BtnOpen', 'TToolButton', iptrw);
33518: BtnSave', 'TToolButton', iptrw);
33519: BtnPrint', 'TToolButton', iptrw);
33520: BtnColors', 'TToolButton', iptrw);
33521: btnClassReport', 'TToolButton', iptrw);
33522: BtnRotateRight', 'TToolButton', iptrw);
33523: BtnFullSize', 'TToolButton', iptrw);
33524: BtnFitToWindowSize', 'TToolButton', iptrw);
33525: BtnZoomMinus', 'TToolButton', iptrw);
33526: BtnZoomPlus', 'TToolButton', iptrw);
33527: Panel1', 'TPanel', iptrw);
33528: LabelBrettgroesse', 'TLabel', iptrw);
33529: CB1SCList', 'TComboBox', iptrw);
33530: ImageListNormal', 'TImageList', iptrw);
33531: spbtnexplore', 'TSpeedButton', iptrw);
33532: spbtnexample', 'TSpeedButton', iptrw);
33533: spbsaveas', 'TSpeedButton', iptrw);
33534: imglogobox', 'TImage', iptrw);
33535: EnlargeFont1', 'TMenuItem', iptrw);
33536: EnlargeFont2', 'TMenuItem', iptrw);
33537: ShrinkFont1', 'TMenuItem', iptrw);
33538: ThreadDemol', 'TMenuItem', iptrw);
33539: HEXEditor1', 'TMenuItem', iptrw);
33540: HEXView1', 'TMenuItem', iptrw);
33541: HEXInspect1', 'TMenuItem', iptrw);
33542: SynExporterHTML1', 'TSynExporterHTML', iptrw);
33543: ExporttoHTML1', 'TMenuItem', iptrw);
33544: ClassCount1', 'TMenuItem', iptrw);
33545: HTMLOutput1', 'TMenuItem', iptrw);
33546: HEXEditor2', 'TMenuItem', iptrw);
33547: Minesweeper1', 'TMenuItem', iptrw);
33548: N17', 'TMenuItem', iptrw);
33549: PicturePuzzle1', 'TMenuItem', iptrw);
33550: sbvclhelp', 'TSpeedButton', iptrw);
33551: DependencyWalker1', 'TMenuItem', iptrw);
33552: WebScanner1', 'TMenuItem', iptrw);
33553: View1', 'TMenuItem', iptrw);
33554: mnToolbar1', 'TMenuItem', iptrw);
33555: mnStatusbar2', 'TMenuItem', iptrw);
33556: mnConsole2', 'TMenuItem', iptrw);
33557: mnCoolbar2', 'TMenuItem', iptrw);
33558: mnSplitter2', 'TMenuItem', iptrw);
33559: WebServer1', 'TMenuItem', iptrw);
33560: Tutorial17Server1', 'TMenuItem', iptrw);
33561: Tutorial18Arduinol', 'TMenuItem', iptrw);
33562: SynPerlSyn1', 'TSynPerlSyn', iptrw);
33563: PerlSyntax1', 'TMenuItem', iptrw);
33564: SynPythonSyn1', 'TSynPythonSyn', iptrw);
33565: PythonSyntax1', 'TMenuItem', iptrw);
33566: DMathLibrary1', 'TMenuItem', iptrw);
33567: IntfNavigator1', 'TMenuItem', iptrw);
33568: EnlargeFontConsole1', 'TMenuItem', iptrw);
33569: ShrinkFontConsole1', 'TMenuItem', iptrw);
33570: SetInterfaceList1', 'TMenuItem', iptrw);
33571: popintfList', 'TPopupMenu', iptrw);
```

```

33572:   intfAdd1', 'TMenuItem', iptrw);
33573:   intfDelete1', 'TMenuItem', iptrw);
33574:   intfRefactor1', 'TMenuItem', iptrw);
33575:   Defactor1', 'TMenuItem', iptrw);
33576:   Tutorial19COMArduinol', 'TMenuItem', iptrw);
33577:   Tutorial20Regex', 'TMenuItem', iptrw);
33578:   N18', 'TMenuItem', iptrw);
33579:   ManualE1', 'TMenuItem', iptrw);
33580:   FullTextFinder1', 'TMenuItem', iptrw);
33581:   Move1', 'TMenuItem', iptrw);
33582:   FractalDemol', 'TMenuItem', iptrw);
33583:   Tutorial21Android1', 'TMenuItem', iptrw);
33584:   Tutorial0Function1', 'TMenuItem', iptrw);
33585:   SimuLogBox1', 'TMenuItem', iptrw);
33586:   OpenExamples1', 'TMenuItem', iptrw);
33587:   SynJavaScriptSyn1', 'TSynJavaScriptSyn', iptrw);
33588:   JavaScriptSyntax1', 'TMenuItem', iptrw);
33589:   Halt1', 'TMenuItem', iptrw);
33590:   CodeSearch1', 'TMenuItem', iptrw);
33591:   SynRubySyn1', 'TSynRubySyn', iptrw);
33592:   RubySyntax1', 'TMenuItem', iptrw);
33593:   Undo1', 'TMenuItem', iptrw);
33594:   SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
33595:   LinuxShellScript1', 'TMenuItem', iptrw);
33596:   Rename1', 'TMenuItem', iptrw);
33597:   spdcodesearch', 'TSpeedButton', iptrw);
33598:   Preview1', 'TMenuItem', iptrw);
33599:   Tutorial22Services1', 'TMenuItem', iptrw);
33600:   Tutorial23RealTime1', 'TMenuItem', iptrw);
33601:   Configuration1', 'TMenuItem', iptrw);
33602:   MP3Player1', 'TMenuItem', iptrw);
33603:   DLLSpy1', 'TMenuItem', iptrw);
33604:   SynURIOpener1', 'TSynURIOpener', iptrw);
33605:   SynURISyn1', 'TSynURISyn', iptrw);
33606:   URILinksClicks1', 'TMenuItem', iptrw);
33607:   EditReplace1', 'TMenuItem', iptrw);
33608:   GotoLine1', 'TMenuItem', iptrw);
33609:   ActiveLineColor1', 'TMenuItem', iptrw);
33610:   ConfigFile1', 'TMenuItem', iptrw);
33611:   Sort1IntfList', 'TMenuItem', iptrw);
33612:   Redol', 'TMenuItem', iptrw);
33613:   Tutorial24CleanCode1', 'TMenuItem', iptrw);
33614:   Tutorial25Configuration1', 'TMenuItem', iptrw);
33615:   IndentSelection1', 'TMenuItem', iptrw);
33616:   UnindentSection1', 'TMenuItem', iptrw);
33617:   SkyStyle1', 'TMenuItem', iptrw);
33618:   N19', 'TMenuItem', iptrw);
33619:   CountWords1', 'TMenuItem', iptrw);
33620:   imbookmarkimages', 'TImageList', iptrw);
33621:   Bookmark11', 'TMenuItem', iptrw);
33622:   N20', 'TMenuItem', iptrw);
33623:   Bookmark21', 'TMenuItem', iptrw);
33624:   Bookmark31', 'TMenuItem', iptrw);
33625:   Bookmark41', 'TMenuItem', iptrw);
33626:   SynMultiSyn1', 'TSynMultiSyn', iptrw);
33627:
33628: Proc IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
33629: Proc IFPS3ClassesPlugin1ExecImport(Sender:TObject;Exec:TPSEExec;x:TPSRuntimeClassImporter);
33630: Proc PSScriptCompile( Sender : TPSScript)
33631: Proc Compile1Click( Sender : TObject)
33632: Proc PSScriptExecute( Sender : TPSScript)
33633: Proc open1Click( Sender : TObject)
33634: Proc Save2Click( Sender : TObject)
33635: Proc Savebefore1Click( Sender : TObject)
33636: Proc Largefont1Click( Sender : TObject)
33637: Proc FormActivate( Sender : TObject)
33638: Proc SBytecode1Click( Sender : TObject)
33639: Proc FormKeyPress( Sender : TObject; var Key : Char)
33640: Proc Saveas3Click( Sender : TObject)
33641: Proc Clear1Click( Sender : TObject)
33642: Proc Slinenumbers1Click( Sender : TObject)
33643: Proc About1Click( Sender : TObject)
33644: Proc Search1Click( Sender : TObject)
33645: Proc FormCreate( Sender : TObject)
33646: Proc Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:str;Line,Column:Int;
33647:                               var Action : TSynReplaceAction)
33648: Proc Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
33649: Proc WordWrap1Click( Sender : TObject)
33650: Proc SearchNext1Click( Sender : TObject)
33651: Proc Replace1Click( Sender : TObject)
33652: Func PSScriptNeedFile(Sdr:TObject;const OrginFileName:Str;var FName,Output:Str):Bool;
33653: Proc ShowInclude1Click( Sender : TObject)
33654: Proc Printout1Click( Sender : TObject)
33655: Proc mnuPrintFont1Click( Sender : TObject)
33656: Proc Include1Click( Sender : TObject)
33657: Proc FormDestroy( Sender : TObject)
33658: Proc FormClose( Sender : TObject; var Action : TCloseAction)
33659: Proc UpdateView1Click( Sender : TObject)
33660: Proc CodeCompletionList1Click( Sender : TObject)

```



```
33661: Proc SaveOutput1Click( Sender : TObject)
33662: Proc ExportClipboard1Click( Sender : TObject)
33663: Proc Close1Click( Sender : TObject)
33664: Proc Manual1Click( Sender : TObject)
33665: Proc LoadLastFile1Click( Sender : TObject)
33666: Proc MemolChange( Sender : TObject)
33667: Proc Decompile1Click( Sender : TObject)
33668: Proc StepInto1Click( Sender : TObject)
33669: Proc StepOut1Click( Sender : TObject)
33670: Proc Reset1Click( Sender : TObject)
33671: Proc cedebugAfterExecute( Sender : TPSScript)
33672: Proc cedebugBreakpoint( Sender:TObject;const FileName:str;Position,Row,Col:Card)
33673: Proc cedebugCompile( Sender : TPSScript)
33674: Proc cedebugExecute( Sender : TPSScript)
33675: Proc cedebugIdle( Sender : TObject)
33676: Proc cedebugLineInfo(Sender:TObject;const FileName:str;Position,Row,Col:Card)
33677: Proc MemolSpecialLineColors(Sender:TObject;Line:Int;var Speci:Bool;var FG,BG:TColor);
33678: Proc BreakPointMenuClick( Sender : TObject)
33679: Proc DebugRun1Click( Sender : TObject)
33680: Proc tutorial4Click( Sender : TObject)
33681: Proc ImportfromClipboard1Click( Sender : TObject)
33682: Proc ImportfromClipboard2Click( Sender : TObject)
33683: Proc tutorial1Click( Sender : TObject)
33684: Proc ShowSpecChars1Click( Sender : TObject)
33685: Proc StatusBar1Db1Click( Sender : TObject)
33686: Proc PSScriptLine( Sender : TObject)
33687: Proc OpenDirectory1Click( Sender : TObject)
33688: Proc procMessClick( Sender : TObject)
33689: Proc tbtnUseCaseClick( Sender : TObject)
33690: Proc EditFont1Click( Sender : TObject)
33691: Proc tutorial21Click( Sender : TObject)
33692: Proc tutorial31Click( Sender : TObject)
33693: Proc HTMLSyntax1Click( Sender : TObject)
33694: Proc ShowInterfaces1Click( Sender : TObject)
33695: Proc Tutorial5Click( Sender : TObject)
33696: Proc ShowLastException1Click( Sender : TObject)
33697: Proc PlayMP31Click( Sender : TObject)
33698: Proc AllFunctionsList1Click( Sender : TObject)
33699: Proc texSyntax1Click( Sender : TObject)
33700: Proc GetEMails1Click( Sender : TObject)
33701: Proc DelphiSite1Click(Sender: TObject);
33702: Proc TerminalStyle1Click(Sender: TObject);
33703: Proc ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
33704: Proc ShellStyle1Click(Sender: TObject);
33705: Proc Console1Click(Sender: TObject); //3.2
33706: Proc BigScreen1Click(Sender: TObject);
33707: Proc Tutorial91Click(Sender: TObject);
33708: Proc SaveScreenshotClick(Sender: TObject);
33709: Proc Tutorial101Click(Sender: TObject);
33710: Proc SQLSyntax1Click(Sender: TObject);
33711: Proc XMLSyntax1Click(Sender: TObject);
33712: Proc ComponentCount1Click(Sender: TObject);
33713: Proc NewInstance1Click(Sender: TObject);
33714: Proc CSyntax1Click(Sender: TObject);
33715: Proc Tutorial6Click(Sender: TObject);
33716: Proc New1Click(Sender: TObject);
33717: Proc AllObjectsList1Click(Sender: TObject);
33718: Proc LoadBytecode1Click(Sender: TObject);
33719: Proc CipherFile1Click(Sender: TObject); //V3.5
33720: Proc NewInstance1Click(Sender: TObject);
33721: Proc toolbtnTutorialClick(Sender: TObject);
33722: Proc Memory1Click(Sender: TObject);
33723: Proc JavaSyntax1Click(Sender: TObject);
33724: Proc SyntaxCheck1Click(Sender: TObject);
33725: Proc ScriptExplorer1Click(Sender: TObject);
33726: Proc FormOutput1Click(Sender: TObject); //V3.6
33727: Proc GotoEnd1Click(Sender: TObject);
33728: Proc AllResourceList1Click(Sender: TObject);
33729: Proc tbtn6resClick(Sender: TObject); //V3.7
33730: Proc InfolClick(Sender: TObject);
33731: Proc Tutorial10Statistics1Click(Sender: TObject);
33732: Proc Tutorial11Forms1Click(Sender: TObject);
33733: Proc Tutorial12SQL1Click(Sender: TObject); //V3.8
33734: Proc ResourceExplore1Click(Sender: TObject);
33735: Proc InfolClick(Sender: TObject);
33736: Proc CryptoBox1Click(Sender: TObject);
33737: Proc ModulesCount1Click(Sender: TObject);
33738: Proc N4GewinntGame1Click(Sender: TObject);
33739: Proc PHPSyntax1Click(Sender: TObject);
33740: Proc SerialRS2321Click(Sender: TObject);
33741: Proc CSyntax2Click(Sender: TObject);
33742: Proc Calculator1Click(Sender: TObject);
33743: Proc Tutorial13Ciphering1Click(Sender: TObject);
33744: Proc Tutorial14Async1Click(Sender: TObject);
33745: Proc PHPSyntax1Click(Sender: TObject);
33746: Proc BtnZoomPlusClick(Sender: TObject);
33747: Proc BtnZoomMinusClick(Sender: TObject);
33748: Proc btnClassReportClick(Sender: TObject);
33749: Proc ThreadDemo1Click(Sender: TObject);
```

```

33750: Proc HEXView1Click(Sender: TObject);
33751: Proc ExporttoHTML1Click(Sender: TObject);
33752: Proc Minesweeper1Click(Sender: TObject);
33753: Proc PicturePuzzle1Click(Sender: TObject); //V3.9
33754: Proc sbvclhelpClick(Sender: TObject);
33755: Proc DependencyWalker1Click(Sender: TObject);
33756: Proc CBISCLISTDrawItem(Control:TWinCtrl;Index:Int;aRect:TRect;State:TOwnerDrawState);
33757: Proc WebScanner1Click(Sender: TObject);
33758: Proc mnToolbar1Click(Sender: TObject);
33759: Proc mnStatusBar2Click(Sender: TObject);
33760: Proc mnConsole2Click(Sender: TObject);
33761: Proc mnCoolbar2Click(Sender: TObject);
33762: Proc mnSplitter2Click(Sender: TObject);
33763: Proc WebServer1Click(Sender: TObject);
33764: Proc PerlSyntax1Click(Sender: TObject);
33765: Proc PythonSyntax1Click(Sender: TObject);
33766: Proc DMathLibrary1Click(Sender: TObject);
33767: Proc IntfNavigator1Click(Sender: TObject);
33768: Proc FullTextFinder1Click(Sender: TObject);
33769: Func AppName:Str;
33770: Func ScriptName:Str;
33771: Func LastName:Str;
33772: Proc FractalDemolClick(Sender: TObject);
33773: Proc SimuLogBox1Click(Sender: TObject);
33774: Proc OpenExamples1Click(Sender: TObject);
33775: Proc Halt1Click(Sender: TObject);
33776: Proc Stop;
33777: Proc CodeSearch1Click(Sender: TObject);
33778: Proc RubySyntax1Click(Sender: TObject);
33779: Proc Undo1Click(Sender: TObject);
33780: Proc LinuxShellScript1Click(Sender: TObject);
33781: Proc WebScannerDirect(urls:Str);
33782: Proc WebScanner(urls:Str);
33783: Proc LoadInterfaceList2;
33784: Proc DLLSpy1Click(Sender: TObject);
33785: Proc Memo1Db1Click(Sender: TObject);
33786: Proc URILinksClicks1Click(Sender: TObject);
33787: Proc GotoLine1Click(Sender: TObject);
33788: Proc ConfigFile1Click(Sender: TObject);
33789: Proc Sort1IntfListClick( Sender : TObject)
33790: Proc RedolClick( Sender : TObject)
33791: Proc Tutorial24CleanCode1Click( Sender : TObject)
33792: Proc IndentSelection1Click( Sender : TObject)
33793: Proc UnindentSection1Click( Sender : TObject)
33794: Proc SkyStyle1Click( Sender : TObject)
33795: Proc CountWords1Click( Sender : TObject)
33796: Proc Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
33797: Proc Memo1GutterClick(Send:TObject;Button:TMouseButton;X,Y,Line:Int;Mark:TSynEditMark);
33798: Proc Bookmark11Click( Sender : TObject)
33799: Proc Bookmark21Click( Sender : TObject)
33800: Proc Bookmark31Click( Sender : TObject)
33801: Proc Bookmark41Click( Sender : TObject)
33802: Proc SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
33803: 'STATMemoryReport', 'boolean', iptrw);
33804: 'IPPort', 'Int', iptrw);
33805: 'COMPort', 'Int', iptrw);
33806: 'lbintfList', 'TListBox', iptrw);
33807: Func GetStatChange :Bool
33808: Proc SetStatChange( vstat :Bool)
33809: Func GetActFileName :Str
33810: Proc SetActFileName( vname :Str)
33811: Func GetLastFileName :Str
33812: Proc SetLastFileName( vname :Str)
33813: Proc WebScannerDirect( urls :Str)
33814: Proc LoadInterfaceList2
33815: Func GetStatExecuteShell :Bool
33816: Proc DoEditorExecuteCommand( EditorCommand : word)
33817: Func GetActiveLineColor: TColor
33818: Proc SetActiveLineColor(acolor: TColor)
33819: Proc ScriptListbox1Click(Sender: TObject);
33820: Proc Memo2KeyPress(Sender: TObject; var Key: Char);
33821: Proc EnlargeGutter1Click(Sender: TObject);
33822: Proc Tetris1Click(Sender: TObject);
33823: Proc ToDoList1Click(Sender: TObject);
33824: Proc ProcessList1Click(Sender: TObject);
33825: Proc MetricReport1Click(Sender: TObject);
33826: Proc ProcessList1Click(Sender: TObject);
33827: Proc TCPSockets1Click(Sender: TObject);
33828: Proc ConfigUpdate1Click(Sender: TObject);
33829: Proc ADOWorkbench1Click(Sender: TObject);
33830: Proc SocketServer1Click(Sender: TObject);
33831: Proc FormDemolClick(Sender: TObject);
33832: Proc Richedit1Click(Sender: TObject);
33833: Proc SimpleBrowser1Click(Sender: TObject);
33834: Proc DOSShell1Click(Sender: TObject);
33835: Proc SynExport1Click(Sender: TObject);
33836: Proc ExporttoRTF1Click(Sender: TObject);
33837: Proc FormCloseQuery(Sender: TObject; var CanClose:Bool);
33838: Proc SOAPTester1Click(Sender: TObject);

```

```

33839: Proc Sniffer1Click(Sender: TObject);
33840: Proc AutoDetectSyntax1Click(Sender: TObject);
33841: Proc FPlot1Click(Sender: TObject);
33842: Proc PasStyle1Click(Sender: TObject);
33843: Proc Tutorial183RGBLED1Click(Sender: TObject);
33844: Proc ReversilClick(Sender: TObject);
33845: Proc ManualmaxBox1Click(Sender: TObject);
33846: Proc BlaisePascalMagazine1Click(Sender: TObject);
33847: Proc AddToDo1Click(Sender: TObject);
33848: Proc CreateGUID1Click(Sender: TObject);
33849: Proc Tutorial27XML1Click(Sender: TObject);
33850: Proc CreateDLLStub1Click(Sender: TObject);
33851: Proc Tutorial28DLL1Click(Sender: TObject);
33852: Proc ResetKeyPressed;
33853: Proc SetKeyPressed;
33854: Proc KeyPressedFalse;
33855: Proc FileChanges1Click(Sender: TObject);
33856: Proc OpenGLTry1Click(Sender: TObject);
33857: Proc AllUnitList1Click(Sender: TObject);
33858: Proc Tutorial29UMLClick(Sender: TObject);
33859: Proc CreateHeader1Click(Sender: TObject);
33860: Proc Oscilloscope1Click(Sender: TObject);
33861: Proc Tutorial30WOT1Click(Sender: TObject);
33862: Proc GetWebScript1Click(Sender: TObject);
33863: Proc Checkers1Click(Sender: TObject);
33864: Proc TaskMgr1Click(Sender: TObject);
33865: Proc WebCam1Click(Sender: TObject);
33866: Proc Tutorial31Closure1Click(Sender: TObject);
33867: Proc GEOMapView1Click(Sender: TObject);
33868: Proc Run1Click(Sender: TObject);
33869: MaxForm1.GPSSatView1Click, 'GPSSatView1Click;
33870: MaxForm1.N3DLab1Click, 'N3DLab1Click;
33871: Proc ExternalApp1Click(Sender: TObject);
33872: Proc PANView1Click(Sender: TObject);
33873: Proc Tutorial39GEOMaps1Click(Sender: TObject);
33874: Proc UnitConverter1Click(Sender: TObject);
33875: maxform1.myscript1click(self)
33876: Proc Terminal1Click(Sender: TObject);
33877: Proc Tutorial361Click(Sender: TObject);
33878: Proc TrainingArduino1Click(Sender: TObject);
33879: Proc Chess41Click(Sender: TObject);
33880: Proc OrangeStyle1Click(Sender: TObject);
33881: Proc Darkcolor1Click(Sender: TObject);
33882: //-----
33883: //*****mX4 Editor SynEdit Tools API *****
33884: //-----
33885: Proc SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
33886: begin //with RegClassS(CL, 'TCustomControl', 'TCustomSynEdit') do
33887:   with FindClass('TCustomControl', 'TCustomSynEdit') do begin
33888:     Constructor Create( AOwner : TComponent)
33889:       SelStart, 'Int', iptrw);
33890:       SelEnd, 'Int', iptrw); AlwaysShowCaret, 'Boolean', iptrw);
33891:       Proc UpdateCaret
33892: Proc AddKey(Command:TSynEditorCommand;Key1:word;SS1:TShiftState;Key2:word;SS2:TShiftState);
33893: Proc AddKey(Command:TSynEditorCommand;Key1:word;SS1:TShiftState;Key2:word;SS2:TShiftState);
33894:   Proc BeginUndoBlock
33895:   Proc BeginUpdate
33896:   Func CaretInView :Bool
33897:   Func CharIndexToRowCol( Index : Int) : TBufferCoord
33898:   Proc Clear
33899:   Proc ClearAll
33900:   Proc ClearBookMark( BookMark : Int)
33901:   Proc ClearSelection
33902:   Proc CommandProcessor(Command:TSynEditorCommand; AChar : char; Data : pointer)
33903:   Proc ClearUndo
33904:   Proc CopyToClipboard
33905:   Proc CutToClipboard
33906:   Proc DoCopyToClipboard( const SText :Str)
33907:   Proc EndUndoBlock
33908:   Proc EndUpdate
33909:   Proc EnsureCursorPosVisible
33910:   Proc EnsureCursorPosVisibleEx( ForceToMiddle :Bool)
33911:   Proc FindMatchingBracket
33912:   Func GetMatchingBracket : TBufferCoord
33913:   Func GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
33914:   Proc ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
33915:   Func GetBookMark( BookMark : Int; var X, Y : Int) :Bool
33916:   Func GetHighlighterAttriAtRowCol(const XY: TBufferCoord; var Token:str; var Attri
33917:     : TSynHighlighterAttributes) :Bool
33918:   Func GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token :Str;
33919:     var TokenType, Start : Int; var Attri:TSynHighlighterAttributes):boolean
33920:   Func GetPositionOfMouse( out aPos : TBufferCoord) :Bool
33921:   Func GetWordAtRowCol( const XY : TBufferCoord) :Str
33922:   Proc GotoBookMark( BookMark : Int)
33923:   Proc GotoLineAndCenter( ALine : Int)
33924:   Func IdentChars : TSynIdentChars
33925:   Proc InvalidateGutter
33926:   Proc InvalidateGutterLine( aLine : Int)
33927:   Proc InvalidateGutterLines( FirstLine, LastLine : Int)

```

```

33928: Proc InvalidateLine( Line : Int)
33929: Proc InvalidateLines( FirstLine, LastLine : Int)
33930: Proc InvalidateSelection
33931: Func IsBookmark( BookMark : Int) :Bool
33932: Func IsPointInSelection( const Value : TBufferCoord) :Bool
33933: Proc LockUndo
33934: Func BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
33935: Func DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
33936: Func LineToRow( aLine : Int) : Int
33937: Func RowToLine( aRow : Int) : Int
33938: Func NextWordPos : TBufferCoord
33939: Func NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
33940: Proc PasteFromClipboard
33941: Func WordStart : TBufferCoord
33942: Func WordStartEx( const XY : TBufferCoord) : TBufferCoord
33943: Func WordEnd : TBufferCoord
33944: Func WordEndEx( const XY : TBufferCoord) : TBufferCoord
33945: Func PrevWordPos : TBufferCoord
33946: Func PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
33947: Func PixelsToRowColumn( aX, aY : Int) : TDisplayCoord
33948: Func PixelsToNearestRowColumn( aX, aY : Int) : TDisplayCoord
33949: Proc Redo
33950: Proc RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerDat:pointr);
33951: Func RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
33952: Func RowColToCharIndex( RowCol : TBufferCoord) : Int
33953: Func SearchReplace( const ASearch,AReplace:str; AOptions:TSynSearchOptions): int
33954: Proc SelectAll
33955: Proc SetBookMark( BookMark : Int; X : Int; Y : Int)
33956: Proc SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
33957: Proc SetDefaultKeystrokes
33958: Proc SetSelWord
33959: Proc SetWordBlock( Value : TBufferCoord)
33960: Proc Undo Proc UnlockUndo
33961: Proc UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
33962: Proc AddKeyUpHandler( aHandler : TKeyEvent)
33963: Proc RemoveKeyUpHandler( aHandler : TKeyEvent)
33964: Proc AddKeyDownHandler( aHandler : TKeyEvent)
33965: Proc RemoveKeyDownHandler( aHandler : TKeyEvent)
33966: Proc AddKeyPressHandler( aHandler : TKeyPressEvent)
33967: Proc RemoveKeyPressHandler( aHandler : TKeyPressEvent)
33968: Proc AddFocusControl( aControl : TWinControl)
33969: Proc RemoveFocusControl( aControl : TWinControl)
33970: Proc AddMouseDownHandler( aHandler : TMouseEvent)
33971: Proc RemoveMouseDownHandler( aHandler : TMouseEvent)
33972: Proc AddMouseUpHandler( aHandler : TMouseEvent)
33973: Proc RemoveMouseUpHandler( aHandler : TMouseEvent)
33974: Proc AddMouseCursorHandler( aHandler : TMouseEvent)
33975: Proc RemoveMouseCursorHandler( aHandler : TMouseEvent)
33976: Proc SetLinesPointer( ASynEdit : TCustomSynEdit)
33977: Proc RemoveLinesPointer
33978: Proc HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
33979: Proc UnHookTextBuffer
33980: BlockBegin', 'TBufferCoord', iptrw);
33981: BlockEnd', 'TBufferCoord', iptrw);
33982: CanPaste', 'Boolean', iptr); 'CanRedo', 'boolean', iptr);
33983: CanUndo', 'boolean', iptr); 'CaretX', 'Int', iptrw);
33984: CaretY', 'Int', iptrw); 'CaretXY', 'TBufferCoord', iptrw);
33985: ActiveLineColor', 'TColor', iptrw);
33986: DisplayX', 'Int', iptr); 'DisplayY', 'Int', iptr);
33987: DisplayXY', 'TDisplayCoord', iptr); 'DisplayLineCount', 'Int', iptr);
33988: CharsInWindow', 'Int', iptr);
33989: CharWidth', 'Int', iptr);
33990: Font', 'TFont', iptrw);
33991: GutterWidth', 'Int', iptr);
33992: Highlighter', 'TSynCustomHighlighter', iptrw);
33993: LeftChar', 'Int', iptrw);
33994: LineHeight', 'Int', iptr);
33995: LinesInWindow', 'Int', iptr);
33996: LineText', 'string', iptrw); Lines', 'TStrings', iptrw);
33997: Marks', 'TSynEditMarkList', iptr);
33998: MaxScrollWidth', 'Int', iptrw);
33999: Modified', 'Boolean', iptrw);
34000: PaintLock', 'Int', iptr);
34001: ReadOnly', 'Boolean', iptrw);
34002: SearchEngine', 'TSynEditSearchCustom', iptrw);
34003: SelAvail', 'Boolean', iptr); SelLength', 'Int', iptrw);
34004: SelTabBlock', 'Boolean', iptr);
34005: SelTabLine', 'Boolean', iptr); SelText', 'string', iptrw);
34006: StateFlags', 'TSynStateFlags', iptr);
34007: Text', 'string', iptrw); TopLine', 'Int', iptrw);
34008: WordAtCursor', 'string', iptr); 'WordAtMouse', 'string', iptr);
34009: UndoList', 'TSynEditUndoList', iptr);
34010: RedoList', 'TSynEditUndoList', iptr);
34011: OnProcessCommand', 'TProcessCommandEvent', iptrw);
34012: BookMarkOptions', 'TSynBookMarkOpt', iptrw);
34013: BorderStyle', 'TSynBorderStyle', iptrw);
34014: ExtraLineSpacing', 'Int', iptrw);
34015: Gutter', 'TSynGutter', iptrw);
34016: HideSelection', 'boolean', iptrw);

```

```

34017: InsertCaret', 'TSynEditCaretType', iptrw);
34018: InsertMode', 'boolean', iptrw); 'IsScrolling', 'Boolean', iptr);
34019: Keystrokes', 'TSynEditKeyStrokes', iptrw);
34020: MaxUndo', 'Int', iptrw); 'Options', 'TSynEditorOptions', iptrw);
34021: OverwriteCaret', 'TSynEditCaretType', iptrw);
34022: RightEdge', 'Int', iptrw); RightEdgeColor', 'TColor', iptrw);
34023: ScrollHintColor', 'TColor', iptrw);
34024: ScrollHintFormat', 'TScrollHintFormat', iptrw);
34025: ScrollBars', 'TScrollStyle', iptrw);
34026: SelectedColor', 'TSynSelectedColor', iptrw);
34027: SelectionMode', 'TSynSelectionMode', iptrw);
34028: ActiveSelectionMode', 'TSynSelectionMode', iptrw);
34029: TabWidth', 'Int', iptrw); WantReturns', 'boolean', iptrw);
34030: WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
34031: WordWrapGlyph', 'TSynGlyph', iptrw);
34032: OnChange', 'TNotifyEvent', iptrw);
34033: OnClearBookmark', 'TPlaceMarkEvent', iptrw);
34034: OnCommandProcessed', 'TProcessCommandEvent', iptrw);
34035: OnContextHelp', 'TContextHelpEvent', iptrw);
34036: OnDropFiles', 'TDropFilesEvent', iptrw);
34037: OnGutterClick', 'TGutterClickEvent', iptrw);
34038: OnGutterGetText', 'TGutterGetTextEvent', iptrw);
34039: OnGutterPaint', 'TGutterPaintEvent', iptrw);
34040: OnMouseCursor', 'TMouseCursorEvent', iptrw);
34041: OnPaint', 'TPaintEvent', iptrw);
34042: OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
34043: OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
34044: OnReplaceText', 'TReplaceTextEvent', iptrw);
34045: OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
34046: OnStatusChange', 'TStatusChangeEvent', iptrw);
34047: OnPaintTransient', 'TPaintTransient', iptrw);
34048: OnScroll', 'TScrollEvent', iptrw);
34049: end;
34050: Proc RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
34051: Func GetPlaceableHighlighters : TSynHighlighterList
34052: Func EditorCommandToDescrString( Cmd : TSynEditorCommand) :Str
34053: Func EditorCommandToCodeString( Cmd : TSynEditorCommand) :Str
34054: Proc GetEditorCommandValues( Proc : TGetStrProc)
34055: Proc GetEditorCommandExtended( Proc : TGetStrProc)
34056: Func IdentToEditorCommand( const Ident :Str; var Cmd : longint) :Bool
34057: Func EditorCommandToIdent( Cmd : longint; var Ident :Str) :Bool
34058: Func ConvertCodeStringToExtended( AString :Str) :Str
34059: Func ConvertExtendedToCodeString( AString :Str) :Str
34060: Func ConvertExtendedToCommand( AString :Str) : TSynEditorCommand
34061: Func ConvertCodeStringToCommand( AString :Str) : TSynEditorCommand
34062: Func IndexToEditorCommand( const AIndex : Int) : Int
34063: TSynEditorOption = (
34064: eoAltSetsColumnMode, //Holding down Alt Key will put select mode into columnar format
34065: eoAutoIndent, //Will indent caret onnewlines same amountof leading whitespace as
34066: // preceding line
34067: eoAutoSizeMaxScrollWidth, //Automatically resizes MaxScrollWidth property when insert text
34068: eoDisableScrollArrows, //Disables scroll bar arrow buttons when you cant scroll that
34069: //direction any more
34070: eoDragDropEditing, //Allows to select a textblock and drag in doc to another location
34071: eoDropFiles, //Allows the editor accept OLE file drops
34072: eoEnhanceHomeKey, //enhances home key positioning, similar to visual studio
34073: eoEnhanceEndKey, //enhances End key positioning, similar to JDeveloper
34074: eoGroupUndo, //When undoing/redoin actions,handle all cont.changes same kind in onecall
34075: //instead undoing/redoin each command separately
34076: eoHalfPageScroll, //By scrolling with page-up/page-down commands, scroll half page attime
34077: eoHideShowScrollbars, //if enabled, then scrollbars will only show if necessary.
34078: If you have ScrollPastEOL,then it horizontal bar will always there (uses MaxLength instead)
34079: eoKeepCaretX, //When moving through lines w/o cursor Past EOL, keeps X posof cursor
34080: eoNoCaret, //Makes it so the caret is never visible
34081: eoNoSelection, //Disables selecting text
34082: eoRightMouseMovesCursor, //When clicking right mouse for popup menu,moves cursor to location
34083: eoScrollByOneLess, //Forces scrolling to be one less
34084: eoScrollHintFollows, //The scroll hint follows the mouse when scrolling vertically
34085: eoScrollPastEof, //Allows the cursor to go past the end of file marker
34086: eoScrollPastEol, //Allows cursor go past lastchar into white space at end ofline
34087: eoShowScrollHint, //Shows a hint of the visible line numbers when scrolling vertically
34088: eoShowSpecialChars, //Shows the special Characters
34089: eoSmartTabDelete, //similar to Smart Tabs, but when you delete characters
34090: eoSmartTabs, //When tabbing, cursor will goto non-whitespace charof prev line
34091: eoSpecialLineDefaultFg, //disables foreground textcolor override OnSpecialLineColor event
34092: eoTabIndent, //If active <Tab>and<Shift><Tab> act block indent,unindent iftext select
34093: eoTabsToSpaces, //Converts a tab character to a specified number of space characters
34094: eoTrimTrailingSpaces //Spaces at the end of lines will be trimmed and not saved
34095:
34096: Change above Options in Editor (for bootscrip also):
34097: for eg. to indent blocks with tab:
34098:
34099: mem1.options:= mem1.options + [eoTabIndent]
34100: or to control something:
34101: mem1.wanttabs:= true;
34102: if (mem1.wanttabs) and (GetKeyState(VK_CONTROL)) then writeln('wanttabs0^');
34103:
34104: *****Important Editor Short Cut Cuts shortcut*****);
34105: Double click to select a word and count words with highlighting.

```



```

34106: Triple click to select a line.
34107: CTRL+SHIFT+click to extend a selection.
34108: Drag with the ALT key down to select columns of text !!!
34109: Drag and drop is supported.
34110: With CTRL 3 to jump to the last change (change tracker)
34111: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
34112: Type CTRL+A to select all.
34113: Type CTRL+N to set a new line.
34114: Type CTRL+T to delete a line or token. //Tokenizer
34115: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
34116: Type CTRL+Shift+S to Save as....
34117: Type CTRL+Alt+M to jump to Memo Terminal.
34118: Type CTRL+Shift+T to add ToDo in line and list.
34119: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
34120: Type CTRL[0..9] to jump or get to bookmarks.
34121: Type Home to position cursor at beginning of curr line and End to position at end of line.
34122: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of docu.
34123: Page Up and Page Down work as expected.
34124: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
34125: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
34126:
34127: {$ Short Key Positions Ctrl<A-Z>: }
34128: def
34129: <A> Select All
34130: <B> Count Words
34131: <C> Copy
34132: <D> Internet Start
34133: <E> Script List
34134: <F> Find
34135: <G> Goto
34136: <H> Mark Line
34137: <I> Interface List
34138: <J> Code Completion
34139: <K> Console
34140: <L> Interface List Box
34141: <M> Font Smaller -
34142: <N> New Line
34143: <O> Open File
34144: <P> Font Larger +
34145: <Q> Quit
34146: <R> Replace
34147: <S> Save!
34148: <T> Delete Line
34149: <U> Use Case Editor
34150: <V> Paste
34151: <W> URI Links
34152: <X> Reserved for coding use internal
34153: <Y> Delete Line
34154: <Z> Undo
34155:
34156: ref F1 Help to Func List
34157: F2 Syntax Check
34158: F3 Search Next
34159: F4 New Instance
34160: F5 Line Mark /Breakpoint
34161: F6 Goto End
34162: F7 Debug Step Into
34163: F8 Debug Step Out
34164: F9 Compile
34165: F10 Menu
34166: F11 Word Count Highlight
34167: F12 Reserved for coding use internal
34168:
34169: AddRegisteredVariable('Application', 'TApplication;
34170: AddRegisteredVariable('Screen', 'TScreen;
34171: AddRegisteredVariable('Self', 'TForm;
34172: AddRegisteredVariable('Memo1', 'TSynMemo;
34173: AddRegisteredVariable('memo2', 'TMemo;
34174: AddRegisteredVariable('maxForm1', 'TMaxform1; ///!
34175: AddRegisteredVariable('debugout', 'Tdebugoutput; ///!
34176: AddRegisteredVariable('hlog', 'THotlog; ///!
34177: AddRegisteredVariable('mouse', 'TMouse; ///!
34178: AddRegisteredVariable( it ,Int; //for closure and each loop!!
34179: AddRegisteredVariable( sr ,string; //for closure
34180: AddRegisteredVariable( bt ,boolean; //for closure
34181: AddRegisteredVariable( ft ,double; //for closure
34182: AddRegisteredVariable( srlist ,TStringlist; //for closures
34183:
34184: def ReservedWords: array[0..86] of string =
34185: ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
34186: 'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
34187: 'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
34188: 'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
34189: 'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
34190: 'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
34191: 'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
34192: 'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
34193: 'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
34194: 'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected', xraise

```

MAXBOX10 C:\maXbox\works2021\maxbox4\docs\maxbox_functions.txt
<http://www.softwareschule.ch/maxbox.htm>

```

34282: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
34283: - IDE menu /Help/Tools/ open the Task Manager
34284: - Editor: set doc section after end. end. in units and interface with begin main end!
34285: - new Style: menu//Output/Darkcolor
34286: http://www.softwareschule.ch/images/maXbox4\_darkcolor.png
34287: http://www.softwareschule.ch/images/maXbox4\_darkcolor3.png
34288:
34289: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
34290: - Add on when no browser is available start /Options/Add ons/Easy Browser
34291: - Add on SOAP Tester with SOP POST File
34292: - Add on IP Protocol Sniffer with List View
34293: - Add on OpenGL mX Robot Demo for android
34294: - Add on Checkers Game, Add on Oscilloscope /View GEO Map View3
34295:
34296: - Menu: Help/Tools as a Tool Section with DOS Opener
34297: - Menu Editor: export the code as RTF File
34298: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
34299: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
34300: - Context: Auto Detect of Syntax depending on file extension
34301: - Code: some Windows API Func start with w in the name like wGetAtomName();
34302: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
34303: - IDE File Check with menu ..View/File Changes/...
34304: - Context: Create a Header with Create Header in Navigator List at right window
34305: - Code: use SysErrorMessage to get a real Error Description, Ex.
34306: RemoveDir('c:\NoSuchFold;writeln('System Error Message:'+SysErrorMessage(GetLastError));
34307: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
34308: - Editor: with <Ctrl W> you can click on hyperlinks in Code - CTRL Click on link
34309: - Editor: with <Ctrl+Alt+R> you can write in RTF format with RichEdit link
34310: - Editor: with <Ctrl+Alt+T> you can fill your To Do List.
34311: - Editor: with <Ctrl+Alt+M> you jump to output window below (memo2).
34312: - Menu: Check Help/Tools! you can use richedit, DOS Shell or Explorer
34313: - Menu: Start View/MyScript of ini file maxboxdef.ini [MYSCRIPT]= path of script
34314: - Indent: close you doc lines with // for {...}
34315: - Menu: Help/ToDo List add prio and user with //1.....//
34316: Name#Prio# example:
34317: {/1 TODO (maxname#7#): do the new task one34 then mX4' //}
34318:
34319: - With this code I get an EIdHTTPProtocolException with the message "HTTP/1.1 403 Forbidden".
34320: Set the IdHTTPL1.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.
34321:
34322: with TIdHTTP.Create(self) do begin
34323: Request.UserAgent:= 'maXbox4 compatible';
34324: get('http://www.softwareschule.ch/examples/robo.txt')
34325: free
34326: end;
34327:
34328: use for TRegistry.create constructor TRegistry.Create1
34329: in maxbox_functions.txt add TRegistry.Create1
34330:
34331: //In the ServiceAfterUninstall method add these lines:
34332: // begin
34333: // Delete registry entries for event viewer.
34334: Key:= '\SYSTEM\CurrentControlSet\Services\Eventlog\Application\' + Self.Name;
34335: Reg:= TRegistry.Create1(KEY_READ or KEY_WRITE);
34336: try
34337: Reg.RootKey := HKEY_LOCAL_MACHINE;
34338: if Reg.KeyExists(Key) then
34339: Reg.DeleteKey(Key);
34340: finally
34341: Reg.Free;
34342: end;
34343:
34344: Func AnsiByteArrayStringLen2(Data: TBytes): Integer;
34345: Func StringToAnsiByteArray2(const S:Str): TBytes;
34346: Func AnsiByteArrayToString2(const Data: TBytes; Count: Integer):Str;
34347: Func BytesOf(const Value: Ansistr): TBytes;
34348: Func BytesOfChar(const Value: AnsiChar): TBytes;
34349: Func StringOf(const Bytes: array of Byte): Ansistr;
34350:
34351: - using DLL example in maXbox: //function: {*****}
34352: Func GetProcessMemoryInfo(Process:THandle;var
    MemoryCounters:TProcessMemoryCounters;cb:DWORD):BOOL;stdcall;
34353: External 'GetProcessMemoryInfo@psapi.dll stdcall';
34354: Func OpenProcess(dwDesiredAccess:DWORD;bInheritHandle:BOOL;dwProcessId: DWORD):THandle;
34355: External 'OpenProcess@kernel32.dll stdcall';
34356:
34357: Win API direct DLL with array of char result:
34358:
34359: //https://docs.microsoft.com/en-us/windows/win32/api/shlobj\_core/nf-shlobj\_core-shgetfolderpatha
34360:
34361: function SHGetFolderPathA(hwnd: HWND; csidl: integer; htoken: THANDLE;
34362: dwflags: DWORD; pszPath: string): HRESULT;
34363: external 'SHGetFolderPathA@Shell32.dll stdcall';
34364:
34365: function GetMyPictures: string;
34366: var pStr: string; //array[0..260] of Char;
34367: begin
34368: setLength(pStr, 260-1);

```

```

34369:   if SHGetFolderPathA(0, CSIDL_MYPICTURES, 0,0, pStr) = S_OK then
34370:     Result:= pStr;
34371: end;
34372:
34373: >>> GetMyPictures path C:\Users\Max\OneDrive\Pictures
34374:
34375: GCC Compile Ex Script
34376: Proc TFormMain_btnCompileClick(Sender: TObject);
34377: begin
34378:   AProcess:= TProcess.Create(nil);
34379:   try   AProcess.CommandLine := 'gcc.exe "' + OpenFileDialog1.FileName + '"
34380:     + ' -o "' + OpenFileDialog2.FileName + '"';
34381:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
34382:     AProcess.Execute;
34383:     Memo2.Lines.BeginUpdate;
34384:     Memo2.Lines.Clear;
34385:     Memo2.Lines.LoadFromStream(AProcess.Output);
34386:     Memo2.Lines.EndUpdate;
34387:   finally
34388:     AProcess.Free;
34389:   end;
34390: end;
34391:
34392: ref Stopwatch pattern snip
34393: Timel:= Time;
34394:   writeln(formatdatetime('start:" hh:mm:ss:zzz',Time))
34395:   if initAndStartBoard then
34396:     writeln('Filesize: '+inttostr(filesize(FILESAVE)));
34397:     writeln(formatdatetime('stop:" hh:mm:ss:zzz',Time))
34398:     PrintF('%d %s',[Trunc((Time-Timel)*24),FormatDateTime('h runtime:" nn:ss:zzz',Time-Timel)])
34399:     POST git-receive-pack (chunked)
34400: Pushing to https://github.com/maxkleiner/maxbox3.git
34401: To https://github.com/maxkleiner/maxbox3.git f127d21..c6a98da masterbox2 -> masterbox2
34402: updating local tracking ref 'refs/remotes/maxbox3Remote/masterbox2'
34403:
34404: -----
34405: History Shell Hell - Walk the Talk - CoolCode
34406: PCT Precompile Technology , mX4 ScriptStudio
34407: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
34408: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
34409: emax layers: system-package-component-unit-class-function-block
34410: new keywords def ref using maxCalcF UML xraise: use case act class state seq pac comp dep - lib lab
34411: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
34412: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
34413: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
34414: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
34415: 1 DLL Report, 24 Units add, DRTTable, Remote+, Cindy functions!
34416: DLL Report, UML Tutor, 32 Units add, DRTTable, Remote+, Cindy functions!
34417: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
34418: SendBuffer, Color+Caption hack,ComboSet,SetPrivilege,WakeOnLAN,ParaDice 3DCube Polygraph,OCR
34419: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
34420: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
34421: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
34422: Inno Install and Setup Routines Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
34423: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
34424: VFW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
34425: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
34426: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
34427: Func Menu/View/GEO Map View, DownloadFile, wgetX, sensors
34428: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtills
34429: ByteCode2, IPUtils2, GEOCode, CGI-Powtills, GPS_2, External App, Unit Converter
34430: Add 16 Units, 1 Slide,Tutor, Big Numbers (Decimals, TInt), ModBusTCP, TGEOInfo
34431: Add 36 Units, 1 Tutor, SOAPConn, AVI Res, OLEUtils, ACM, CDS, XMLDoc, DDE
34432: Add 77 Units, 12 Tutors, ChangeTracker, CPP+,OLEUtils2,xmldom,Chess4,3DFrame,XMLRPC,X509
34433: Add 15 Units, 1 Tutor, Pipe Libraray2, KLog, FPlot42, KDialogs, NumEdit, KControls, Kronos
34434: add 12 units and 125 functions - Class Helper - KMemo RTF
34435: add 16 units and 365 functions- WMI Script Type Library - webbox -restbox
34436: V4.6.2.10 Jan 2018:Tutor 56 Neural Network-Python Checker- 3 more Units PascalCoin
34437: V4.7.1.10 Sep 2019:Tutor 57-70, VState Machine,CGI,MachineLearning +20 Units +Tensorflow dll
34438: V4.7.1.82 Dec 2019:Tutor-72,EKON23Fixing,WebSocks/Spider,PHP_CGI,JS,OpenOffice,7+overbyte Units
34439: V4.7.4.64 June 2020: EKON24 Fixing, uPSI_SimpleRSS, Json Base Prometheus, neural CAI, Dendron
34440: V4.7.5.20 Jan 2021: few Fundamentals 5.00, JCL fixes, GraphMathLibrary, StringBuilder
34441: V4.7.5.80 July 2021: RSS+, WDCC, P4D_Beta (Python4Delphi), Wininet Threads<br>
34442: V4.7.5.90 October 2021: FLC Vectors, CAI Neuralnet, SingleListClass<br>
34443: V4.7.6.20 October 2022: Threadslst,CAI Neuralnet3,P4D+,EKON25 fixing,klib,HTTPClient2, -c CLI, PSProc<br>
34444: V4.7.6.20 Jan 2023: Threadslst, CAI Neuralnet3, P4D+, EKON25 fixing, klib, HTTPClient2, -c CLI, PSProc
34445: V4.7.6.50 May 2023: internals - TProcess2 dprocess - xmlstorage -AsphyreTimer -pacman core
34446:
34447: Ref:
34448: https://unibe-ch.academia.edu/MaxKleiner
34449: http://www.slideshare.net/maxkleiner1
34450: http://www.scribd.com/max_kleiner
34451: http://www.delphiforfun.org/Programs/Utilities/index.htm
34452: http://www.slideshare.net/maxkleiner1
34453: http://s3.amazonaws.com/PreviewLinks/22959.html
34454: http://www.softwareschule.ch/arduino_training.pdf
34455: http://www.jrsoftware.org/isinfo.php
34456: http://www.be-precision.com/products/precision-builder/express/
34457: http://www.blaise-pascal.eu/

```

```

34458: http://www.delphibasics.co.uk/
34459: http://www.youtube.com/watch?v=av89HAbqAsI
34460: http://www.angelfire.com/hi5/delphizeus/modal.html
34461: http://www.retroarchive.org/garbo/pc/turbopas/index.html
34462: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
34463: http://delphi.org/2014/01/every-android-api-for-delphi/
34464: https://en.wikipedia.org/wiki/User:Maxkleiner
34465: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
34466: https://bitbucket.org/max_kleiner/maxbox3
34467: https://bitbucket.org/max_kleiner/maxbox3/downloads
34468: https://bitbucket.org/max_kleiner/maxbox3/wiki/maXbox%20Tutorials
34469: http://www.slideshare.net/maxkleiner1/codereview-topics
34470: http://www.abaecker.biz/images/abakus/images/TAbCompass.gif
34471: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
34472: https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
34473: https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
34474: http://max.kleiner.com/boxart.htm
34475: https://maxbox4.wordpress.com
34476: https://archive.org/details/maxbox4
34477:
34478: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
34479: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
=renderBasicSearchNarrative&q=%s';
34480: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
%s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
34482:
34483: SAMPLES Snippets Blocks and Error or Errors or Bugs or Bug List or Trouble Bug Fix Errorlist bugfix codes
34484: ----- Troubleshooter
34485:
34486: [Error] C:\maXbox\TestApp2\sonysavefeb2014\TestApp\tetris\175_pas_anti_tetris2.txt(1:1): Variable Expected
34487: dont inc an longint in plus Inc(Score+5); ---> solution: Inc(Score);
34488:
34489: Editor: index out of bounds error exception in editor after open or modified
34490: 03/12/2019 20:10:50 V:4.7.1.20 [Max] MAXBOX10 List index out of bounds (147).
34491:
34492: This is due of Editor Indent ../Options/Show Indent
34493: 4 Workarounds:
34494: 1 simply move cursor to right or left and save (Ctrl+S)
34495: 2 Goto to end or line (147) or what it says and return new line or tab
34496: 3 Disable <Options Show Indent> in menu
34497: 4 Set INDENT=N in maxboxdef.ini for runtime systems
34498:
34499: can not see the Form , Form not visible ---> add Show;
34500: frm:= TmyForm.create(self)
34501: with frm do begin
34502:   Left := 245
34503:   Top := 108
34504:   BorderStyle := bsDialog
34505:   Caption := 'Suggestions'
34506:   ClientHeight := 219
34507:   ClientWidth := 297
34508:   Color := clBtnFace
34509:   //TextHeight := 20
34510:   Show; ----->
34511:
34512: Exception Handling Raise and XRaise
34513: type EPowerException = Exception;
34514: ...end else
34515:   Xraise ( EPowerException.Create('Infinite Result'));
34516: end;
34517: if (X > 0) then
34518:   try
34519:     result := exp(N * ln(X));
34520:   exit;
34521: except
34522:   raise EPowerException.Create('Overflow/Underflow Result');
34523: end;
34524:
34525: - Object with an Record inside get first a record instance and second you get the attribute
34526:
34527: Proc testprimesPerformance; // Performance Test
34528: var count, beg, j, r: integer;
34529: var f: boolean;
34530: begin
34531:   processmessagesOFF
34532:   Println(' prime time performance check ?????: ');
34533:   count:= 0;
34534:   beg:= Random(1000000000)+2;
34535:   for it:=beg to beg+5000 do begin
34536:     f:= True;
34537:     j:= 2;
34538:     r:= Round(Sqrt(it));
34539:     while f and (j<=r) do
34540:       if it mod j = 0 then f:= False
34541:       else inc(j);
34542:     if f then begin
34543:       Print(itoa(it)+' ');
34544:       count:= count+ 1;
34545:       if count mod 8 = 0 then

```



```

34546:     Println('');
34547: end;
34548: end;
34549: processmessagesON;
34550: end;
34551:
34552: RSS-FEED Rss Feed
34553: IIndyHTTP:=TIdHTTP.create(self);
34554: with TSimpleRSS.create(self) do begin
34555:     XMLType:= xtrDFrss;
34556:     IndyHTTP:= iindyhttp;
34557:     LoadFromHTTP(RSS_NewsFeed)
34558:     for it:= 0 to items.count-1 do begin
34559:         //Items[it].description
34560:         writeln(ittoa(it)+' : '+Items[it].title+' :'+Items[it].link));
34561:         println(GetSentimentStream7(URLSentimentAPI2,Items[it].title));
34562:     end;
34563:     free;
34564:     IIndyHTTP.Free;
34565: end; //}
34566:
34567: Ver: 4.7.4.64 (474). Workdir: C:\maXbox\mX47464\maxbox4
34568: [][] mX4 executed: 05/11/2020 17:57:05 Runtime: 0:0:39.224 Memload: 60% use
34569:
34570: Func OpenMap(const Data:Str):Bool;
34571: var encURL:Str;
34572: begin
34573:     encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
34574:     try //HttpGet(EncodedURL, mapStream); //WinInet
34575:     Result:= UrlDownloadToFile(nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,nil) = 0;
34576:     //OpenDoc(Exepath+'openmapx.html';
34577:     S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
34578:     finally encURL:= '';
34579:     end;
34580: end;
34581:
34582: Proc GetGEOMap(C_form,apath:Str; const Data:Str);
34583: var encodedURL:Str; mapStream: TMemoryStream;
34584: begin //encodedURL:= Format(UrlGoogleQrCode,[Width,Height,C_Level, HTTPEncode(Data)]);
34585:     encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
34586:     mapStream:= TMemoryStream.create;
34587:     try
34588:         Wininet_HttpGet(EncodedURL, mapStream); //WinInet
34589:         mapStream.Position:= 0;
34590:         mapStream.Savetofile(apath); // OpenDoc(apath);
34591:         S_ShellExecute(apath','',seCmdOpen);
34592:     finally
34593:         mapStream.Free;
34594:     end;
34595: end;
34596:
34597: Web Request API HTTPSend:
34598:
34599: //http://synapse.ararat.cz/doc/help/httpsend.THHTPSend.html#@Description
34600:
34601: Const aURL='https://api.metadefender.com/v4/file/bzIwMDYxNilTSW42NDBPVlprTWw3YjRBMQ';
34602:
34603: Begin //@main
34604:     srlist:= TStringlist.create;
34605:     with THHTPSend.create() do begin
34606:         Headers.Add('apikey: 6b337c92c792174a54acd715ab1aae64');
34607:         writeln(botostr(HTTPMethod('GET',aURL)));
34608:         with TJson.create() do begin
34609:             Parse(StreamToString3(Document));
34610:             Split(Stringify('{',srlist)
34611:         end;
34612:         writeln(ittoa(resultCode)+' :'+resultString+CRLF+srlist.text);
34613:         clear;free;
34614:     end;
34615:     srlist.Free;
34616:
34617:
34618: Proc BtnFactory(a,b,c,d:smallint; title,apic:str;
34619:                 var abtn:TBitBtn; anEvent:TNotifyEvent; afrm:TForm);
34620: begin
34621:     abtn:= TBitBtn.create(afrm);
34622:     with abtn do begin
34623:         parent:= afrm;
34624:         setBounds(a,b,c,d)
34625:         font.size:= 12;
34626:         glyph.LoadFromResourceName(HINSTANCE, apic);
34627:         mXButton(5,5,width, height,12,12,handle);
34628:         caption:= title;
34629:         onClick:= anEvent As TNotifyEvent;
34630:     end;
34631: end;
34632:
34633: Func GetSystemDirectory2(var S:Str):Bool;
34634: var Len: Int;

```

```

34635: begin
34636:   Len:= {Windows.}GetSystemDirectory('Nil', 0);
34637:   if Len > 0 then begin
34638:     SetLength(S, Len); //writeln(itoa(len))
34639:     Len:= {Windows.}GetSystemDirectory(PChar(S), Len);
34640:     SetLength(S, Len); //writeln(itoa(len))
34641:     Result:= Len > 0;
34642:   end else
34643:     Result:= False;
34644: end;
34645:
34646: Func RecordsetToXML2(const Recordset: variant):Str;
34647: var
34648:   RS: Variant;
34649:   Stream: TStringStream;
34650: begin
34651:   Result := '';
34652:   //if Recordset = unassigned then Exit;
34653:   Stream:= TStringStream.Create('');
34654:   try
34655:     RS:= CreateOleObject('ADODB.Recordset;
34656:     RS:= Recordset;
34657:     RS.Save(TStreamAdapter.Create(stream,soReference) as IUnknown, adPersistXML);
34658:     Stream.Position := 0;
34659:     Result:= Stream.DataString;
34660:   finally
34661:     Stream.Free;
34662:   end;
34663: end;
34664:
34665: convert JPEG to Bitmap
34666: if version = '4.7.6.20' then begin
34667:   JPGImage:= TJPEGImage.Create;
34668:   try
34669:     JPGImage.LoadFromStream(TResourceStream.Create(hInstance,'Everest','JPEG'));
34670:     with TImage.create(form1) do begin
34671:       setbounds(50,500,500,200)
34672:       parent:= form1;
34673:       Picture.Graphic:= JPGImage;
34674:       stretch:= true;
34675:     end;
34676:   finally
34677:     JPGImage.Free;
34678:   end; //}
34679: end;
34680:
34681: Func BigFactSum(n: integer):Str;
34682: var tbig1, tbig2, tbig3: TInteger;
34683: begin
34684:   result:= '0'
34685:   tbig1:= TInteger.create(1); //facultry
34686:   tbig2:= TInteger.create(0); //assign
34687:   tbig3:= TInteger.create(0);
34688:   for it:= 1 to n do begin
34689:     tbig2.assign1(it)
34690:     tbig1.mult(tbig2)
34691:     tbig3.add(tbig1);
34692:   end;
34693:   result:= tbig3.toString(false)
34694:   tbig3.free;
34695:   tbig2.free;
34696:   tbig1.free;
34697: end;
34698:
34699: No helper class possible or predict:
34700: USES Classes,StrUtils;
34701:
34702: TYPE
34703: TStringListHelper = CLASS HELPER FOR TStringList
34704: Func ToSQL :Str;
34705: END;
34706: Func TStringListHelper.ToSQL :Str;
34707: VAR S :Str;
34708:
34709: Func QuotedStr(CONST S :Str) :Str;
34710: BEGIN
34711: Result:='' + ReplaceStr(S, '''', ''''') + ''''
34712: END;
34713:
34714: BEGIN
34715: Result:='';
34716: FOR S IN Self DO BEGIN
34717: IF Result='' THEN Result:='(' ELSE Result:=Result+', '
34718: Result:=Result+QuotedStr(S)
34719: END;
34720: IF Result<>' THEN Result:=Result+') '
34721: END;
34722:
34723: SL:=TStringList.Create;

```

```

34724: SL.Add('One');
34725: SL.Add('Two');
34726: SL.Add('Number Three');
34727: SL.Add('It's number 4');
34728: WRITELN('SELECT * FROM TABLE WHERE FIELD IN '+SL.ToSQL);
34729: SELECT * FROM TABLE WHERE FIELD IN ('One','Two','Number Three','It's number 4')
34730:
34731: RSA Crypto Demo
34732: 965_RSA_IBZ_PrivatePublicKey_Form.pas
34733: const _n = '9516311845790656153499716760847001433441357'; /// p*q = modulus
34734: _e = '65537';
34735: _d = '5617843187844953170308463622230283376298685';
34736: abig:=TInteger.create(1);
34737: abig.Assignhex(ASCIItohex(message));
34738: writeln('big plain text integer: '+abig.ToString(normal))
34739: s_encrypt:= BigPowMod(abig.ToString(normal),_e,_n) //bigmulu(ap,aq));
34740: writeln('encrypted text integer '+s_encrypt)
34741: s_decrypt:= BigPowMod(s_encrypt,_d,_n) //bigmulu(ap,aq))
34742: writeln('decrypted text integer '+s_decrypt)
34743:
34744: Func ConvertHex(HexString :Str; Len : integer) :Str;
34745: begin
34746: if trim(HexString) = '' then Result := '0' else
34747: Result := IntToStr(StrToInt('$' + trim(HexString)));
34748: while length(Result) < Len do Result := '0' + Result;
34749: end;
34750: Proc SetBit(const a,b:Card);
34751: var d: Int64; // absolute dest;
34752: begin
34753: d:= $F00000000+(Int64(a) shl 21)+b shl 6;
34754: end;
34755:
34756: var alabel: TComponent; //Exception: A component named lblCell0 already exists.
34757: lc: integer;
34758: begin
34759: //object frMain: TfrMain
34760: if assigned(TLabel(FindComponent('lblCell'+intToStr(lc))) then begin
34761: write('exist&Nil ')
34762: TLabel(FindComponent('lblCell'+ IntToStr(lc)).Name:= '';
34763: alabel:= TLabel(FindComponent('lblCell'+ IntToStr(lc)));
34764: alabel:= Nil;
34765: end;
34766: end;
34767:
34768: //Discuss the Recursion of findfile!
34769: Proc FindFilePattern3(root:str; pattern:str);
34770: var SR: TFindRec;
34771: begin
34772: root:=IncludeTrailingPathDelimiter(root);
34773: if FindFirst3(root+'*.*',SR) = TRue then begin
34774: repeat
34775: Application.ProcessMessages;
34776: if ((sr.attributes and faDirectory)= SR.Attributes) and
34777: (pos('.',SR.name)=0) then
34778: FindFilePattern3(root+SR.Name,pattern)
34779: else begin
34780: if pos(pattern,SR.Name)>0 then
34781: //Form1.ListBox1.Items.Add(Root+SR.Name);
34782: writeln(Root+SR.Name);
34783: end;
34784: until FindNext3(SR) = False;
34785: FindClose3(SR);
34786: end;
34787: end;
34788:
34789: Encoding Decoding
34790: const tname= 'Subject: =?iso-8859-1?Q?Die_besten_W=FCnsche_zum_Neuen_Jahr_2021?=';
34791:
34792: writeln('mimecharset '+CharsetConversion(tn,ISO_8859_1, UTF_8));
34793: writeln('mimecharset '+CharsetConversion(tname,ISO_8859_1, win_1252));
34794: writeln( MimeCharsetToCharsetString(ISO_8859_1) );
34795:
34796: solution: writeln(DecodeHeader(tname)); or writeln(ForceDecodeHeader(tname));
34797: -----> Subject: Die besten Wünsche zum Neuen Jahr 2021
34798: writeln(UTF8toAnsi(ALUTF8HTMLdecode('Kaufbest&auml;tigung'))))
34799: -----> Kaufbestätigung
34800:
34801: RegEx by example:
34802: The Regexlibrary is inbuilt so it works independent of Operating System:
34803:
34804: Proc RegEX_ExtractDemo;
34805: var regEx: TRegExpr; //or OleVariant if HISUtils;
34806: InStr, ResStr:Str;
34807: begin
34808: ResStr:='';
34809: InStr:= 'Please e-mail us at support@mycompany.com or to sales@mycompany.com';
34810: // Create a regular expression
34811: regEx:= TRegExpr.create; //HISUtils.RegExpr;
34812: // Set regular expression pattern that specifies an e-mail address

```

```

34813:   regEx.Expression:='\w+(\.|\w-|+)*@([\\w-|+|.|]+[a-zA-Z]{2,6})';
34814:   // Execute search
34815:   if regEx.Exec(InStr) then Repeat
34816:     ResStr:= ResStr + regEx.Match[0] + ' ';
34817:   Until not regEx.ExecNext;
34818:
34819:   //Log.Message('These e-mails were extracted: '+ResStr);
34820:   writeln('These e-mails were extracted: '+ResStr);
34821:   // Posts the following to the log:
34822:   //'These e-mails were extracted: support@mycompany.com; sales@mycompany.com;'
34823: end;
34824:
34825: Proc LoadFilesByMask(lst: TStrings; const SpecDir, WildCard:Str);
34826: var SearchRec: TFindRec;
34827: begin lst.Clear;
34828:   if FindFirst3(SpecDir + WildCard, SearchRec) then
34829:     repeat
34830:       lst.Add(SpecDir + SearchRec.Name);
34831:     until FindNext3(SearchRec) = false;
34832:   FindClose3(SearchRec);
34833: end;
34834:
34835: Proc regExTest; //SynRegExpr unit RegEx Snippet
34836: var RegEx: TRegExpr; // This is a class, need to be freed!
34837: //Match: TMatch;
34838: i: Integer;
34839: begin
34840:   with TRegExpr.Create do begin
34841:     Expression:= ('^v(ol\.|olume)?s*([0-9]+)$');
34842:     if Exec('vol.3456') then
34843:       for i:=0 to SubExprMatchCount do
34844:         PrintF('Group %d : %s', [i, Match[i]]);
34845:       //Until not regEx.ExecNext;
34846:     Free;
34847:   end
34848: end;
34849:
34850: //(?) ---> case insensitive
34851: //(?) case-insensitive mode ON
34852: //(?-i) case-insensitive mode OFF
34853: writeln(getMatchString('(?)G[a-b].*', 'this GB string'));
34854: PrintF('CountPos: %d', [CountPos(uppercase('Selftest'),uppercase(sr))])
34855:
34856: Set Mapping
34857: There are some differences in the use of set operators in PascalScript and Delphi.
34858: In DelphiScript the [...] set operator is not supported or the in operator not always works.
34859: Set mapping with ord(CS) works:
34860: type TMimeCharset = (
34861:   CS_DEFAULT1,
34862:   CS_NOTMAPPED,
34863:   UTF_8,
34864:   WIN_1250,
34865:   WIN_1254,
34866:   WIN_1255,
34867:   WIN_1257,
34868:   WIN_1258,
34869:   ISO_8859_1,
34870:   ISO_8859_2)
34871:
34872: Cs: TMimeCharset;
34873: CS:= CS_NOTMAPPED;
34874: I := Ord(CS);
34875: CharsetInfos[I].MimeCharset := CS_NOTMAPPED;
34876: //CharsetInfos[I].CodePage := ERR_CP_NOTMAPPED;
34877: CharsetInfos[I].MimeName := ('');
34878: CharsetInfos[I].FriendlyName := 'sUnicodeUTF844';
34879:
34880: EnvironmentString:
34881: envlist:= TStringlist.create;
34882: StrtoList(GetEnvironmentString, envlist, CR)
34883: writeln(envlist.Text);
34884: writeln('envlist.count: '+itoa(envlist.count));
34885: envlist.Free;
34886:
34887: Note arrays in PascalScript cannot be initialized statically. For instance, the following code raises an
error:
34888: var I : array[0..1] of Integer = (3,4) // <-- Error!!!
34889:
34890: To solve problem, initialize array elements in your code. There are several ways to do this:
34891: Initializing each element individually:
34892: DelphiScript
34893: var a : array [0..3] of string;
34894: begin
34895:   a[0] := 'Element 0';
34896:   a[1] := 'Element 1';
34897:   a[2] := 'Element 2';
34898:   a[3] := 'Element 3';
34899:   ...
34900: end;

```

```

34901:
34902: Initializing elements in a loop:
34903: DelphiScript
34904: var a : array [0..3] of string;
34905: begin
34906:   for i := 0 to 3 do
34907:     a[i] := 'Element ' + IntToStr(i);
34908:   ...
34909: end;
34910:
34911: var array with 2 dimensions:
34912: Var frMain      : TfrMain;
34913:   iXPos         : array [1..3] of array [ 1..3] of Integer;
34914:   iOPos         : array [1..3] of array [ 1..3] of Integer;
34915:
34916: Constructor with override do most have a second one:
34917:
34918: var Images : TObjectList;
34919: //Create TObject list with AOwnsObjects set to True means that destroying
34920: //the object list will also destroy all of the objects it contains
34921: //NOTE: On ARC compiler destroying TObjectList will only remove the reference
34922: //to the objects and they will be destroyed only if thir reference count
34923: //drops to 0
34924:
34925: Images := TObjectList.Create;
34926: Images := TObjectList.Create1(true); //override
34927: al := TStatArray.Create1(0); // Growing array.
34928:
34929: Return Values
34930: Use ArcTan3 instead of ArcTan
34931: Func ArcTan3(const X : Extended) : Extended'; //bugfix of ArcTan()
34932:
34933: To return a value from a function, use Result instead of Func name. Using Func name will cause an error:
34934:
34935: Func Calc(x, y) : Integer;
34936: begin
34937:   Calc := (x - y) * (x + y); // Error !!!
34938:   Result := (x - y) * (x + y); // Correct variant
34939: end;
34940:
34941: Proc TStringsExchange(Index1, Index2: Integer; stlist: TStringlist);
34942: var
34943:   TempObject: TObject;
34944:   TempString: Str;
34945: begin
34946:   stlist.BeginUpdate;
34947:   try
34948:     TempString := stlist.Strings[Index1];
34949:     TempObject := stlist.Objects[Index1];
34950:     stlist.Strings[Index1] := stlist.Strings[Index2];
34951:     stlist.Objects[Index1] := stlist.Objects[Index2];
34952:     stlist.Strings[Index2] := TempString;
34953:     stlist.Objects[Index2] := TempObject;
34954:   finally
34955:     stlist.EndUpdate;
34956:   end;
34957: end; (//*)
34958:
34959: Integer Overflow of a DLL import:
34960: https://forum.lazarus.freepascal.org/index.php?topic=26116.0
34961: usually you set stdcall convention for a dll
34962: Func csoundgetversion: Longint;
34963:   external 'csoundGetVersion@csound64.dll stdcall';
34964:
34965: but for procedures or void try with cdecl:
34966: Proc csoundSleep(duration: word);
34967:   external 'csoundSleep@csound64.dll cdecl'; //no stdcall (int overflow)
34968:
34969: [Error]..\maXbox4\examples\558_hirestimer2test2.txt(12:1): program is not allowed at this position cause:
34970: strange typecasts like: TStream(nil) or TString(null)
34971:
34972: set a DLL Template: dll+Ctrl J
34973:
34974: //https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-internetsetoptiona
34975:
34976: Func InternetSetOption(hinternet:HINTERNET; dwoption: DWord;
34977:   buffer: Ansistr; bufflen: DWord):Bool;
34978:   external 'InternetSetOptionA@wininet.dll stdcall';
34979:
34980: { BOOLAPI InternetSetOptionA(
34981:   HINTERNET hInternet,
34982:   DWORD dwOption,
34983:   LPVOID lpBuffer,
34984:   DWORD dwBufferLength
34985: ); }
34986:
34987: Proc testReverse_Proxy;
34988: var ConnectionHandle: HINTERNET;
34989:   UserName: Ansistr;

```



```

34989:   res:Bool;
34990:   EmptyParam: OleVariant;
34991: begin
34992:   // Init ConnectionHandle here
34993:   ... res:= InternetSetOption(ConnectionHandle,
34994:                               INTERNET_OPTION_PROXY_USERNAME,
34995:                               UserName, Length(UserName) + 1);
34996:
34997: Proc Str() doesnt return a value : use intToStr()
34998:
34999: TBytes = array of byte : call of the library problem
35000: Proc SaveBytesToFile(const Data: TBytes; const FileName:Str); //const OK!
35001: Func ShiftByteArray(var A: TBytes): Byte; // var Not OK!
35002: in script
35003: Func ByteToString(var Value: TBytes):Str; // var and const OK!
35004: var I: integer;
35005:   S :Str;
35006:   Letra: char;
35007: begin
35008:   S := '';
35009:   for I := Length(Value)-1 Downto 0 do begin
35010:     letra := Chr(Value[I] + 48);
35011:     S := letra + S;
35012:   end;
35013:   Result := S;
35014: end;
35015:
35016: ----- Func wrapper -----
35017: Its better to call a Func wrapper than the Func itself:
35018:
35019: Proc QuickSortX(StringList : TStringList; L, R: Integer);
35020: Var
35021: I, J, P : Integer;
35022: Begin
35023:   P := StringList.Count;
35024:   If (L >= P) or (R >= P) Then Exit;
35025: {.....}
35026:
35027: Proc SortList(StringList : TStringList); //Func wrapper hides paras
35028: Begin
35029:   QuickSortX(StringList, 0, StringList.Count - 1)
35030: End;
35031: -----
35032: tst:= TStringlist.create;
35033: tst.add('7')
35034: tst.add('9')
35035: writeln(objtostr(tst))
35036: SortList(tst) //call to Func wrapper
35037:
35038: Hashmap or Dictionary or stringhash or map
35039:
35040: with TStringHashMap.create(CaseSensitiveTraits, 50) do begin
35041:   //Add( const s :Str; const p);
35042:   data['this is']:= '345'
35043:   data['this is2']:= '3456'
35044:   data['this is25']:= '3456677'
35045:   writeln(botostr(Has('this is2')))
35046:   setlength(sr,120)
35047:   stri:= pchar(data['this is'][10])
35048:   writeln(itoa(length(stri)))
35049:   writeln(stri)
35050:   writeln(itoa(Count));
35051:   writeln(itoa(HashSize))
35052:   clear;
35053:   free;
35054: end;
35055:
35056: Proc stringHashtest;
35057: var hash : TStringHash2;
35058: //Here's unit with those TIntegerHash2 and TStringHash2, TObjectHash2 described @
35059: basehash: THash2;
35060: begin
35061:   hash:= TStringHash2.Create;
35062:   try
35063:     hash['one']:= 'viens value';
35064:     hash['two']:= 'divi value';
35065:     //ShowMessage(hash['one']);
35066:     //ShowMessage(hash['two']);
35067:     if hash.exists('one') then
35068:       writeln(hash['one']);
35069:       writeln(hash['two']);
35070:       hash.Rename('one', 'onerename');
35071:       writeln(hash['onerename']);
35072:     finally
35073:       hash.Free;
35074:     end;
35075: end;
35076:
35077: // word frequency snippet

```

```

35078: for it:= 1 to wordlist.count-1 do begin
35079:   //writeln(itoa(it)+' '+Utf8ToAnsi(wordlist[it]));
35080:   aword:= lowercase(Utf8ToAnsi(wordlist[it]));
35081:   regex.RegEx:= '\w+';
35082:   regex.Subject:= aword;
35083:   if regex.match then aword:= regex.groups[0]; //word filter
35084:   if aword <> '' then
35085:     if dict.exists(aword) then
35086:       dict[aword]:= inttostr(strtoint(dict[aword])+1) else
35087:         dict[aword]:= '1';
35088:   end;
35089:   // iterate through all values
35090:   dict.Restart
35091:   for it:= 1 to (dict.itemcount-1) do
35092:     if dict.next then
35093:       writeln(dict.currentkey+': '+dict.items[dict.currentkey]) ; //}
35094:
35095: for each method - for in methood
35096: theres no for each operator but we can map it:
35097:
35098: //for c := 'a' to 'z' do
35099:   for c:= ord('a') to ord('z') do
35100:
35101: //for ch in adata do begin
35102:   for chi:= low(adata) to high(adata) do begin
35103:     ch:= adata[chi]
35104:
35105:     c: AnsiChar; Dec: Ansistr
35106:     //for c in Dec do
35107:     for it:= 1 to length(dec) do begin
35108:       c:= dec[it]
35109:
35110:     for in and set or include charset
35111:     var
35112:       include: charset; ex:str;
35113:       //if ex[i] in ['a'..'z'] then include:=include+[ex[i]]; -type mismatch
35114:     alternatives:
35115:     1. if (ex[i] > 'a') and (ex[i] < 'z') then //in ['a'..'z']
35116:     2. if ex[i] in StrToCharSet(lowercase(LETTERSET))
35117:       then include:= include+strtocharset(ex[i]);
35118:
35119: use strtcharset and chartostrset as helper Func with sets
35120: stest[2]:= '^';
35121: if stest[2] in
35122:   StrToCharSet(lowercase(LETTERSET)+' '+'?'+' '+'-'+'*+'+'/'+'ö'+'^') then
35123:   writeln('inside') else writeln('outside');
35124: >>> inside
35125: writeln(charsettostr(strtocharset('*+'?')))
35126:
35127: function IsSTRLegal2(c: string): boolean;
35128: begin
35129:   //result:= (Length(c)=1) and (c[1] in StrToCharSet('++' '-' '.'+ DIGISET));
35130:   result:= (Length(c)=1) and (c[1] in StrToCharSet('+-.,:+'DIGISET));
35131: end;
35132:
35133: Func IsInvalid(s:Str):Bool;
35134: var c: char;
35135: //letters: set of char;
35136: letters: CharSet;
35137: firstE:Bool;
35138: begin
35139:   //writeln(itoa(IndexStr(s,['k'])));
35140:   //Result := (Length(s) < 3) or (s.IndexOf('k') = -1) or (s.Length > 9);
35141:   Result:= (Length(s) < 3) or (pos('k',s) = 0) or (Length(s) > 9);
35142:   if not Result then begin
35143:     letters:= ['d', 'e', 'g', 'k', 'l', 'n', 'o', 'w'];
35144:     firstE:= true;
35145:     //for c in s do
35146:     for it:= 1 to length(s) do begin
35147:       c:= s[it];
35148:       if c in letters then
35149:         if (c = 'e') and (firstE) then
35150:           firstE:= false
35151:         else
35152:           //Exclude(letters, AnsiChar(c))
35153:           letters:= letters-strtochars(c)
35154:         else begin
35155:           result:= true
35156:           exit;
35157:         end;
35158:       end;
35159:     end;
35160:   end;
35161:
35162: Func TrimRight7(Str:Str):Str;
35163: var i: Integer;
35164: begin
35165:   i:= Length(Str);
35166:   //while (i > 0) and (Str[i] in [' ', #9]) do

```

```

35167:         while (i > 0) and (Str[i] in strtoccharset(' '+#9)) do
35168:             i:= i - 1;
35169:             Result:= Copy(Str,1,i);
35170:         end;
35171:
35172:         if MatchesMask('abc.xyz', 'a*.*',true) then
35173:             writeln('File name is OK.')
35174:         else
35175:             writeln('Invalid file name.');
```

Find all files of a defined date:

```

35177: 8 C:\>dir /s /o N *.dcu | find "16/06/2020" > C:\maXbox\maxboxunitalldisk_sort.txt
35179:
35180: Operator Overloading Surrogate:
35181: type complexreal = record
35182:     re : real;
35183:     im : real;
35184: end;
35185: { TFR }
35186: { frequency response }
35187: TFR = record
35188:     M, phi: extended; { magnitude and phase }
35189:     F: complexreal; { complex FR (F = M(omega) * exp(i * phi(omega))) }
35190: end;
35191:
35192: Func cexp (z : complexreal) : complexreal;
35193: { exponential : r := exp(z) }
35194: { exp(x + iy) = exp(x).exp(iy) = exp(x).[cos(y) + i sin(y)] }
35195: var expz : real;
35196: begin
35197:     expz := exp(z.re);
35198:     cexp.re := expz * cos(z.im);
35199:     cexp.im := expz * sin(z.im);
35200: end;
35201: Func complexmulu(z1 : complexreal; r : real): complexreal;
35202: { multiplication : z := z1 * r }
35203: begin
35204:     result.re := z1.re * r;
35205:     result.im := z1.im * r;
35206: end;
35207: FFr.F:= FFr.M * cexp(i * FFr.phi); { M and phi encoded in polar coordinates }
35208: //complextemp:= complexmulu(i,FFr.phi);
35209: FFr.F:= complexmulu(cexp(complexmulu(i,FFr.phi)), FFr.M); { M and phi encoded in polar }
35210:
35211: replace move with iteration to move()
35212:
35213: // big note to move the move to an iteration
35214: For i:= BListLenght downto 1 do BList[i]:= BList[i-1]; //or
35215:
35216:     move(Blist[0],Blist[1],BListLenght*SizeOf(Blist[0]));
35217:
35218: Array of const
35219: you cant set an array like :
35220: {CONST} VAR t : TA;
35221: (* =
35222:         (0,3,1,7,5,9,8,6,4,2),
35223:         (7,0,9,2,1,5,4,8,6,3),
35224:         (4,2,0,6,8,7,1,3,5,9),
35225:         (1,7,5,0,9,8,3,4,2,6),
35226:         (6,1,2,3,0,4,5,9,7,8),
35227:         (3,6,7,4,2,0,9,5,8,1),
35228:         (5,8,6,9,7,2,0,1,3,4),
35229:         (8,9,4,5,3,6,2,0,1,7),
35230:         (9,4,3,8,6,1,7,2,0,5),
35231:         (2,5,8,1,4,3,6,7,9,0)); *)
35232:
35233: but you can convert:
35234:
35235: const arrstr= '0,3,1,7,5,9,8,6,4,2'+
35236:               '7,0,9,2,1,5,4,8,6,3'+
35237:               '4,2,0,6,8,7,1,3,5,9'+
35238:               '1,7,5,0,9,8,3,4,2,6'+
35239:               '6,1,2,3,0,4,5,9,7,8'+
35240:               '3,6,7,4,2,0,9,5,8,1'+
35241:               '5,8,6,9,7,2,0,1,3,4'+
35242:               '8,9,4,5,3,6,2,0,1,7'+
35243:               '9,4,3,8,6,1,7,2,0,5'+
35244:               '2,5,8,1,4,3,6,7,9,0';
35245:
35246: procedure InitTable4;
35247: var i,j,k: integer; repl: string;
35248: begin
35249:     repl:= StringReplace(arrstr,',','',[rfReplaceAll]);
35250:     k:= 0
35251:     for i:= 0 to 9 do
35252:         for j:= 0 to 9 do begin
35253:             inc(k)
35254:             t[i][j]:= strtoint(repl[k])
35255:         end;
```

```

35256: end;
35257:
35258: Convert Static Array to Dynamic Array
35259:
35260: procedure TForm1Button1Click(Sender: TObject);
35261: var
35262:   ADate: TDateTime;
35263:   days: array[1..7] of string;
35264: begin
35265:   days[1] := 'Sunday';
35266:   days[2] := 'Monday';
35267:   days[3] := 'Tuesday';
35268:   days[4] := 'Wednesday';
35269:   days[5] := 'Thursday';
35270:   days[6] := 'Friday';
35271:   days[7] := 'Saturday';
35272:   ADate := StrToDate('31/01/1981');
35273:   ShowMessage('31/01/1981' + ' was a ' + days[{SysUtils.DayOfWeek(ADate)}]);
35274: end;
35275:
35276: //Incompatible types: 'Dynamic array' and 'array of string' to open array
35277:
35278: procedure TForm1Button1ClickDynArray(Sender: TObject);
35279: var aDate: TDateTime;
35280:   days: array of string;
35281: begin
35282:   setLength(days, 7);
35283:   days := ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];
35284:   aDate := StrToDate('31/01/1981');
35285:   ShowMessage('31/01/1981' + ' was a ' + days[{SysUtils.DayOfWeek(aDate)-1}]);
35286: end;
35287: // with SplitStr
35288: begin
35289:   aDate := StrToDate('31/01/1981');
35290:   days := SplitStr('Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday', ',');
35291:   writeln('31/01/1981' + ' was a ' + days[{SysUtils.DayOfWeek(aDate)-1}]);
35292: end;
35293:
35294: AssignFile versus TStringlist
35295: -----
35296: 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.
35297: 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
35298:
35299: filelist:= TStringlist.create;
35300: // assignfile(f,opendialog1.filename);
35301: filelist.loadfromfile( filename)
35302: // reset(f);
35303: // while not eof(f) do
35304: for it:= 0 to filelist.count-1 do begin
35305:   //readln(f,line);
35306:   line:= filelist.strings[it];
35307:   writeln(line)
35308:   if (length(line)>2) and (copy(line,1,2)='ID') then {skip Id records}
35309:   else begin {it's a data record}
35310:     if totcount=0 then begin
35311:       savedecimalseparator:=decimalseparator;
35312:       if pos(',',line)>0 then setdecimalseparator(',');
35313:       else setdecimalseparator('.');
35314:     end;
35315:     errcode:=0;
35316:     try
35317:       x:=strttoFloat(trim(line));
35318:     except errcode:=1;
35319:     end;
35320:     if (errcode=0) and (x>0) and (x<=amaxtime)
35321:     then begin
35322:       inc(samps[round(maxindex*x)]); {increment the count for the bar for x}
35323:       inc(totcount);
35324:     end;
35325:   end;
35326: end;
35327: // closefile(f);
35328: filelist.Free;
35329:
35330: Random Generator:
35331: https://github.com/TurboPack/SysTools/blob/master/source/StRandom.pas
35332:
35333: strand:= TStRandomSystem.create(42); //if 0 then randomseed, else randomstate
35334: sleep(200)
35335: writeln(floattostr(strand.asFloat))
35336: writeln(itoa(strand.asInt(10000)))
35337: writeln(itoa(strand.asInt($E15)))
35338: UniformityTest2(strand, ChiSquare, DegsFreedom)
35339: writeln('ChiSquare: '+floattostr(ChiSquare)+' ,degsfreedom: '+itoa(degsfreedom));
35340: UniformityTest3(strand, ChiSquare, DegsFreedom)
35341: writeln('ChiSquare: '+floattostr(ChiSquare)+' ,degsfreedom: '+itoa(degsfreedom));

```

```

35342: strand.Free;
35343:
35344: Https Post API: http://www.softwareschule.ch/download/maxbox_starter94.pdf
35345: translator for different countries:
35346: var httpReq,hr: Olevariant;
35347: try
35348:   hr:= httpReq.Open('POST','https://libretranslate.pussthecat.org/translate', false);
35349:   httpReq.setRequestHeader('user-agent',
35350: 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0');
35351:   httpReq.setRequestHeader('content-type','application/x-www-form-urlencoded');
35352:   //httpReq.setRequestHeader('X-RapidAPI-Host','nlp-translation.p.rapidapi.com');
35353:   if hr= S_OK then HttpReq.Send('q='+HTTPEncode(feedstream)+
35354:     '&source=es'+&target=en');
35355:   If HttpReq.Status = 200 Then
35356:     result:= HttpReq.responseText
35357:   Else result:= 'Failed at getting response: '+itoa(HttpReq.Status)+HttpReq.responseText;
35358:     //writeln('debug response '+HttpReq.GetAllResponseHeaders);
35359:   finally
35360:     httpreq:= unassigned;
35361:
35362: File Size Reference: filegetsize()
35363: Func FileGetSizeX(const FileName:Str): Int64;
35364: {$ifndef fpc}
35365: var
35366:   FileInfo: TSearchRec;
35367: {$endif}
35368: begin
35369:   {$ifdef fpc}
35370:     Result:= FileUtil.FileSize(FileName);
35371:   {$else}
35372:     // from LCL FileUtil code
35373:     FileInfo.Name:= Filename;
35374:     FileInfo.FindHandle:= (Windows.)FindFirstFile((Windows.){LPTSTR}(FileInfo.Name)
35375:       ,FileInfo.FindData);
35376:     if FileInfo.FindHandle={Windows.)Invalid_Handle_value then begin
35377:       Result:=-1;
35378:       Exit;
35379:     end;
35380:     Result:= (int64(FileInfo.FindData.nFileSizeHigh)
35381:       shl 32)+ FileInfo.FindData.nFileSizeLow;
35382:     (Windows.)FindCloseW(FileInfo.FindHandle);
35383:   {$endif}
35384: end;
35385: { this func is a wrapper for platform-specific file fns
35386:   IE a way to get rid of those portability warnings
35387:   and a place to put the equivalent linux fns }
35388:
35389: RTL OSLibrary+
35390: Func MySoundcard: Longint; external 'waveOutGetNumDevs@winmm.dll stdcall';
35391: Func isSound:Bool; begin result:= mySoundcard > 0 end;
35392: Func StringtoHex(Data:Str):Str;
35393: Func GetAnsistrRefCount(const S:Str):Card;
35394: Func WideStringToString(const ws: WideString; codePage: Word): Ansistr;
35395: Func StringToWideString(const s: Ansistr; codePage: Word): WideString;
35396: Proc FreeObjectList(List: TObjectList);
35397: Func SecondToTime(const Seconds:Card): Double;
35398: Func CopyDir2(const fromDir, toDir:Str):Bool;;
35399: Func MoveDir(const fromDir, toDir:Str):Bool;;
35400: Func DelDir(dir:Str):Bool;;
35401: Proc DeleteScansRect(Src, Dest: TBitmap; rs, rd: TRect);
35402: Proc FadeIn(ImageFileName: TFileName; aForm1: TForm);
35403: Proc FadeOut(ImageFileName: TFileName);
35404: Proc FadeOut32(const Bmp: TImage; Pause: Int);
35405: Func CheckBDEInstalled:Bool; //IsBDE
35406: Proc SaveStringToStream( const Str :Str; Stream : TStream);
35407: Func LoadStringFromStream( Stream : TStream) :Str;
35408: Func WaitForSyncObject(SyncObject:THandle;Timeout:Card;BlockInput:Bool):Card;
35409: Func ProcessMessage :Bool;
35410: Proc ProcessMessages( Timeout : DWORD); //without application!
35411: Func GetNumberOfEventLogRecords(hEventLog:THandle;var NumberOfRecords:DWORD): BOOL;
35412: Func GetOldestEventLogRecord(hEventLog:THandle; var OldestRecord : DWORD): BOOL;
35413: ex.:EventLog:=RegisterEventSource('0','maxbox4.exe');if GetNumberOfEventLogRecords(eventlog,recs)
35414: Proc InitializeLocaleSupport); Proc TerminateLocaleSupport;
35415: Func StrReplaceRegEx(const Subject,Pattern:Ansistr; Args:array of const : Ansistr;
35416: Func TickCountToDateTime( Ticks :Card ) : TDateTime;
35417: Proc OutputDebugStringmax( const S :Str);
35418: Proc OutputDebugFormat( const FmtStr :Str; Args : array of const);
35419: Func IsAppRunningInDelphi :Bool;
35420: Func RunExecutable( const AFileName :Str; AWaitForIt :Bool ) : DWORD;
35421: Func SystemCodePage : Int;
35422: Func CPUSpd:Str;
35423: Func CPUSpeed:Str;
35424: Func BigFib(n: Int):Str; //BigFibo
35425: Func BigFac(n: Int):Str; //BigFact
35426: Func UpTime:Str;
35427: Func NetLogon(const Server,User,Password:WideString; out ErrorMessage:str) :Bool;
35428: Func NetLogoff( const Server, User, Password : WideString) :Bool;
35429: Proc ErrorNamedPipe( const Message :Str);
35430: Proc BroadcastChange; //method that broadcasts the necessary message WM_SETTINGCHANGE

```



```

35431: Func WriteFile2(hFile:THandle;const Buffer,nNumberOfBytesToWrite:DWORD; var lpNumberOfBytesWritten:DWORD;
      lpOverlapped : Tobject):BOOL;
35432: Func ReadFile2(hFile:THandle;var Buffer,nNumberOfBytesToRead:DWORD;var
      lpNumberOfBytesRead:DWORD;lpOverlapped : Tobject):BOOL;
35433: // Security helper functions
35434: Proc InitializeSecurity(var SA: TSecurityAttributes);
35435: Proc FinalizeSecurity(var SA: TSecurityAttributes);
35436: // Pipe helper functions
35437: Proc CheckPipeName(Value:Str);
35438: Proc ClearOverlapped(var Overlapped: TOverlapped; ClearEvent:Bool);
35439: Proc CloseHandleClear(var Handle: THandle);
35440: Func ComputerName2:Str;
35441: Proc DisconnectAndClose(Pipe: HPIPE; IsServer:Bool = True);
35442: Func EnumConsoleWindows(Window: HWND; lParam: Int): BOOL; stdcall;
35443: Proc FlushMessages;
35444: Func IsHandle(Handle: THandle):Bool;
35445: Proc RaiseWindowsError;
35446: Func UpTime:Str; //OS uptime
35447: Func CharSetToCP(ACharSet: TFontCharSet): Int;
35448: Func CPToCharSet(ACodePage: Int): TFontCharSet;
35449: Func TwipsToPoints(AValue: Int): Int;
35450: Func PointsToTwips(AValue: Int): Int;
35451: Proc LoadGraphicFromResource(Graphic:TGraphic; const ResName:str; ResType:PChar);
35452: Func SplitStr(sInput:str; Delimiter:str): TStringArray;
35453: Func GetDataFromFile2(sFileName: Ansistr): Ansistr;
35454: Func ExtractFileNameWithoutExt(const FileName:Str):Str;
35455: Func SubstringCount (Substring :Str; Str :Str): Int;
35456: Func loadForm(vx, vy: smallint): TForm; //alias getForm()
35457: Func getForm(vx, vy: smallint): TForm; //alias getForm()
35458: Func getForm2(vx,vy:smallint;acolor:TColor;aname:str):TForm; //sizeable!
35459: Proc paintProcessingstar2(ppform: TForm);
35460: Func ContentTypeGetExtn(Const Content:Str; var CLSID:Str):Str;
35461: Func ContentTypeFromExtn(Const Extension:Str):Str;
35462: Func DateTimeDiff2(Start, Stop : TDateTime) : int64;
35463: Func checkSystem:Str;;
35464: Func getSystemReport:Str;;
35465: Func SystemCheck:Str;;
35466: Func SetThreadDesktop( hDesktop : HDESK) : BOOL;
35467: Func RSAAEncrypt(aval, apow, amod:Str):Str;;
35468: Func RSADecrypt(aval, apow, amod:Str):Str;;
35469: Func EncryptRSA(aval, apow, amod:Str):Str;;
35470: Func DecryptRSA(aval, apow, amod:Str):Str;;
35471: Func SwitchToThread : BOOL;
35472: Func CloseDesktop( hDesktop : HDESK) : BOOL;
35473: Func GetThreadDesktop( dwThreadId : DWORD) : HDESK;
35474: Func SetSyscallHook():Bool;;
35475: Func SetSwapcontextHook():Bool;;
35476: Func UnhookAll():Bool;;
35477: Func getWorld:Str;;
35478: Func getIPConfigAll:Str;;
35479: Func getIPConfig:Str;;
35480: Func WinsockEnabled:Bool;
35481: Func HTTPEncode2(const AStr:Str):Str;
35482: RSAAEncryptStr(const EncryptionType:TRSAEncryptionType;const PublicKey:TRSAPublicKey;const Plain:Ansistr):
      Ansistr;
35483:
35484: //////////////////////////////////////
35485: // Object instance functions
35486: //////////////////////////////////////
35487: Func AllocateHWnd(Method: TWndMethod): HWND;
35488: Proc DeallocateHWnd(Wnd: HWND);
35489: Func DeleteDC( DC : HDC) : BOOL;
35490: Func DeleteMetaFile( pl : HMETAFILE) : BOOL;
35491: Func DeleteObject( pl : HGDIOBJ) : BOOL;
35492: Func FormInstanceCount2(AFormClass: TFormClass): Int;;
35493: Func FormInstanceCount(const AFormClassName:Str): Int;;
35494: ClassN(CL.FindClass('Class of TForm'),'TFormClass;
      Func ObjToStr( const O : Tobject) :Str
35495: MessageBox(0,PChar('CPU speedis '+CPUSpd+' MHz'),'CPU Speed Check',MB_IconInformation+MB_OK);
35497: TKLogEvent', 'Proc (Sender : Tobject; Code : TKLogType; const Text :Str);
35498: AddRegisteredVariable('FormatSettings','TFormatSettings; //at sysutils!
35499: AddRegisteredVariable('mouse','TMouse; //at controls HPIPE', 'THandle;
35500:
35501: Func ListIdentical2(l1,l2:TStringList):Bool;
35502: begin Result:= False;
35503:   if l1.count = l2.count then begin
35504:     for it:= 0 to l1.count-1 do
35505:       if (l1[it] <> l2[it]) then Exit;
35506:     Result:= True;
35507:   end;
35508: end;
35509:
35510: // Converts String To Hexadecimal Maybe usefull for a hex-editor
35511: // For example: Input = 'ABCD' Output = '41 42 43 44'
35512: Func StringtoHex(Data:Str):Str;
35513: var i, i2: Int; s:Str;
35514: begin
35515:   i2:= 1;
35516:   for i:= 1 to Length(Data) do begin

```

```

35517:     Inc(i2);
35518:     if i2 = 2 then
35519:         s:= s + ' '; i2 := 1;
35520:         s:= s + IntToHex(Ord(Data[i]), 2);
35521:     end;
35522:     Result:= s;
35523: end;
35524:
35525: Func XRTLIsInMainThread:Bool;
35526: begin
35527:     Result:= GetCurrentThreadID = MainThreadID;
35528: end;
35529:
35530: computers:= TStringList.Create;
35531: FindComputers(computers)
35532: for it:= 0 to computers.count-1 do
35533:     writeln(computers.strings[it]); computers.free;
35534:
35535: s:= (Sender AS TStringGrid).Cells[col,row];
35536: Sender AS TStringGrid).Canvas.Font.Color := clBlack;
35537: twidth:= (Sender AS TStringGrid).Canvas.TextWidth('X;
35538: (Sender AS TStringGrid).Canvas.TextRect(Rect, Rect.Left + twidth div 2,
35539:     YCenter(Rect, (Sender AS TStringGrid).Canvas,s),s)
35540:
35541: Proc | EXceptionsOnmaXbox; Exception Handling EXceptions On maXbox
35542: var filename,msg:Str;
35543: begin
35544:     filename:= '';
35545:     try
35546:         if filename = '' then
35547:             RaiseException(erCustomError, 'Exception: File name cannot be blank;
35548:         except
35549:             msg:= ExceptionToString(ExceptionType, ExceptionParam);
35550:             //do act with the exception message i.e. email it or save to a log etc
35551:             writeln(msg)
35552:         end;
35553:     end;
35554:
35555: XRaise raise an exception without except block:
35556: if (DLLHandle < HINSTANCE_ERROR) then
35557:     Xraise(Exception.Create(DLLName + ' library can not be loaded or not found.'+
35558:         SysErrorMessage(GetLastError)));
35559:
35560: Proc CallTicker(msecs: Int);
35561: begin
35562:     with TStopWatch.create do begin
35563:         start
35564:         | //Delay2StopWatch(msecs)
35565:         stop
35566:         writeln('Stop Watch CPU Tester: '+getValueStr)
35567:     end;
35568: end;
35569:
35570: MessageDlg Example:
35571: mr:=messagedlg('Save clock first?',mtconfirmation,[mbYes,mbNo,mbCancel],0);
35572: if mr=mrYes then writeln('yes')
35573:
35574: Func getFilefrom
35575: if not (Fileexists('1004unixdict.txt') or fileexists('unixdict.txt')) then
35576:     if IsInternet then begin
35577:         writeln(dictURL)
35578:         wGetX2(dictUrl,Exepath+'1004unixdict.txt');
35579:         openFile(Exepath+'1004unixdict.txt')
35580:     end; //}
35581:
35582: Func IsASCIIDigit(const Ch: Char):Bool;
35583: begin //Result := Ord(Ch) in [Ord('0')..Ord('9')];
35584:     Result:= (Ord(Ch) >= Ord('0')) And (ord(ch) <= Ord('9'));
35585: end;
35586:
35587: Func RemainingBatteryPercent: Int;
35588: var Stat: {Windows.}TSystemPowerStatus;
35589: begin
35590:     {Windows.}GetSystemPowerStatus(Stat);
35591:     Result:= Stat.BatteryLifePercent;
35592:     if (Result < 0) Or (Result > 100) then
35593:         Result:= -1;
35594: end;
35595: // *****
35596: // TIdBaseComponent is the base class for all Indy components.
35597: // *****
35598: type
35599:     TIdBaseComponent = class(TComponent)
3600:     public
36001:         Func GetVersion:Str;
36002:         property Version:Str read GetVersion;
36003:     published
36004:     end;
36005:

```

```

35606: if IsInternet then
35607:   wGetX('http://www.softwareschule.ch/download/maxbox_starter4.pdf','mywgettestpdf.pdf';
35608:   RegEx tester sr := '123.456.789-00$p'
35609:   writeln(ReplaceRegExpr('\D',sr,'',true))
35610:   >>> 12345678900 //D is a non-digit, and \W is a non-word character, both should work)
35611:   ref: lambda: a+b --> a.+(b) in preparations
35612:   namedpipel:= namedpipe as TNamedpipe;
35613:   printf('this is %.18f ',[maxCalc('ln(2)^e')]); this is 0.369248502937272178
35614:   printf('this is Area of r=1 %.18f ',[maxCalc('PI*(1^2)')]);
35615:   printf('this is Area of d=2r %.18f ',[maxCalc('PI/4*(2^2)')]);
35616:   println(ReplaceRegExpr('\d\d{4,12}[]',),',')
35617:   GetSentimentStream7(URLSentimentAPI2,Items[it].title),'',false))
35618:   {"probability": {"neg": 0.5180 "neutral": 0.5201 "pos": 0.4819 , "label": "neutral"}}
35619: begin
35620:   initCRCarray2;
35621:   aStringStream := TStringStream.Create(filetostring(exepath+testFile));
35622:   try
35623:     //aStringStream.loadfromfile
35624:     writeln('CRC32_ : '+IntToHex(CalculateCRC32X(aStringStream), 8));
35625:   finally
35626:     aStringStream.Free;
35627:   end;
35628: end;
35629:
35630: Proc TForm1Findwindow_Remote_Notepad;
35631: var wnd: HWND; i: Int; s:Str;
35632: begin
35633:   wnd := FindWindow('notepad', '');
35634:   if wnd <> 0 then begin
35635:     wnd := FindWindowEx(wnd, 0, 'Edit', '');
35636:     // Write Text in Notepad.
35637:     s := 'Hello maxPad 3 ecotronics'+#13#10+'second line:.';
35638:     for i := 1 to Length(s) do
35639:       SendMessage(wnd, WM_CHAR, Ord(s[i]), 0);
35640:     // Simulate Return Key.
35641:     PostMessage(wnd, WM_KEYDOWN, VK_RETURN, 0);
35642:     // Simulate Space.
35643:     PostMessage(wnd, WM_KEYDOWN, VK_SPACE, 0);
35644:     SendMessage(wnd, WM_CHAR, 5, 0);
35645:     //send time of F5 command in notepad!
35646:     PostMessage(wnd, WM_KEYDOWN, VK_F5, 0);
35647:   end;
35648: end;
35649:
35650: Proc XPATHTest; var XML: OLEVariant;
35651: begin
35652:   XML:= CreateOleObject('MSXML2.DOMDocument.3.0') ;
35653:   XML.async:= false;
35654:   XML.SetProperty('SelectionLanguage','XPath');
35655:   XML:= unassigned;
35656: end;
35657:
35658: with adoQuery do begin //recordset hack
35659:   cachesize:= 500;
35660:   //commandtext:= 'INSERT INTO Table1 (FirstName, LastName, Phone)+'
35661:   'VALUES (''MaXML'', ''Box42810'', ''031-3337788'')';
35662:   commandText:= 'SELECT * FROM Table1';
35663:   connectionString:='Provider=MSDASQL.1;Persist Security Info=False;Data Source=mX4base';
35664:   Open;
35665:   Recordset20(recordset).save(Exepath+'recordset6.xml',adPersistXML)
35666:   writeln(GetISODateString(1995,12,24))
35667:   writeln(botostr(IsValidISODateTimeString(GetISODateString(1995,12,12), true)))
35668:   writeln(MicrosoftCodePageToMIMECharSet(1252))writeln(GetISOTimeString(12,45,33, true));
35669:   writeln(botostr(IsValidISOTimeString(GetISOTimeString(12,45,33, true), true)))
35670:   IDoc:= LoadXMLDocument3(BOOKSXML); //('.\books.xml');
35671:   INode:= SelectXMLNode(IDoc.getDocumentElement,'/bookstore/book[2]/title');
35672:   writeln(FormatXMLFile2String(BOOKSXML));
35673:   writeln(DataSetToXML(TDataset(adoquery)));
35674:
35675: Proc EnableDEP2; // DEP
35676: const PROCESS_DEP_ENABLE: DWORD=$00000001;
35677: var SetProcessDEPPolicy: function(dwFlags: DWORD): BOOL; stdcall;
35678: begin SetProcessDEPPolicy := GetProcAddress(GetModuleHandle(kernel32),'SetProcessDEPPolicy');
35679: if Assigned(SetProcessDEPPolicy) then begin
35680:   //don't bother checking for errors since we don't need to know if it fails
35681:   SetProcessDEPPolicy(PROCESS_DEP_ENABLE);
35682: end;
35683: end;
35684:
35685: unit uPSI_UTGraphSearch; chNGE tnode to tnode2
35686:
35687: procedure SIRegister_TNode2(CL: TPSPascalCompiler);
35688: begin
35689:   //with RegClassS(CL,'tObject', 'TNode2') do
35690:   with CL.AddClassN(CL.FindClass('tObject'),'TNode') do begin
35691:     RegisterProperty('adjacents', 'array of TEdge', iptrw);
35692:     RegisterProperty('nbradjacents', 'integer', iptrw);
35693:     RegisterProperty('index', 'integer', iptrw);
35694:     RegisterProperty('x', 'integer', iptrw);

```

```

35695: RegisterProperty('y', 'integer', iptrw);
35696: RegisterMethod('Constructor create');
35697: end;
35698: end;
35699:
35700: TList.count and capacity!
35701: RegisterProperty('Q', 'TList', iptrw);
35702: type __Pointer; as TObject cause of NIL
35703: FreeAndNilStream (Output); Output : TStream;
35704: FreeAndNilBmp (Output); Output : TBitmap; sign: About!
35705:
35706: Convert a dfm to string: var abt:Bool;
35707: srlst:= TStringlist.create;
35708: if LoadDFMFile2Strings('C:\maxBox\mX42810\backgroundworker\Demo\Primes\Main.dfm', srlst, abt)= 0 then
  writeln(srlst.text); srlst.Free; *) Assert Test: _outd( al.StandardDeviation );
35709: Assert(trunc(al.StandardDeviation)=46, 'must = 46 ');
35710: isloc:= WMIStart; iserv:= WMIConnect(isloc, 'localhost', '', '');
35711: wmi_dom:= 'Win32_NetworkAdapterConfiguration';
35712:
35713: V 4.7.5.90 VI + 4.7.6.10
35714: Amount of Functions: 20549 Amount of Procedures: 12307
35715: Amount of Constructors: 1937 Amount of Destructors: 14
35716: Totals of Calls: 34807 , Size: 31,974,216 bytes
35717: SHA1: of 4.7.5.90 96DCDE2028125E00B67E42A801721AC513A5EAFB CRC32: BBC3A7E5
35718: mX4 executed: 03/11/2021 15:25:47 Runtime: 0:12:33.807 Memload: 39% use
35719: 4.7.6.10 III Totals of Calls: 35218
35720: SHA1: of 4.7.6.10 6647E6200E46702944A6B8175A53EDC1ABEF8E14
35721: CRC32: 1A2922DA: 32,221,000 bytes
35722: mX4 executed: 12/12/2021 10:03:11 Runtime: 0:12:50.630 Memload: 45% use
35723: 4.7.6.10 IV Total of Function Calls: 35351
35724: SHA1: of 4.7.6.10 A9C0D35723F9D2E72E9270207640F938FF58555B
35725: CRC32: 8230EA82 31.3 MB (32,917,832 bytes)
35726: 4.7.6.10 VII
35727: Amount of Functions: 20994 Amount of Procedures: 12492
35728: Amount of Constructors: 1984 Amount of Destructors: 14
35729: Totals of Calls: 35484 SHA1: of 4.7.6.10 EC20D0A371A57AA078762D6D45F7229A370549A1
35730: CRC32: 82E481A4: 33273160 bytes
35731: mX4 executed: 14/03/2022 15:21:38 Runtime: 0:13:41.253 Memload: 42% use
35732:
35733: 17/03/2022 14:08 194,307 IFSI_WinFormIppuzzle.dcu
35734: 17/03/2022 20:30 46,854 uPSC_classes.dcu
35735: 17/03/2022 19:48 5,226 uPSC_std.dcu
35736: 17/03/2022 18:49 53,416 uPSR_classes.dcu
35737: 17/03/2022 20:30 19,528 uPSI_SynEditHighlighter.dcu
35738: 17/03/2022 16:03 11,784 uPSI_SynEditMiscProcs.dcu
35739: 17/03/2022 21:15 173,341 uPSI_TeEngine.dcu
35740: 17/03/2022 21:16 30,854 MathsLib.dcu
35741: 17/03/2022 20:13 77,627 uPSI_neuralvolume.dcu
35742: 17/03/2022 13:10 13,188 uPSI_UcomboV2.dcu
35743: 17/03/2022 13:26 14,813 uPSI_UTGraphSearch.dcu
35744: 17/03/2022 13:22 21,041 UTGraphSearch.dcu
35745:
35746: Release Notes maxBox 4.7.6.10 VIII August 2022 mX476
35747: Amount of Functions: 21179 Amount of Procedures: 12591
35748: Amount of Constructors: 1987 Amount of Destructors: 14
35749: Totals of Calls: 35771
35750: Release Notes maxBox 4.7.6.20 Oct 2022 mX476
35751: Amount of Functions: 21309 Amount of Procedures: 12640
35752: Amount of Constructors: 1989 Amount of Destructors: 14
35753: Totals of Calls: 35952
35754: SHA1: of 4.7.6.20 C0459A5DD0CCC12CE9F3CDEC82E3E0C920D38D9C
35755: CRC32: 498B62AF: 33690392 bytes
35756: mX4 executed: 09/10/2022 20:59:18 Runtime: 0:14:21.810 Memload: 39% use
35757: mXbox4.exe now available as a NFT
35758: sha256: 09eb95c4cc3b7fe60772405b170af4705085da89d190213228c65313c8733499
35759: https://lawrencebarsanti.wordpress.com/2009/11/28/introduction-to-pascal-script/
35760: Amount of Functions: 21538 Amount of Procedures: 12755
35761: Amount of Constructors: 2010 Amount of Destructors: 14
35762: Totals of Calls: 36317
35763: SHA1: of 4.7.6.20 D90222BDC9D9648E75A3A749D4C8EE1DF4655C30
35764: CRC32: 6CB990C1: 34309912 bytes
35765: mX4 executed: 24/01/2023 23:23:35 Runtime: 0:15:16.661 Memload: 44% use
35766: Total of Function Calls: 36317
35767: SHA1: 4.7.6.20 D90222BDC9D9648E75A3A749D4C8EE1DF4655C30
35768: d90222bdc9d9648e75a3a749d4c8ee1df4655c30
35769: CRC32: 6CB990C1 32.7 MB (34,309,912 bytes)
35770: Compilation Timestamp 2023-01-24 21:15:08 UTC Signing time 24 Jan2023 22:18:14
35771: Entry Point 25025064 - Contained Sections 10 sha1: d90222bdc9d9648e75a3a749d4c8ee1df4655c30
35772: sha256: 97943841d5908ea0846dc6763b14941350a0c1aa5d2b6248bfc79ffad7a0314e
35773: https://www.hybrid-analysis.com/
35774: Suspicious Indicators General Found a potential E-Mail address in binary/memory
35775: Network Related Found potential IP address in binary/memory
35776: Remote Access Related Contains indicators of bot communication commands
35777: Contains references to WMI/WMIC Reads terminal service related keys (often RDP related)
35778: Spyware/Information Retrieval Touches files in program files directory
35779: Unusual Characteristics Input file contains API references not part of its Import Address Table (IAT)
35780: Installs hooks/patches the running process
35781: https://www.virustotal.com/gui/file/212f8fda437e877c7f89f771ce40e57814cfb4380ee010c090df78d8dbbc4d9f/behavior

```

```

35782: This report is generated from a file or URL submitted to this webservice on December 15th 2022 18:16:53
      (UTC)
35783: Guest System: Windows 10 64 bit, Professional, 10.0 (build 16299),
35784: Report generated by Falcon Sandbox v9.5.2 © Hybrid Analysis
35785: 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
35786: DetectItEasy PE32 Compiler: Borland Delphi (2007) Linker: Turbo Linker (2.25*,Delphi) [GUI32,signed]
35787: 5.4% (.EXE) Win64 Executable (generic) - 5.1% (.EXE) DOS Borland compiled Executable (generic)
35788:
35789: Total of Function Calls: 36449 SHA1: 4.7.6.50 C56F0A26DFBE706E6A1A9D0E23B6114AFCC09CDC CRC32: 29BCAE4D
      33.1 MB (34,724,120 bytes) Compilation Timestamp 2023-05-21 13:42:16 UTC Signtime 21 May 2023 15:45:22
35790: This report is generated from a file or URL submitted to this webservice on May 21st 2023 18:54:05 (UTC)
35791: Guest System: Windows 7 64 bit, Professional, 6.1 (build 7601), Service Pack 1
35792: Report generated by Falcon Sandbox v10.1.2 © Hybrid Analysis
35793: 60.1% (.OCX) Windows ActiveX control 22.2% (.EXE) InstallShield setup
35794: 5.4% (.EXE) Win64 Executable (generic) 5.1% (.EXE) DOS Borland compiled Executable (generic)
35795: 2.3% (.EXE) Win32 Executable (generic)
35796: Amount of Functions: 21907 Amount of Procedures: 12848
35797: Amount of Constructors: 2029 Amount of Destructors: 14
35798: Totals of Calls: 36798 SHA1: of 4.7.6.50 CAA09249B338ED2814B760FA549DAD47F6C789D2
35799: CRC32: 17A61798: 35033880 bytes
35800: mX4 executed: 30/05/2023 15:46:15 Runtime: 0:16:22.226 Memload: 40% use
35801: Amount of Functions: 21924
35802: Amount of Procedures: 12863
35803: Amount of Constructors: 2033
35804: Amount of Destructors: 14
35805: Totals of Calls: 36834
35806: SHA1: of 4.7.6.50 EDE730F514834A85ECA6D0190DAC8CA6046C26EE
35807: CRC32: 4E7C4A2B: 35117336 bytes
35808: mX4 executed: 31/05/2023 11:30:52 Runtime: 0:15:51.194 Memload: 45% use
35809: Amount of Functions: 21939
35810: Amount of Procedures: 12865
35811: Amount of Constructors: 2034
35812: Amount of Destructors: 14
35813: Totals of Calls: 36852
35814: SHA1: of 4.7.6.50 D4FD4CACFD766EB8F78F2BB7B5EFDDEBB386597A
35815: CRC32: C1DCD693: 35128600 bytes
35816: mX4 executed: 31/05/2023 16:16:38 Runtime: 0:15:26.396 Memload: 38% use
35817:
35818: Amount of Functions: 21987
35819: Amount of Procedures: 12906
35820: Amount of Constructors: 2040
35821: Amount of Destructors: 14
35822: Totals of Calls: 36947
35823: SHA1: of 4.7.6.50 C430FDBA9880317E31D4A2A766C884799298FCC3
35824: CRC32: 5BD9839C: 35257624 bytes
35825: mX4 executed: 06/06/2023 17:16:17 Runtime: 0:15:37.770 Memload: 40% use
35826: Amount of Functions: 22031
35827: Amount of Procedures: 12918
35828: Amount of Constructors: 2045
35829: Amount of Destructors: 14
35830: Totals of Calls: 37008
35831: SHA1: of 4.7.6.50 FDA4E3CE0FEC5F86EF0F95278D5C35FEC42FDA1B
35832: CRC32: 46294349: 35454744 bytes
35833: mX4 executed: 08/06/2023 19:57:04 Runtime: 0:15:51.769 Memload: 46% use
35834: https://www.hybrid-analysis.com/sample/4dfbada6765e47c72b7c2496f831419b3461fa7c4ac6e05e2a941b501e10e022
35835: --- Falcon Sandbox Analysis Summary ---
35836: File Name: maxbox4.exe
35837: Analysis State: SUCCESS
35838: Threat Verdict: no specific threat
35839: Threat Score: n/a/100 AV Detection Ratio: n/a
35840: AV Family Name: Time of analysis: 2023-05-31 14:40:33
35841: File Size (bytes): 35128600
35842: File Type: PE32 executable (GUI) Intel 80386, for MS Windows
35843: Contacted Domains: none Contacted Hosts: none
35844: Environment: Windows 7 64 bit (ID: 120)
35845: --- Falcon Sandbox Analysis Summary ---
35846: File Name: maxbox4.exe Analysis State: SUCCESS
35847: Threat Verdict: no specific threat Threat Score: n/a/100
35848: AV Detection Ratio: n/a AV Family Name:
35849: Time of analysis: 2023-06-06 16:36:58
35850: File Size (bytes): 35257624
35851: File Type: PE32 executable (GUI) Intel 80386, for MS Windows
35852: Contacted Domains: none Contacted Hosts: none
35853: Environment: Windows 7 64 bit (ID: 120)
35854: Submission name: maxbox4.exe
35855: Size: 34MiB Type: peexe executable Mime: application/x-dosexec
35856: SHA256: fa0f30abf34292e91070a5bd4682040eb6af79a8f6c7f5511lc9692153120988 Copy SHA256 to clipboard
35857: Last Anti-Virus Scan: 06/12/2023 14:22:38 (UTC)
35858: Last Sandbox Report: 66/12/2023 14:22:30 (UTC)
35859:
35860: Amount of Functions: 22257
35861: Amount of Procedures: 13051
35862: Amount of Constructors: 2050
35863: Amount of Destructors: 14
35864: Totals of Calls: 37372
35865: SHA1: of 4.7.6.50 D047DBD5412C3E4A436089018B9C7FACF17A2EB5
35866: CRC32: 38562FA8: 35697944 bytes
35867: mX4 executed: 15/06/2023 09:59:46 Runtime: 0:16:2.832 Memload: 40% use

```



```

35868: --- Falcon Sandbox Analysis Summary ---
35869: File Name: maxbox4.exe Analysis State: SUCCESS
35870: Threat Verdict: no specific threat Threat Score: n/a/100
35871: AV Detection Ratio: n/a AV Family Name:
35872: Time of analysis: 2023-06-15 07:31:25 File Size (bytes): 35697944
35873: File Type: PE32 executable (GUI) Intel 80386, for MS Windows
35874: Contacted Domains: none Contacted Hosts: none
35875: Environment: Windows 7 64 bit (ID: 120)
35876: This report is generated from a file or URL submitted to this webservice on June 15th 2023 07:31:25 (UTC)
35877: Guest System: Windows 7 64 bit, Professional, 6.1 (build 7601), Service Pack 1
35878: Report generated by Falcon Sandbox v10.1.5 © Hybrid Analysis
35879: 15/06/2023 08:31 7,947 API_services.dcu
35880: 15/06/2023 08:40 30,857 MathsLib.dcu
35881: 15/06/2023 08:31 81,205 uPSI_GUIUtils.dcu
35882: Recompiled: dir /s /o N *.dcu | find "14/06/2023" > C:\maxbox\maxboxunitalldisk_sort.txt
35883: 14/06/2023 14:35 47,303 uPSC_classes.dcu
35884: 14/06/2023 14:35 54,393 uPSR_classes.dcu
35885: 14/06/2023 15:02 10,731 API_audio.dcu
35886: 14/06/2023 11:54 15,783 API_base.dcu
35887: 14/06/2023 16:31 56,163 API_files.dcu
35888: 14/06/2023 14:54 13,136 API_graphics.dcu
35889: 14/06/2023 14:59 12,145 API_ledgrid.dcu
35890: 14/06/2023 16:31 30,566 api_strings.dcu
35891: 14/06/2023 17:06 40,971 API_tools.dcu
35892: 14/06/2023 18:31 8,219 API_winprocess.dcu
35893: 14/06/2023 18:40 30,876 MathsLib.dcu
35894: 14/06/2023 16:05 72,193 uPSI_GUIAutomation.dcu
35895: 14/06/2023 18:34 65,834 uPSI_GUIUtils.dcu
35896:
35897:
35898: *****
35899: unit List asm internal end units all unitlist unitslist allunits
35900: *****
35901: 01 unit RRegister_StrUtils_Routines(exec); //Delphi
35902: 02 unit SRegister_IdStrings //Indy Sockets
35903: 03 unit RRegister_niSTRING_Routines(Exec); //from Register
35904: 04 unit uPSI_fMain Functions; //maxbox Open Tools API
35905: 05 unit IFSI_WinFormlpuzzle; //maxbox
35906: 06 unit RRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
35907: 07 unit RegisterDateTimeLibrary_R(exec); //Delphi
35908: 08 unit RRegister_MathMax_Routines(exec); //Jedi & Delphi
35909: 09 unit RRegister_IdGlobal_Routines(exec); //Indy Sockets
35910: 10 unit RRegister_SysUtils_Routines(Exec); //Delphi
35911: 11 unit uPSI_IdTCPConnection; //Indy some functions
35912: 12 unit uPSCCompiler.pas; //PS kernel functions
35913: 13 unit uPSI_DBCommon; //DB Common Routines and Types
35914: 14 unit uPSI_Printers.pas //Delphi VCL
35915: 15 unit uPSI_MPlayer.pas //Delphi VCL
35916: 16 unit uPSC_comobj; //COM Functions
35917: 17 unit uPSI_Clipbrd; //Delphi VCL
35918: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
35919: 19 unit uPSI_SqlExpr; //DBX3
35920: 20 unit uPSI_ADODB; //ADODB
35921: 21 unit uPSI_StrHlpr; //String Helper Routines
35922: 22 unit uPSI_DateUtils; //Expansion to DateTimeLib
35923: 23 unit uPSI_FileUtils; //Expansion to Sys/File Utils
35924: 24 unit JUtils / gsUtils; //Jedi / Metabase
35925: 25 unit JvFunctions_max; //Jedi Functions
35926: 26 unit HTTPParser; //Delphi VCL
35927: 27 unit HTTPUtil; //Delphi VCL
35928: 28 unit uPSI_XMLUtil; //Delphi VCL
35929: 29 unit uPSI_SOAPHTTPClient; //Delphi VCL SOAP WebService V3.5
35930: 30 unit uPSI_Contnrs; //Delphi RTL Container of Classes
35931: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
35932: 32 unit uPSI_MyBigInt; //big Int class with Math
35933: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
35934: 34 unit Types_Variants; //Delphi\Win32\rtl\syst
35935: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
35936: 36 unit uPSI_IdHashMessageDigest //Indy Crypto;
35937: 37 unit uPSI_IdASN1Util; //Indy ASN1Utility Routines;
35938: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
35939: 39 unit uPSI_IdIcmpClient; //Indy Ping ICMP
35940: 40 unit uPSI_IdHashMessageDigest_max //Indy Crypto & OpenSSL;
35941: 41 unit uPSI_FileCtrl; //Delphi RTL
35942: 42 unit uPSI_Outline; //Delphi VCL
35943: 43 unit uPSI_ScktComp; //Delphi RTL
35944: 44 unit uPSI_Calendar; //Delphi VCL
35945: 45 unit uPSI_VListView //VListView;
35946: 46 unit uPSI_DBGGrids; //Delphi VCL
35947: 47 unit uPSI_DBCtrls; //Delphi VCL
35948: 48 unit ide_debugoutput; //maxbox
35949: 49 unit uPSI_ComCtrls; //Delphi VCL
35950: 50 unit uPSC_stdctrls; //Delphi VCL
35951: 51 unit uPSI_Dialogs; //Delphi VCL
35952: 52 unit uPSI_StdConvs; //Delphi RTL
35953: 53 unit uPSI_DBClient; //Delphi RTL
35954: 54 unit uPSI_DBPlatform; //Delphi RTL
35955: 55 unit uPSI_Provider; //Delphi RTL
35956: 56 unit uPSI_FMTBcd; //Delphi RTL

```

```
35957: 57 unit uPSI_DBCGrids; //Delphi VCL
35958: 58 unit uPSI_CDSUtil; //MIDAS
35959: 59 unit uPSI_VarHlpr; //Delphi RTL
35960: 60 unit uPSI_ExtDlgs; //Delphi VCL
35961: 61 unit sdpStopwatch; //maXbox
35962: 62 unit uPSI_JclStatistics; //JCL
35963: 63 unit uPSI_JclLogic; //JCL
35964: 64 unit uPSI_JclMiscel; //JCL
35965: 65 unit uPSI_JclMath_max; //JCL RTL
35966: 66 unit uPSI_uTPLb_StreamUtils; //LockBox 3
35967: 67 unit uPSI_MathUtils; //BCB
35968: 68 unit uPSI_JclMultimedia; //JCL
35969: 69 unit uPSI_WideStrUtils; //Delphi API/RTL
35970: 70 unit uPSI_GraphUtil; //Delphi RTL
35971: 71 unit uPSI_TypeTrans; //Delphi RTL
35972: 72 unit uPSI_HTTPApp; //Delphi VCL
35973: 73 unit uPSI_DBWeb; //Delphi VCL
35974: 74 unit uPSI_DBBdeWeb; //Delphi VCL
35975: 75 unit uPSI_DBXpressWeb; //Delphi VCL
35976: 76 unit uPSI_ShadowWnd; //Delphi VCL
35977: 77 unit uPSI_ToolWin; //Delphi VCL
35978: 78 unit uPSI_Tabs; //Delphi VCL
35979: 79 unit uPSI_JclGraphUtils; //JCL
35980: 80 unit uPSI_JclCounter; //JCL
35981: 81 unit uPSI_JclSysInfo; //JCL
35982: 82 unit uPSI_JclSecurity; //JCL
35983: 83 unit uPSI_JclFileUtils; //JCL
35984: 84 unit uPSI_IdUserAccounts; //Indy
35985: 85 unit uPSI_IdAuthentication; //Indy
35986: 86 unit uPSI_uTPLb_AES; //LockBox 3
35987: 87 unit uPSI_IdHashSHA1; //LockBox 3
35988: 88 unit uTPLb_BlockCipher; //LockBox 3
35989: 89 unit uPSI_ValEdit.pas; //Delphi VCL
35990: 90 unit uPSI_JvVCLUtils; //JCL
35991: 91 unit uPSI_JvDBUtil; //JCL
35992: 92 unit uPSI_JvDBUtils; //JCL
35993: 93 unit uPSI_JvAppUtils; //JCL
35994: 94 unit uPSI_JvCtrlUtils; //JCL
35995: 95 unit uPSI_JvFormToHtml; //JCL
35996: 96 unit uPSI_JvParsing; //JCL
35997: 97 unit uPSI_SerDlgs; //Toolbox
35998: 98 unit uPSI_Serial; //Toolbox
35999: 99 unit uPSI_JvComponent; //JCL
36000: 100 unit uPSI_JvCalc; //JCL
36001: 101 unit uPSI_JvBdeUtils; //JCL
36002: 102 unit uPSI_JvDateUtil; //JCL
36003: 103 unit uPSI_JvGenetic; //JCL
36004: 104 unit uPSI_JclBase; //JCL
36005: 105 unit uPSI_JvUtils; //JCL
36006: 106 unit uPSI_JvStrUtil; //JCL
36007: 107 unit uPSI_JvStrUtils; //JCL
36008: 108 unit uPSI_JvFileUtil; //JCL
36009: 109 unit uPSI_JvMemoryInfos; //JCL
36010: 110 unit uPSI_JvComputerInfo; //JCL
36011: 111 unit uPSI_JvgCommClasses; //JCL
36012: 112 unit uPSI_JvgLogics; //JCL
36013: 113 unit uPSI_JvLED; //JCL
36014: 114 unit uPSI_JvTurtle; //JCL
36015: 115 unit uPSI_SortThds; unit uPSI_ThSort; //maXbox
36016: 116 unit uPSI_JvgUtils; //JCL
36017: 117 unit uPSI_JvExprParser; //JCL
36018: 118 unit uPSI_HexDump; //Borland
36019: 119 unit uPSI_DBLogDlg; //VCL
36020: 120 unit uPSI_SqlTimSt; //RTL
36021: 121 unit uPSI_JvHtmlParser; //JCL
36022: 122 unit uPSI_JvgXMLSerializer; //JCL
36023: 123 unit uPSI_JvJCLUtils; //JCL
36024: 124 unit uPSI_JvStrings; //JCL
36025: 125 unit uPSI_uTPLb_IntUtils; //TurboPower
36026: 126 unit uPSI_uTPLb_HugeCardinal; //TurboPower
36027: 127 unit uPSI_uTPLb_HugeCardinalUtils; //TurboPower
36028: 128 unit uPSI_SynRegExpr; //SynEdit
36029: 129 unit uPSI_StUtils; //SysTools4
36030: 130 unit uPSI_StToHTML; //SysTools4
36031: 131 unit uPSI_StStrms; //SysTools4
36032: 132 unit uPSI_StFIN; //SysTools4
36033: 133 unit uPSI_StAstroP; //SysTools4
36034: 134 unit uPSI_StStat; //SysTools4
36035: 135 unit uPSI_StNetCon; //SysTools4
36036: 136 unit uPSI_StDecMth; //SysTools4
36037: 137 unit uPSI_StOStr; //SysTools4
36038: 138 unit uPSI_StPtrns; //SysTools4
36039: 139 unit uPSI_StNetMessage; //SysTools4
36040: 140 unit uPSI_StMath; //SysTools4
36041: 141 unit uPSI_StExpEng; //SysTools4
36042: 142 unit uPSI_StCRC; //SysTools4
36043: 143 unit uPSI_StExport; //SysTools4
36044: 144 unit uPSI_StExpLog; //SysTools4
36045: 145 unit uPSI_ActnList; //Delphi VCL
```

```

36046: 146 unit uPSI_jpeg; //Borland
36047: 147 unit uPSI_StRandom; //SysTools4
36048: 148 unit uPSI_StDict; //SysTools4
36049: 149 unit uPSI_StBCD; //SysTools4
36050: 150 unit uPSI_StTxtDat; //SysTools4
36051: 151 unit uPSI_StRegex; //SysTools4
36052: 152 unit uPSI_IMouse; //VCL
36053: 153 unit uPSI_SyncObjs; //VCL
36054: 154 unit uPSI_AsyncCalls; //Hausladen
36055: 155 unit uPSI_ParallelJobs; //Saraiva
36056: 156 unit uPSI_Variants; //VCL
36057: 157 unit uPSI_VarCmplx; //VCL Wolfram
36058: 158 unit uPSI_DTDSchema; //VCL
36059: 159 unit uPSI_ShLwApi; //Brakel
36060: 160 unit uPSI_IBUtils; //VCL
36061: 161 unit uPSI_CheckLst; //VCL
36062: 162 unit uPSI_JvSimpleXml; //JCL
36063: 163 unit uPSI_JclSimpleXml; //JCL
36064: 164 unit uPSI_JvXmlDatabase; //JCL
36065: 165 unit uPSI_JvMaxPixel; //JCL
36066: 166 unit uPSI_JvItemsSearchs; //JCL
36067: 167 unit uPSI_StExpEng2; //SysTools4
36068: 168 unit uPSI_StGenLog; //SysTools4
36069: 169 unit uPSI_JvLogFile; //Jcl
36070: 170 unit uPSI_CPort; //ComPort Lib v4.11
36071: 171 unit uPSI_CPortCtl; //ComPort
36072: 172 unit uPSI_CPortEsc; //ComPort
36073: 173 unit BarCodeScanner; //ComPort
36074: 174 unit uPSI_JvGraph; //JCL
36075: 175 unit uPSI_JvComCtrls; //JCL
36076: 176 unit uPSI_GUITesting; //D Unit
36077: 177 unit uPSI_JvFindFiles; //JCL
36078: 178 unit uPSI_StSystem; //SysTools4
36079: 179 unit uPSI_JvKeyboardStates; //JCL
36080: 180 unit uPSI_JvMail; //JCL
36081: 181 unit uPSI_JclConsole; //JCL
36082: 182 unit uPSI_JclLANMan; //JCL
36083: 183 unit uPSI_IdCustomHTTPServer; //Indy
36084: 184 unit IdHTTPServer //Indy
36085: 185 unit uPSI_IdTCPServer; //Indy
36086: 186 unit uPSI_IdSocketHandle; //Indy
36087: 187 unit uPSI_IdIOHandlerSocket; //Indy
36088: 188 unit IdIOHandler; //Indy
36089: 189 unit uPSI_cutils; //Bloodshed
36090: 190 unit uPSI_BoldUtils; //boldsoft
36091: 191 unit uPSI_IdSimpleServer; //Indy
36092: 192 unit uPSI_IdSSLOpenSSL; //Indy
36093: 193 unit uPSI_IdMultipartFormData; //Indy
36094: 194 unit uPSI_SynURI opener; //SynEdit
36095: 195 unit uPSI_PperlRegEx; //PCRE
36096: 196 unit uPSI_IdHeaderList; //Indy
36097: 197 unit uPSI_StFirst; //SysTools4
36098: 198 unit uPSI_JvCtrls; //JCL
36099: 199 unit uPSI_IdTrivialFTPBase; //Indy
36100: 200 unit uPSI_IdTrivialFTP; //Indy
36101: 201 unit uPSI_IdUDFBase; //Indy
36102: 202 unit uPSI_IdUDFClient; //Indy
36103: 203 unit uPSI_utypes; //for DMATH.DLL
36104: 204 unit uPSI_ShellAPI; //Borland
36105: 205 unit uPSI_IdRemoteCMDClient; //Indy
36106: 206 unit uPSI_IdRemoteCMDServer; //Indy
36107: 207 unit IdRexecServer; //Indy
36108: 208 unit IdRexec; (unit uPSI_IdRexec;) //Indy
36109: 209 unit IdUDFServer; //Indy
36110: 210 unit IdTimeUDFServer; //Indy
36111: 211 unit IdTimeServer; //Indy
36112: 212 unit IdTimeUDP; (unit uPSI_IdUDFServer;) //Indy
36113: 213 unit uPSI_IdIPWatch; //Indy
36114: 214 unit uPSI_IdIrcServer; //Indy
36115: 215 unit uPSI_IdMessageCollection; //Indy
36116: 216 unit uPSI_cPEM; //Fundamentals 4
36117: 217 unit uPSI_cFundamentUtils; //Fundamentals 4
36118: 218 unit uPSI_uwinplot; //DMATH
36119: 219 unit uPSI_xrtl_util_CPUUtils; //ExtendedRTL
36120: 220 unit uPSI_GR32_System; //Graphics32
36121: 221 unit uPSI_cFileUtils; //Fundamentals 4
36122: 222 unit uPSI_cDateTime; (timemachine) //Fundamentals 4
36123: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
36124: 224 unit uPSI_cRandom; //Fundamentals 4
36125: 225 unit uPSI_ueval; //DMATH
36126: 226 unit uPSI_xrtl_net_URIUtils; //ExtendedRTL
36127: 227 unit xrtl_net_URIUtils; //ExtendedRTL
36128: 228 unit uPSI_ufft; (FFT) //DMATH
36129: 229 unit uPSI_DBXChannel; //Delphi
36130: 230 unit uPSI_DBXIndyChannel; //Delphi Indy
36131: 231 unit uPSI_xrtl_util_COMCat; //ExtendedRTL
36132: 232 unit uPSI_xrtl_util_StrUtils; //ExtendedRTL
36133: 233 unit uPSI_xrtl_util_VariantUtils; //ExtendedRTL
36134: 234 unit uPSI_xrtl_util_FileUtils; //ExtendedRTL

```

```
36135: 235 unit xrtl_util_Compat; //ExtendedRTL
36136: 236 unit uPSI_OleAuto; //Borland
36137: 237 unit uPSI_xrtl_util_COMUtils; //ExtendedRTL
36138: 238 unit uPSI_CmAdmCtl; //Borland
36139: 239 unit uPSI_ValEdit2; //VCL
36140: 240 unit uPSI_GR32; //Graphics32
36141: 241 unit uPSI_GR32_Image; //Graphics32
36142: 242 unit uPSI_xrtl_util_TimeUtils; //ExtendedRTL
36143: 243 unit uPSI_xrtl_util_TimeZone; //ExtendedRTL
36144: 244 unit uPSI_xrtl_util_TimeStamp; //ExtendedRTL
36145: 245 unit uPSI_xrtl_util_Map; //ExtendedRTL
36146: 246 unit uPSI_xrtl_util_Set; //ExtendedRTL
36147: 247 unit uPSI_CPortMonitor; //ComPort
36148: 248 unit uPSI_StIniStm; //SysTools4
36149: 249 unit uPSI_GR32_ExtImage; //Graphics32
36150: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32
36151: 251 unit uPSI_GR32_Rasterizers; //Graphics32
36152: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
36153: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
36154: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
36155: 255 unit uPSI_FlatSB; //VCL
36156: 256 unit uPSI_JvAnalogClock; //JCL
36157: 257 unit uPSI_JvAlarms; //JCL
36158: 258 unit uPSI_JvSQLS; //JCL
36159: 259 unit uPSI_JvDBSecur; //JCL
36160: 260 unit uPSI_JvDBQBE; //JCL
36161: 261 unit uPSI_JvStarfield; //JCL
36162: 262 unit uPSI_JVCLMiscal; //JCL
36163: 263 unit uPSI_JvProfiler32; //JCL
36164: 264 unit uPSI_JvDirectories; //JCL
36165: 265 unit uPSI_JclSchedule; //JCL
36166: 266 unit uPSI_JclSvcCtrl; //JCL
36167: 267 unit uPSI_JvSoundControl; //JCL
36168: 268 unit uPSI_JvBDESQLScript; //JCL
36169: 269 unit uPSI_JvgDigits; //JCL>
36170: 270 unit uPSI_ImgList; //TCustomImageList
36171: 271 unit uPSI_JclMIDI; //JCL>
36172: 272 unit uPSI_JclWinMidi; //JCL>
36173: 273 unit uPSI_JclNTFS; //JCL>
36174: 274 unit uPSI_JclAppInst; //JCL>
36175: 275 unit uPSI_JvRle; //JCL>
36176: 276 unit uPSI_JvRas32; //JCL>
36177: 277 unit uPSI_JvImageDrawThread; //JCL>
36178: 278 unit uPSI_JvImageWindow; //JCL>
36179: 279 unit uPSI_JvTransparentForm; //JCL>
36180: 280 unit uPSI_JvWinDialogs; //JCL>
36181: 281 unit uPSI_JvSimLogic; //JCL>
36182: 282 unit uPSI_JvSimIndicator; //JCL>
36183: 283 unit uPSI_JvSimPID; //JCL>
36184: 284 unit uPSI_JvSimPIDLinker; //JCL>
36185: 285 unit uPSI_IdRFCReply; //Indy
36186: 286 unit uPSI_IdIdent; //Indy
36187: 287 unit uPSI_IdIdentServer; //Indy
36188: 288 unit uPSI_JvPatchFile; //JCL
36189: 289 unit uPSI_StNetPfm; //SysTools4
36190: 290 unit uPSI_StNet; //SysTools4
36191: 291 unit uPSI_JclPeImage; //JCL
36192: 292 unit uPSI_JclPrint; //JCL
36193: 293 unit uPSI_JclMime; //JCL
36194: 294 unit uPSI_JvRichEdit; //JCL
36195: 295 unit uPSI_JvDBRichEd; //JCL
36196: 296 unit uPSI_JvDice; //JCL
36197: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
36198: 298 unit uPSI_JvDirFrm; //JCL
36199: 299 unit uPSI_JvDualList; //JCL
36200: 300 unit uPSI_JvSwitch; ///JCL
36201: 301 unit uPSI_JvTimerLst; ///JCL
36202: 302 unit uPSI_JvMemTable; //JCL
36203: 303 unit uPSI_JvObjStr; //JCL
36204: 304 unit uPSI_StLArr; //SysTools4
36205: 305 unit uPSI_StWmDCpy; //SysTools4
36206: 306 unit uPSI_StText; //SysTools4
36207: 307 unit uPSI_StNTLog; //SysTools4
36208: 308 unit uPSI_xrtl_math_Int; //ExtendedRTL
36209: 309 unit uPSI_JvImagPrvw; //JCL
36210: 310 unit uPSI_JvFormPatch; //JCL
36211: 311 unit uPSI_JvPicClip; //JCL
36212: 312 unit uPSI_JvDataConv; //JCL
36213: 313 unit uPSI_JvCpuUsage; //JCL
36214: 314 unit uPSI_JclUnitConv_mX2; //JCL
36215: 315 unit JvDualListForm; //JCL
36216: 316 unit uPSI_JvCpuUsage2; //JCL
36217: 317 unit uPSI_JvParserForm; //JCL
36218: 318 unit uPSI_JvJanTreeView; //JCL
36219: 319 unit uPSI_JvTransLED; //JCL
36220: 320 unit uPSI_JvPlaylist; //JCL
36221: 321 unit uPSI_JvFormAutoSize; //JCL
36222: 322 unit uPSI_JvYearGridEditForm; //JCL
36223: 323 unit uPSI_JvMarkupCommon; //JCL
```

```

36224: 324 unit uPSI_JvChart; //JCL
36225: 325 unit uPSI_JvXPCore; //JCL
36226: 326 unit uPSI_JvXPCoreUtils; //JCL
36227: 327 unit uPSI_StatsClasses; //mX4 al: TStatArray;
36228: 328 unit uPSI_ExtCtrls2; //VCL
36229: 329 unit uPSI_JvUrlGrabbers; //JCL
36230: 330 unit uPSI_JvXmlTree; //JCL
36231: 331 unit uPSI_JvWavePlayer; //JCL
36232: 332 unit uPSI_JvUnicodeCanvas; //JCL
36233: 333 unit uPSI_JvTFUtils; //JCL
36234: 334 unit uPSI_IdServerIOHandler; //Indy
36235: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
36236: 336 unit uPSI_IdMessageCoder; //Indy
36237: 337 unit uPSI_IdMessageCoderMIME; //Indy
36238: 338 unit uPSI_IdMIMETypes; //Indy
36239: 339 unit uPSI_JvConverter; //JCL
36240: 340 unit uPSI_JvCsvParse; //JCL
36241: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
36242: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
36243: 343 unit uPSI_JvDBGridExport; //JCL
36244: 344 unit uPSI_JvgExport; //JCL
36245: 345 unit uPSI_JvSerialMaker; //JCL
36246: 346 unit uPSI_JvWin32; //JCL
36247: 347 unit uPSI_JvPaintFX; //JCL
36248: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
36249: 349 unit uPSI_JvValidators; (preview) //JCL
36250: 350 unit uPSI_JvNTEventLog; //JCL
36251: 351 unit uPSI_ShellZipTool; //mX4
36252: 352 unit uPSI_JvJoystick; //JCL
36253: 353 unit uPSI_JvMailSlots; //JCL
36254: 354 unit uPSI_JclComplex; //JCL
36255: 355 unit uPSI_SynPdf; //Synopsis
36256: 356 unit uPSI_Registry; //VCL
36257: 357 unit uPSI_TlHelp32; //VCL
36258: 358 unit uPSI_JclRegistry; //JCL
36259: 359 unit uPSI_JvAirBrush; //JCL
36260: 360 unit uPSI_mORMotReport; //Synopsis
36261: 361 unit uPSI_JclLocales; //JCL
36262: 362 unit uPSI_SynEdit; //SynEdit
36263: 363 unit uPSI_SynEditTypes; //SynEdit
36264: 364 unit uPSI_SynMacroRecorder; //SynEdit
36265: 365 unit uPSI_LongIntList; //SynEdit
36266: 366 unit uPSI_devcutils; //DevC
36267: 367 unit uPSI_SynEditMiscClasses; //SynEdit
36268: 368 unit uPSI_SynEditRegexSearch; //SynEdit
36269: 369 unit uPSI_SynEditHighlighter; //SynEdit
36270: 370 unit uPSI_SynHighlighterPas; //SynEdit
36271: 371 unit uPSI_JvSearchFiles; //JCL
36272: 372 unit uPSI_SynHighlighterAny; //Lazarus
36273: 373 unit uPSI_SynEditKeyCmds; //SynEdit
36274: 374 unit uPSI_SynEditMiscProcs; //SynEdit
36275: 375 unit uPSI_SynEditKbdHandler; //SynEdit
36276: 376 unit uPSI_JvAppInst; //JCL
36277: 377 unit uPSI_JvAppEvent; //JCL
36278: 378 unit uPSI_JvAppCommand; //JCL
36279: 379 unit uPSI_JvAnimTitle; //JCL
36280: 380 unit uPSI_JvAnimatedImage; //JCL
36281: 381 unit uPSI_SynEditExport; //SynEdit
36282: 382 unit uPSI_SynExportHTML; //SynEdit
36283: 383 unit uPSI_SynExportRTF; //SynEdit
36284: 384 unit uPSI_SynEditSearch; //SynEdit
36285: 385 unit uPSI_fMain_back //maXbox;
36286: 386 unit uPSI_JvZoom; //JCL
36287: 387 unit uPSI_PMrand; //PM
36288: 388 unit uPSI_JvSticker; //JCL
36289: 389 unit uPSI_XmlVerySimple; //mX4
36290: 390 unit uPSI_Services; //ExtPascal
36291: 391 unit uPSI_ExtPascalUtils; //ExtPascal
36292: 392 unit uPSI_SocketsDelphi; //ExtPascal
36293: 393 unit uPSI_StBarC; //SysTools
36294: 394 unit uPSI_StDbBarC; //SysTools
36295: 395 unit uPSI_StBarPN; //SysTools
36296: 396 unit uPSI_StDbPNBC; //SysTools
36297: 397 unit uPSI_StDb2DBC; //SysTools
36298: 398 unit uPSI_StMoney; //SysTools
36299: 399 unit uPSI_JvForth; //JCL
36300: 400 unit uPSI_RestRequest; //mX4
36301: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
36302: 402 unit uPSI_JvXmlDatabase; //update //JCL
36303: 403 unit uPSI_StAstro; //SysTools
36304: 404 unit uPSI_StSort; //SysTools
36305: 405 unit uPSI_StDate; //SysTools
36306: 406 unit uPSI_StDateSt; //SysTools
36307: 407 unit uPSI_StBase; //SysTools
36308: 408 unit uPSI_StVInfo; //SysTools
36309: 409 unit uPSI_JvBrowseFolder; //JCL
36310: 410 unit uPSI_JvBoxProcs; //JCL
36311: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
36312: 412 unit uPSI_usimann; (unit ugenalg;) //DMath

```



```

36313: 413 unit uPSI_JvHighlighter; //JCL
36314: 414 unit uPSI_Diff; //mX4
36315: 415 unit uPSI_SpringWinAPI; //DSpring
36316: 416 unit uPSI_StBits; //SysTools
36317: 417 unit uPSI_TomDBQueue; //mX4
36318: 418 unit uPSI_MultilangTranslator; //mX4
36319: 419 unit uPSI_HyperLabel; //mX4
36320: 420 unit uPSI_Starter; //mX4
36321: 421 unit uPSI_FileAssocs; //devC
36322: 422 unit uPSI_devFileMonitorX; //devC
36323: 423 unit uPSI_devrun; //devC
36324: 424 unit uPSI_devExec; //devC
36325: 425 unit uPSI_oysUtils; //devC
36326: 426 unit uPSI_DosCommand; //devC
36327: 427 unit uPSI_CppTokenizer; //devC
36328: 428 unit uPSI_JvHLParser; //devC
36329: 429 unit uPSI_JclMapi; //JCL
36330: 430 unit uPSI_JclShell; //JCL
36331: 431 unit uPSI_JclCOM; //JCL
36332: 432 unit uPSI_GR32_Math; //Graphics32
36333: 433 unit uPSI_GR32_LowLevel; //Graphics32
36334: 434 unit uPSI_SimpleHl; //mX4
36335: 435 unit uPSI_GR32_Filters; //Graphics32
36336: 436 unit uPSI_GR32_VectorMaps; //Graphics32
36337: 437 unit uPSI_cXMLFunctions; //Fundamentals 4
36338: 438 unit uPSI_JvTimer; //JCL
36339: 439 unit uPSI_cHTTPUtils; //Fundamentals 4
36340: 440 unit uPSI_cTLSUtils; //Fundamentals 4
36341: 441 unit uPSI_JclGraphics; //JCL
36342: 442 unit uPSI_JclSynch; //JCL
36343: 443 unit uPSI_IdTelnet; //Indy
36344: 444 unit uPSI_IdTelnetServer; //Indy
36345: 445 unit uPSI_IdEcho; //Indy
36346: 446 unit uPSI_IdEchoServer; //Indy
36347: 447 unit uPSI_IdEchoUDP; //Indy
36348: 448 unit uPSI_IdEchoUDPServer; //Indy
36349: 449 unit uPSI_IdSocks; //Indy
36350: 450 unit uPSI_IdAntiFreezeBase; //Indy
36351: 451 unit uPSI_IdHostnameServer; //Indy
36352: 452 unit uPSI_IdTunnelCommon; //Indy
36353: 453 unit uPSI_IdTunnelMaster; //Indy
36354: 454 unit uPSI_IdTunnelSlave; //Indy
36355: 455 unit uPSI_IdRSH; //Indy
36356: 456 unit uPSI_IdRSHServer; //Indy
36357: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi
36358: 458 unit uPSI_MapReader; //devC
36359: 459 unit uPSI_LibTar; //devC
36360: 460 unit uPSI_IdStack; //Indy
36361: 461 unit uPSI_IdBlockCipherIntercept; //Indy
36362: 462 unit uPSI_IdChargenServer; //Indy
36363: 463 unit uPSI_IdFTPServer; //Indy
36364: 464 unit uPSI_IdException; //Indy
36365: 465 unit uPSI_utexplore; //DMath
36366: 466 unit uPSI_uwinstr; //DMath
36367: 467 unit uPSI_VarRecUtils; //devC
36368: 468 unit uPSI_JvStringListToHtml; //JCL
36369: 469 unit uPSI_JvStringHolder; //JCL
36370: 470 unit uPSI_IdCoder; //Indy
36371: 471 unit uPSI_SynHighlighterDfm; //Synedit
36372: 472 unit uHighlighterProcs; in 471 //Synedit
36373: 473 unit uPSI_LazFileUtils; //LCL
36374: 474 unit uPSI_IDECmdLine; //LCL
36375: 475 unit uPSI_lazMasks; //LCL
36376: 476 unit uPSI_ip_misc; //mX4
36377: 477 unit uPSI_Barcode; //LCL
36378: 478 unit uPSI_SimpleXML; //LCL
36379: 479 unit uPSI_JclIniFiles; //JCL
36380: 480 unit uPSI_D2XXUnit; {$X-} //FTDI
36381: 481 unit uPSI_JclDateTime; //JCL
36382: 482 unit uPSI_JclEDI; //JCL
36383: 483 unit uPSI_JclMiscel2; //JCL
36384: 484 unit uPSI_JclValidation; //JCL
36385: 485 unit uPSI_JclAnsistrs; {-PString} //JCL
36386: 486 unit uPSI_SynEditMiscProcs2; //Synedit
36387: 487 unit uPSI_JclStreams; //JCL
36388: 488 unit uPSI_QRCode; //mX4
36389: 489 unit uPSI_BlockSocket; //ExtPascal
36390: 490 unit uPSI_Masks_Utils; //VCL
36391: 491 unit uPSI_synauttil; //Synapse!
36392: 492 unit uPSI_JclMath_Class; //JCL RTL
36393: 493 unit ugamdist; //Gamma Func //DMath
36394: 494 unit uibeta, ucorrel; //IBeta //DMath
36395: 495 unit uPSI_SRMgr; //mX4
36396: 496 unit uPSI_HotLog; //mX4
36397: 497 unit uPSI_DebugBox; //mX4
36398: 498 unit uPSI_ustrings; //DMath
36399: 499 unit uPSI_uregtest; //DMath
36400: 500 unit uPSI_usimplex; //DMath
36401: 501 unit uPSI_uhyper; //DMath

```

```

36402: 502 unit uPSI_IdHLL7; //Indy
36403: 503 unit uPSI_IdIPMCastBase; //Indy
36404: 504 unit uPSI_IdIPMCastServer; //Indy
36405: 505 unit uPSI_IdIPMCastClient; //Indy
36406: 506 unit uPSI_unlfit; //nlregression //DMath
36407: 507 unit uPSI_IdRawHeaders; //Indy
36408: 508 unit uPSI_IdRawClient; //Indy
36409: 509 unit uPSI_IdRawFunctions; //Indy
36410: 510 unit uPSI_IdTCPStream; //Indy
36411: 511 unit uPSI_IdSNPP; //Indy
36412: 512 unit uPSI_St2DBarC; //SysTools
36413: 513 unit uPSI_ImageWin; //FTL //VCL
36414: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
36415: 515 unit uPSI_GraphWin; //FTL //VCL
36416: 516 unit uPSI_actionMain; //FTL //VCL
36417: 517 unit uPSI_StSpawn; //SysTools
36418: 518 unit uPSI_CtlPanel; //VCL
36419: 519 unit uPSI_IdLPR; //Indy
36420: 520 unit uPSI_SockRequestInterpreter; //Indy
36421: 521 unit uPSI_ulambert; //DMath
36422: 522 unit uPSI_ucholesk; //DMath
36423: 523 unit uPSI_SimpleDS; //VCL
36424: 524 unit uPSI_DBXSqlScanner; //VCL
36425: 525 unit uPSI_DBXMetaDataUtil; //VCL
36426: 526 unit uPSI_Chart; //TEE
36427: 527 unit uPSI_TeeProcs; //TEE
36428: 528 unit mXBDEUtils; //mX4
36429: 529 unit uPSI_MDIEdit; //VCL
36430: 530 unit uPSI_CopyPrsr; //VCL
36431: 531 unit uPSI_SockApp; //VCL
36432: 532 unit uPSI_AppEvnts; //VCL
36433: 533 unit uPSI_ExtActns; //VCL
36434: 534 unit uPSI_TeEngine; //TEE
36435: 535 unit uPSI_CoolMain; //browser //VCL
36436: 536 unit uPSI_StCRC; //SysTools
36437: 537 unit uPSI_StDecMth2; //SysTools
36438: 538 unit uPSI_frmExportMain; //Synedit
36439: 539 unit uPSI_SynDBEdit; //Synedit
36440: 540 unit uPSI_SynEditWildcardSearch; //Synedit
36441: 541 unit uPSI_BoldComUtils; //BOLD
36442: 542 unit uPSI_BoldIsoDateTime; //BOLD
36443: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
36444: 544 unit uPSI_BoldXMLRequests; //BOLD
36445: 545 unit uPSI_BoldStringList; //BOLD
36446: 546 unit uPSI_BoldFileHandler; //BOLD
36447: 547 unit uPSI_BoldContainers; //BOLD
36448: 548 unit uPSI_BoldQueryUserDlg; //BOLD
36449: 549 unit uPSI_BoldWinINet; //BOLD
36450: 550 unit uPSI_BoldQueue; //BOLD
36451: 551 unit uPSI_JvPcx; //JCL
36452: 552 unit uPSI_IdWhois; //Indy
36453: 553 unit uPSI_IdWhoIsServer; //Indy
36454: 554 unit uPSI_IdGopher; //Indy
36455: 555 unit uPSI_IdDateTimeStamp; //Indy
36456: 556 unit uPSI_IdDayTimeServer; //Indy
36457: 557 unit uPSI_IdDayTimeUDP; //Indy
36458: 558 unit uPSI_IdDayTimeUDPServer; //Indy
36459: 559 unit uPSI_IdDICTServer; //Indy
36460: 560 unit uPSI_IdDiscardServer; //Indy
36461: 561 unit uPSI_IdDiscardUDPServer; //Indy
36462: 562 unit uPSI_IdMappedFTP; //Indy
36463: 563 unit uPSI_IdMappedPortTCP; //Indy
36464: 564 unit uPSI_IdGopherServer; //Indy
36465: 565 unit uPSI_IdQotdServer; //Indy
36466: 566 unit uPSI_JvRgbToHtml; //JCL
36467: 567 unit uPSI_JvRemLog; //JCL
36468: 568 unit uPSI_JvSysComp; //JCL
36469: 569 unit uPSI_JvTMTL; //JCL
36470: 570 unit uPSI_JvWinampAPI; //JCL
36471: 571 unit uPSI_MSysUtils; //mX4
36472: 572 unit uPSI_ESBMaths; //ESB
36473: 573 unit uPSI_ESBMaths2; //ESB
36474: 574 unit uPSI_uLkJJSON; //Lk
36475: 575 unit uPSI_ZURL; //Zeos //Zeos
36476: 576 unit uPSI_ZSysUtils; //Zeos
36477: 577 unit uPSI_unaUtils internals //UNA
36478: 578 unit uPSI_ZMatchPattern; //Zeos
36479: 579 unit uPSI_ZClasses; //Zeos
36480: 580 unit uPSI_ZCollections; //Zeos
36481: 581 unit uPSI_ZEncoding; //Zeos
36482: 582 unit uPSI_IdRawBase; //Indy
36483: 583 unit uPSI_IdNTLM; //Indy
36484: 584 unit uPSI_IdNNTP; //Indy
36485: 585 unit uPSI_usniffer; //PortScanForm //mX4
36486: 586 unit uPSI_IdCoderMIME; //Indy
36487: 587 unit uPSI_IdCoderUUE; //Indy
36488: 588 unit uPSI_IdCoderXXE; //Indy
36489: 589 unit uPSI_IdCoder3to4; //Indy
36490: 590 unit uPSI_IdCookie; //Indy

```

```

36491: 591 unit uPSI_IdCookieManager; //Indy
36492: 592 unit uPSI_WDosSocketUtils; //WDos
36493: 593 unit uPSI_WDosPlcUtils; //WDos
36494: 594 unit uPSI_WDosPorts; //WDos
36495: 595 unit uPSI_WDosResolvers; //WDos
36496: 596 unit uPSI_WDosTimers; //WDos
36497: 597 unit uPSI_WDosPlcs; //WDos
36498: 598 unit uPSI_WDosPneumatics; //WDos
36499: 599 unit uPSI_IdFingerServer; //Indy
36500: 600 unit uPSI_IdDNSResolver; //Indy
36501: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
36502: 602 unit uPSI_IdIntercept; //Indy
36503: 603 unit uPSI_IdIPMCastBase; //Indy
36504: 604 unit uPSI_IdLogBase; //Indy
36505: 605 unit uPSI_IdIOHandlerStream; //Indy
36506: 606 unit uPSI_IdMappedPortUDP; //Indy
36507: 607 unit uPSI_IdQOTDUDPServer; //Indy
36508: 608 unit uPSI_IdQOTDUDP; //Indy
36509: 609 unit uPSI_IdSysLog; //Indy
36510: 610 unit uPSI_IdSysLogServer; //Indy
36511: 611 unit uPSI_IdSysLogMessage; //Indy
36512: 612 unit uPSI_IdTimeServer; //Indy
36513: 613 unit uPSI_IdTimeUDP; //Indy
36514: 614 unit uPSI_IdTimeUDPServer; //Indy
36515: 615 unit uPSI_IdUserAccounts; //Indy
36516: 616 unit uPSI_TextUtils; //mX4
36517: 617 unit uPSI_MandelbrotEngine; //mX4
36518: 618 unit uPSI_delphi_arduino_Unit1; //mX4
36519: 619 unit uPSI_DTDSchema2; //mX4
36520: 620 unit uPSI_fplotMain; //DMath
36521: 621 unit uPSI_FindFileIter; //mX4
36522: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
36523: 623 unit uPSI_PppParser; //PPP
36524: 624 unit uPSI_PppLexer; //PPP
36525: 625 unit uPSI_PCharUtils; //PPP
36526: 626 unit uPSI_uJSON; //WU
36527: 627 unit uPSI_JclStrHashMap; //JCL
36528: 628 unit uPSI_JclHookExcept; //JCL
36529: 629 unit uPSI_EncdDecd; //VCL
36530: 630 unit uPSI_SockAppReg; //VCL
36531: 631 unit uPSI_PJFileHandle; //PJ
36532: 632 unit uPSI_PJEnvVars; //PJ
36533: 633 unit uPSI_PJPipe; //PJ
36534: 634 unit uPSI_PJPipeFilters; //PJ
36535: 635 unit uPSI_PJConsoleApp; //PJ
36536: 636 unit uPSI_UConsoleAppEx; //PJ
36537: 637 unit uPSI_DbxCsocketChannelNative; //VCL
36538: 638 unit uPSI_DbxCDataGenerator; //VCL
36539: 639 unit uPSI_DBXCClient; //VCL
36540: 640 unit uPSI_IdLogEvent; //Indy
36541: 641 unit uPSI_Reversi; //mX4
36542: 642 unit uPSI_Geometry; //mX4
36543: 643 unit uPSI_IdSMTPServer; //Indy
36544: 644 unit uPSI_Textures; //mX4
36545: 645 unit uPSI_IBX; //VCL
36546: 646 unit uPSI_IWDBCommon; //VCL
36547: 647 unit uPSI_SortGrid; //mX4
36548: 648 unit uPSI_IB; //VCL
36549: 649 unit uPSI_IBScript; //VCL
36550: 650 unit uPSI_JvCSVBaseControls; //JCL
36551: 651 unit uPSI_Jvg3DColors; //JCL
36552: 652 unit uPSI_JvHLEditor; //beat //JCL
36553: 653 unit uPSI_JvShellHook; //JCL
36554: 654 unit uPSI_DBCommon2; //VCL
36555: 655 unit uPSI_JvSHFileOperation; //JCL
36556: 656 unit uPSI_uFileexport; //mX4
36557: 657 unit uPSI_JvDialogs; //JCL
36558: 658 unit uPSI_JvDBTreeView; //JCL
36559: 659 unit uPSI_JvDBUltimGrid; //JCL
36560: 660 unit uPSI_JvDBQueryParamsForm; //JCL
36561: 661 unit uPSI_JvExControls; //JCL
36562: 662 unit uPSI_JvBDEMemTable; //JCL
36563: 663 unit uPSI_JvCommStatus; //JCL
36564: 664 unit uPSI_JvMailSlots2; //JCL
36565: 665 unit uPSI_JvgWinMask; //JCL
36566: 666 unit uPSI_StEclipse; //SysTools
36567: 667 unit uPSI_StMime; //SysTools
36568: 668 unit uPSI_StList; //SysTools
36569: 669 unit uPSI_StMerge; //SysTools
36570: 670 unit uPSI_StStrS; //SysTools
36571: 671 unit uPSI_StTree; //SysTools
36572: 672 unit uPSI_StVArr; //SysTools
36573: 673 unit uPSI_StRegIni; //SysTools
36574: 674 unit uPSI_urkf; //DMath
36575: 675 unit uPSI_usvd; //DMath
36576: 676 unit uPSI_DepWalkUtils; //JCL
36577: 677 unit uPSI_OptionsFrm; //JCL
36578: 678 unit yuvconverts; //mX4
36579: 679 uPSI_JvPropAutoSave; //JCL

```

```

36580: 680 uPSI_AclAPI; //alcinoe
36581: 681 uPSI_AviCap; //alcinoe
36582: 682 uPSI_ALAVLBinaryTree; //alcinoe
36583: 683 uPSI_ALFcnMisc; //alcinoe
36584: 684 uPSI_ALStringList; //alcinoe
36585: 685 uPSI_ALQuickSortList; //alcinoe
36586: 686 uPSI_ALStaticText; //alcinoe
36587: 687 uPSI_ALJSONDoc; //alcinoe
36588: 688 uPSI_ALGSMComm; //alcinoe
36589: 689 uPSI_ALWindows; //alcinoe
36590: 690 uPSI_ALMultiPartFormDataParser; //alcinoe
36591: 691 uPSI_ALHttpCommon; //alcinoe
36592: 692 uPSI_ALWebSpider; //alcinoe
36593: 693 uPSI_ALHttpClient; //alcinoe
36594: 694 uPSI_ALFcnHTML; //alcinoe
36595: 695 uPSI_ALFTPClient; //alcinoe
36596: 696 uPSI_ALInternetMessageCommon; //alcinoe
36597: 697 uPSI_ALWininetHttpClient; //alcinoe
36598: 698 uPSI_ALWininetFTPClient; //alcinoe
36599: 699 uPSI_ALWinHttpWrapper; //alcinoe
36600: 700 uPSI_ALWinHttpClient; //alcinoe
36601: 701 uPSI_ALFcnWinSock; //alcinoe
36602: 702 uPSI_ALFcnSQL; //alcinoe
36603: 703 uPSI_ALFcnCGI; //alcinoe
36604: 704 uPSI_ALFcnExecute; //alcinoe
36605: 705 uPSI_ALFcnFile; //alcinoe
36606: 706 uPSI_ALFcnMime; //alcinoe
36607: 707 uPSI_ALPhpRunner; //alcinoe
36608: 708 uPSI_ALGraphic; //alcinoe
36609: 709 uPSI_ALIniFiles; //alcinoe
36610: 710 uPSI_ALMemCachedClient; //alcinoe
36611: 711 unit uPSI_MyGrids; //mX4
36612: 712 uPSI_ALMultiPartMixedParser //alcinoe
36613: 713 uPSI_ALSMTPClient //alcinoe
36614: 714 uPSI_ALNNTPClient; //alcinoe
36615: 715 uPSI_ALHintBalloon; //alcinoe
36616: 716 unit uPSI_ALXmlDoc; //alcinoe
36617: 717 unit uPSI_IPCThrd; //VCL
36618: 718 unit uPSI_MonForm; //VCL
36619: 719 unit uPSI_TeCanvas; //Orpheus
36620: 720 unit uPSI_Ovcmisc; //Orpheus
36621: 721 unit uPSI_ovcfile; //Orpheus
36622: 722 unit uPSI_ovcstate; //Orpheus
36623: 723 unit uPSI_ovccoco; //Orpheus
36624: 724 unit uPSI_ovcrvexp; //Orpheus
36625: 725 unit uPSI_OvcFormatSettings; //Orpheus
36626: 726 unit uPSI_OvcUtils; //Orpheus
36627: 727 unit uPSI_ovcstore; //Orpheus
36628: 728 unit uPSI_ovcstr; //Orpheus
36629: 729 unit uPSI_ovcmru; //Orpheus
36630: 730 unit uPSI_ovccmd; //Orpheus
36631: 731 unit uPSI_ovctimer; //Orpheus
36632: 732 unit uPSI_ovcintl; //Orpheus
36633: 733 uPSI_AfCircularBuffer; //AsyncFree
36634: 734 uPSI_AfUtils; //AsyncFree
36635: 735 uPSI_AfSafeSync; //AsyncFree
36636: 736 uPSI_AfComPortCore; //AsyncFree
36637: 737 uPSI_AfComPort; //AsyncFree
36638: 738 uPSI_AfPortControls; //AsyncFree
36639: 739 uPSI_AfDataDispatcher; //AsyncFree
36640: 740 uPSI_AfViewers; //AsyncFree
36641: 741 uPSI_AfDataTerminal; //AsyncFree
36642: 742 uPSI_SimplePortMain; //AsyncFree
36643: 743 unit uPSI_ovcclock; //Orpheus
36644: 744 unit uPSI_o32intlst; //Orpheus
36645: 745 unit uPSI_o32ledlabel; //Orpheus
36646: 746 unit uPSI_ALMySqlClient; //alcinoe
36647: 747 unit uPSI_ALFBXClient; //alcinoe
36648: 748 unit uPSI_ALFcnSQL; //alcinoe
36649: 749 unit uPSI_AsyncTimer; //mX4
36650: 750 unit uPSI_ApplicationFileIO; //mX4
36651: 751 unit uPSI_PsAPI; //VCLé
36652: 752 uPSI_ovcuser; //Orpheus
36653: 753 uPSI_ovcurl; //Orpheus
36654: 754 uPSI_ovcvlb; //Orpheus
36655: 755 uPSI_ovccolor; //Orpheus
36656: 756 uPSI_ALFBXLib; //alcinoe
36657: 757 uPSI_ovcmeter; //Orpheus
36658: 758 uPSI_ovcpeakm; //Orpheus
36659: 759 uPSI_O32BGSty; //Orpheus
36660: 760 uPSI_ovcBidi; //Orpheus
36661: 761 uPSI_ovctcary; //Orpheus
36662: 762 uPSI_DXPUtils; //mX4
36663: 763 uPSI_ALMultiPartBaseParser; //alcinoe
36664: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe
36665: 765 uPSI_ALPOP3Client; //alcinoe
36666: 766 uPSI_SmallUtils; //mX4
36667: 767 uPSI_MakeApp; //mX4
36668: 768 uPSI_O32MouseMon; //Orpheus

```

```

36669: 769 uPSI_OvcCache; //Orpheus
36670: 770 uPSI_ovccalc; //Orpheus
36671: 771 uPSI_Joystick; //OpenGL
36672: 772 uPSI_ScreenSaver; //OpenGL
36673: 773 uPSI_XCollection; //OpenGL
36674: 774 uPSI_Polynomials; //OpenGL
36675: 775 uPSI_PersistentClasses, //9.86 //OpenGL
36676: 776 uPSI_VectorLists; //OpenGL
36677: 777 uPSI_XOpenGL; //OpenGL
36678: 778 uPSI_MeshUtils; //OpenGL
36679: 779 unit uPSI_JclSysUtils; //JCL
36680: 780 unit uPSI_JclBorlandTools; //JCL
36681: 781 unit JclFileUtils_max; //JCL
36682: 782 uPSI_AfDataControls; //AsyncFree
36683: 783 uPSI_GLSilhouette; //OpenGL
36684: 784 uPSI_JclSysUtils_class; //JCL
36685: 785 uPSI_JclFileUtils_class; //JCL
36686: 786 uPSI_FileUtil; //JCL
36687: 787 uPSI_changefind; //mX4
36688: 788 uPSI_cmdIntf; //mX4
36689: 789 uPSI_fservice; //mX4
36690: 790 uPSI_Keyboard; //OpenGL
36691: 791 uPSI_VRMLParser; //OpenGL
36692: 792 uPSI_GLFileVRML; //OpenGL
36693: 793 uPSI_Octree; //OpenGL
36694: 794 uPSI_GLPolyhedron; //OpenGL
36695: 795 uPSI_GLCrossPlatform; //OpenGL
36696: 796 uPSI_GLParticles; //OpenGL
36697: 797 uPSI_GLNavigator; //OpenGL
36698: 798 uPSI_GLStarRecord; //OpenGL
36699: 799 uPSI_GLTextureCombiners; //OpenGL
36700: 800 uPSI_GLCanvas; //OpenGL
36701: 801 uPSI_GeometryBB; //OpenGL
36702: 802 uPSI_GeometryCoordinates; //OpenGL
36703: 803 uPSI_VectorGeometry; //OpenGL
36704: 804 uPSI_BumpMapping; //OpenGL
36705: 805 uPSI_TGA; //OpenGL
36706: 806 uPSI_GLVectorFileObjects; //OpenGL
36707: 807 uPSI_IMM; //VCL
36708: 808 uPSI_CategoryButtons; //VCL
36709: 809 uPSI_ButtonGroup; //VCL
36710: 810 uPSI_DbExcept; //VCL
36711: 811 uPSI_AxCtrls; //VCL
36712: 812 uPSI_GL_actorUnit1; //OpenGL
36713: 813 uPSI_StdVCL; //VCL
36714: 814 unit CurvesAndSurfaces; //OpenGL
36715: 815 uPSI_DataAwareMain; //AsyncFree
36716: 816 uPSI_TabNotBk; //VCL
36717: 817 uPSI_udwsfiler; //mX4
36718: 818 uPSI_synaip; //Synapse!
36719: 819 uPSI_synacode; //Synapse
36720: 820 uPSI_synachar; //Synapse
36721: 821 uPSI_synamisc; //Synapse
36722: 822 uPSI_synaser; //Synapse
36723: 823 uPSI_synaicnv; //Synapse
36724: 824 uPSI_tlntsend; //Synapse
36725: 825 uPSI_pingsend; //Synapse
36726: 826 uPSI_blksock; //Synapse
36727: 827 uPSI_asn1util; //Synapse
36728: 828 uPSI_dnssend; //Synapse
36729: 829 uPSI_clamsend; //Synapse
36730: 830 uPSI_ldapsend; //Synapse
36731: 831 uPSI_mimemess; //Synapse
36732: 832 uPSI_slogsend; //Synapse
36733: 833 uPSI_mimepart; //Synapse
36734: 834 uPSI_mimeinln; //Synapse
36735: 835 uPSI_ftpsend; //Synapse
36736: 836 uPSI_ftptsend; //Synapse
36737: 837 uPSI_httpsend; //Synapse
36738: 838 uPSI_sntpsend; //Synapse
36739: 839 uPSI_smtpsend; //Synapse
36740: 840 uPSI_snmpsend; //Synapse
36741: 841 uPSI_imapsend; //Synapse
36742: 842 uPSI_pop3send; //Synapse
36743: 843 uPSI_nnptsend; //Synapse
36744: 844 uPSI_ssl_cryptlib; //Synapse
36745: 845 uPSI_ssl_openssl; //Synapse
36746: 846 uPSI_synhttp_daemon; //Synapse
36747: 847 uPSI_NetWork; //mX4
36748: 848 uPSI_PingThread; //Synapse
36749: 849 uPSI_JvThreadTimer; //JCL
36750: 850 unit uPSI_wwSystem; //InfoPower
36751: 851 unit uPSI_IdComponent; //Indy
36752: 852 unit uPSI_IdIOHandlerThrottle; //Indy
36753: 853 unit uPSI_Themes; //VCL
36754: 854 unit uPSI_StdStyleActnCtrls; //VCL
36755: 855 unit uPSI_UDDIHelper; //VCL
36756: 856 unit uPSI_IdIMAP4Server; //Indy
36757: 857 uPSI_VariantSymbolTable; //VCL //3.9.9.92

```



```

36758: 858 uPSI_udf_glob, //mX4
36759: 859 uPSI_TabGrid, //VCL
36760: 860 uPSI_JsDBTreeView, //mX4
36761: 861 uPSI_JsSendMail, //mX4
36762: 862 uPSI_dbTvRecordList, //mX4
36763: 863 uPSI_TreeVwEx, //mX4
36764: 864 uPSI_ECDataLink, //mX4
36765: 865 uPSI_dbTree, //mX4
36766: 866 uPSI_dbTreeCBox, //mX4
36767: 867 unit uPSI_Debug; //TfrmDebug //mX4
36768: 868 uPSI_TimeFnCs; //mX4
36769: 869 uPSI_FileIntf, //VCL
36770: 870 uPSI_SockTransport, //RTL
36771: 871 unit uPSI_WinInet; //RTL
36772: 872 unit uPSI_Wwstr; //mX4
36773: 873 uPSI_DBLookup, //VCL
36774: 874 uPSI_Hotspot, //mX4
36775: 875 uPSI_HList; //History List //mX4
36776: 876 unit uPSI_DrTable; //VCL
36777: 877 uPSI_TConnect, //VCL
36778: 878 uPSI_DataBkr, //VCL
36779: 879 uPSI_HTTPIntr; //VCL
36780: 880 unit uPSI_Mathbox; //mX4
36781: 881 uPSI_cyIndy, //cY
36782: 882 uPSI_cySysUtils, //cY
36783: 883 uPSI_cyWinUtils, //cY
36784: 884 uPSI_cyStrUtils, //cY
36785: 885 uPSI_cyObjUtils, //cY
36786: 886 uPSI_cyDateUtils, //cY
36787: 887 uPSI_cyBDE, //cY
36788: 888 uPSI_cyClasses, //cY
36789: 889 uPSI_cyGraphics, //3.9.9.94_2 //cY
36790: 890 unit uPSI_cyTypes; //cY
36791: 891 uPSI_JvDateTimePicker, //JCL
36792: 892 uPSI_JvCreateProcess, //JCL
36793: 893 uPSI_JvEasterEgg, //JCL
36794: 894 uPSI_WinSvc, //3.9.9.94_3 //VCL
36795: 895 uPSI_SvcMgr, //VCL
36796: 896 unit uPSI_JvPickDate; //JCL
36797: 897 unit uPSI_JvNotify; //JCL
36798: 898 uPSI_JvStrHlder, //JCL
36799: 899 unit uPSI_JclNTFS2; //JCL
36800: 900 uPSI_Jcl8087 //math coprocessor //JCL
36801: 901 uPSI_JvAddPrinter, //JCL
36802: 902 uPSI_JvCabFile, //JCL
36803: 903 uPSI_JvDataEmbedded; //JCL
36804: 904 unit uPSI_U_HexView; //mX4
36805: 905 uPSI_UWavein4, //mX4
36806: 906 uPSI_AMixer, //mX4
36807: 907 uPSI_JvaScrollText, //mX4
36808: 908 uPSI_JvArrow, //mX4
36809: 909 unit uPSI_UrlMon; //mX4
36810: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
36811: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain; //DFF
36812: 912 unit uPSI_DFFUtils; //DFF
36813: 913 unit uPSI_MathsLib; //DFF
36814: 914 uPSI_UIntList; //DFF
36815: 915 uPSI_UGetParens; //DFF
36816: 916 unit uPSI_UGeometry; //DFF
36817: 917 unit uPSI_UAstronomy; //DFF
36818: 918 unit uPSI_UCardComponentV2; //DFF
36819: 919 unit uPSI_UTGraphSearch; //DFF
36820: 920 unit uPSI_UParser10; //DFF
36821: 921 unit uPSI_cyIEUtils; //cY
36822: 922 unit uPSI_UcomboV2; //DFF
36823: 923 uPSI_cyBaseComm, //cY
36824: 924 uPSI_cyAppInstances, //cY
36825: 925 uPSI_cyAttract, //cY
36826: 926 uPSI_cyDERUtils, //cY
36827: 927 unit uPSI_cyDocER; //cY
36828: 928 unit uPSI_ODBC; //mX
36829: 929 unit uPSI_AssocExec; //mX
36830: 930 uPSI_cyBaseCommRoomConnector, //cY
36831: 931 uPSI_cyCommRoomConnector, //cY
36832: 932 uPSI_cyCommunicate, //cY
36833: 933 uPSI_cyImage; //cY
36834: 934 uPSI_cyBaseContainer, //cY
36835: 935 uPSI_cyModalContainer, //cY
36836: 936 uPSI_cyFlyingContainer; //cY
36837: 937 uPSI_RegStr, //VCL
36838: 938 uPSI_HtmlHelpViewer; //VCL
36839: 939 unit uPSI_cyIniForm, //cY
36840: 940 unit uPSI_cyVirtualGrid; //cY
36841: 941 uPSI_Profiler, //DA
36842: 942 uPSI_BackgroundWorker, //DA
36843: 943 uPSI_WavePlay, //DA
36844: 944 uPSI_WaveTimer, //DA
36845: 945 uPSI_WaveUtils; //DA
36846: 946 uPSI_NamedPipes, //TB

```

```

36847: 947 uPSI_NamedPipeServer, //TB
36848: 948 unit uPSI_process, //TB
36849: 949 unit uPSI_DPUtils, //TB
36850: 950 unit uPSI_CommonTools, //TB
36851: 951 uPSI_DataSendToWeb, //TB
36852: 952 uPSI_StarCalc, //TB
36853: 953 uPSI_D2_XPVistaHelperU //TB
36854: 954 unit uPSI_NetTools //TB
36855: 955 unit uPSI_Pipes //TB
36856: 956 uPSI_ProcessUnit, //mX
36857: 957 uPSI_adGSM, //mX
36858: 958 unit uPSI_BetterADODataset, //mX
36859: 959 unit uPSI_AdSelCom; //FTT //mX
36860: 960 unit uPSI_dwsXPlatform; //DWS
36861: 961 uPSI_AdSocket; //mX Turbo Power
36862: 962 uPSI_AdPacket; //mX
36863: 963 uPSI_AdPort; //mX
36864: 964 uPSI_PathFunc; //Inno
36865: 965 uPSI_CmnFunc; //Inno
36866: 966 uPSI_CmnFunc2; //Inno Setup //Inno
36867: 967 unit uPSI_BitmapImage; //mX4
36868: 968 unit uPSI_ImageGrabber; //mX4
36869: 969 uPSI_SecurityFunc, //Inno
36870: 970 uPSI_RedirFunc, //Inno
36871: 971 uPSI_FIFO, (MemoryStream) //mX4
36872: 972 uPSI_Int64Em, //Inno
36873: 973 unit uPSI_InstFunc; //Inno
36874: 974 unit uPSI_LibFusion; //Inno
36875: 975 uPSI_SimpleExpression; //Inno
36876: 976 uPSI_unitResourceDetails, //XN
36877: 977 uPSI_unitResFile, //XN
36878: 978 unit uPSI_simpleComport; //mX4
36879: 979 unit uPSI_AfViewershelfers; //Async
36880: 980 unit uPSI_Console; //mX4
36881: 981 unit uPSI_AnalogMeter; //TB
36882: 982 unit uPSI_XPrinter, //TB
36883: 983 unit uPSI_IniFiles; //VCL
36884: 984 unit uPSI_lazIniFiles; //FP
36885: 985 uPSI_testutils; //FP
36886: 986 uPSI_ToolsUnit; (DBTests) //FP
36887: 987 uPSI_fpcunit //FP
36888: 988 uPSI_testdecorator; //FP
36889: 989 unit uPSI_fpcunittests; //FP
36890: 990 unit uPSI_cTCPBuffer; //Fundamentals 4
36891: 991 unit uPSI_Glut, //Open GL
36892: 992 uPSI_LEDBitmaps, //mX4
36893: 993 uPSI_FileClass, //Inno
36894: 994 uPSI_FileUtilsClass, //mX4
36895: 995 uPSI_ComPortInterface; //Kit //mX4
36896: 996 unit uPSI_SwitchLed; //mX4
36897: 997 unit uPSI_cyDmmCanvas; //cY
36898: 998 uPSI_uColorFunctions; //DFF
36899: 999 uPSI_uSettings; //DFF
36900: 1000 uPSI_cyDebug.pas //cY
36901: 1001 uPSI_cyColorMatrix; //cY
36902: 1002 unit uPSI_cyCopyFiles; //cY
36903: 1003 unit uPSI_cySearchFiles; //cY
36904: 1004 unit uPSI_cyBaseMeasure; //cY
36905: 1005 unit uPSI_PJISStreams; //DD
36906: 1006 unit uPSI_cyRunTimeResize; //cY
36907: 1007 unit uPSI_jcontrolutils; //cY
36908: 1008 unit uPSI_kcMapView; (+GEONames) //kc
36909: 1009 unit uPSI_kcMapViewDESynapse; //kc
36910: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
36911: 1011 unit uPSI_LedNumber; //TurboPower
36912: 1012 unit uPSI_StStrL; //SysTools
36913: 1013 unit uPSI_indGnouMeter; //LAZ
36914: 1014 unit uPSI_Sensors; //LAZ
36915: 1015 unit uPSI_pwmain; //cgi of powtils //Pow
36916: 1016 unit uPSI_HTMLUtil; //Pow
36917: 1017 unit uPSI_synwrap1; //httpsend //Pow
36918: 1018 unit StreamWrap1 //Pow
36919: 1019 unit uPSI_pwmain; //Pow
36920: 1020 unit pwtypes //Pow
36921: 1021 uPSI_W32VersionInfo //LAZ
36922: 1022 unit uPSI_IpAnim; //LAZ
36923: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
36924: 1024 unit uPSI_LrtPoTools; //LAZ
36925: 1025 unit uPSI_Laz_DOM; //LAZ
36926: 1026 unit uPSI_hhAvComp; //LAZ
36927: 1027 unit uPSI_GPS2; //mX4
36928: 1028 unit uPSI_GPS; //mX4
36929: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo //mX4
36930: 1030 unit uPSI_NMEA; // GPS //mX4
36931: 1031 unit uPSI_ScreenThreeDLab; //mX4
36932: 1032 unit uPSI_Spin; //VCL
36933: 1033 unit uPSI_DynaZip; //mX4
36934: 1034 unit uPSI_clockExpert; //TB
36935: 1035 unit debugLn //mX4

```

```

36936: 1036 uPSI_SortUtils; //Jcl
36937: 1037 uPSI_BitmapConversion; //Jcl
36938: 1038 unit uPSI_JclTD32; //Jcl
36939: 1039 unit uPSI_ZDbcUtils; //Zeos
36940: 1040 unit uPSI_ZScriptParser; //Zeos
36941: 1041 uPSI_JvIni; //JCL
36942: 1042 uPSI_JvFtpGrabber; //JCL
36943: 1043 unit uPSI_NeuralNetwork; //OCL
36944: 1044 unit uPSI_StExpr; //SysTools
36945: 1045 unit uPSI_GR32_Geometry; //GR32
36946: 1046 unit uPSI_GR32_Containers; //GR32
36947: 1047 unit uPSI_GR32_Backends_VCL; //GR32
36948: 1048 unit uPSI_StSaturn; //Venus+Mercury+Mars++ //SysTools
36949: 1049 unit uPSI_JclParseUses; //JCL
36950: 1050 unit uPSI_JvFinalize; //JCL
36951: 1051 unit uPSI_panUnit1; //GLScene
36952: 1052 unit uPSI_DD83ul; //Arduino Tester //mX4
36953: 1053 unit uPSI_BigIni; //Hinzen
36954: 1054 unit uPSI_ShellCtrls; //VCL
36955: 1055 unit uPSI_fmMath; //FMATH
36956: 1056 unit uPSI_fComp; //FMATH
36957: 1057 unit uPSI_HighResTimer; //Lauer
36958: 1058 unit uconvMain; (Unit Converter) //PS
36959: 1059 unit uPSI_uconvMain; //PS
36960: 1060 unit uPSI_ParserUtils; //PS
36961: 1061 unit uPSI_uPSUtils; //PS
36962: 1062 unit uPSI_ParserU; //PS
36963: 1063 unit uPSI_TypInfo; //VCL
36964: 1064 unit uPSI_ServiceMgr; //mX
36965: 1065 unit uPSI_UDict; //DFF
36966: 1066 unit uPSI_ubigFloatV3; //DFF
36967: 1067 unit uPSI_UBigIntsV4; //DFF
36968: 1068 unit uPSI_ServiceMgr2; //mX
36969: 1069 unit uPSI_UP10Build; //PS
36970: 1070 unit uPSI_UParser10; //PS
36971: 1071 unit uPSI_IdModBusServer; //MB
36972: 1072 unit uPSI_IdModBusClient; //MB
36973: 1073 unit uPSI_ColorGrd; //VCL
36974: 1074 unit uPSI_DirOutln; //VCL
36975: 1075 unit uPSI_Gauges; //VCL
36976: 1076 unit uPSI_Customizedlg; //VCL
36977: 1077 unit uPSI_ActnMan; //VCL
36978: 1078 unit uPSI_CollPanl; //VCL
36979: 1079 unit uPSI_Calendar2; //VCL
36980: 1080 unit uPSI_IBCtrls; //VCL
36981: 1081 unit uPSI_IdStackWindows; //Indy
36982: 1082 unit uPSI_CTSVendorUtils; //DBX
36983: 1083 unit uPSI_VendorTestFramework; //DBX
36984: 1084 unit uPSI_TInterval; //mX4
36985: 1085 unit uPSI_JvAnimate; //JCL
36986: 1086 unit uPSI_DBXCharDecoder; //DBX
36987: 1087 unit uPSI_JvDBLists; //JCL
36988: 1088 unit uPSI_JvFileInfo; //JCL
36989: 1089 unit uPSI_SOAPConn; //VCL
36990: 1090 unit uPSI_SOAPLinked; //VCL
36991: 1091 unit uPSI_XSBuiltIns; //VCL
36992: 1092 unit uPSI_JvgDigits; //JCL
36993: 1093 unit uPSI_JvDesignUtils;
36994: 1094 unit uPSI_JvgCrossTable;
36995: 1095 unit uPSI_JvgReport;
36996: 1096 unit uPSI_JvDBRichEdit;
36997: 1097 unit uPSI_JvWinHelp;
36998: 1098 unit uPSI_WaveConverter;
36999: 1099 unit uPSI_ACMConvertor;
37000: 1100 unit XSBuiltIns_Routines
37001: 1101 unit uPSI_ComObjOleDB_utils.pas
37002: 1102 unit uPSI_SMScript;
37003: 1103 unit uPSI_CompFileIo;
37004: 1104 unit uPSI_SynHighlighterGeneral;
37005: 1105 unit uPSI_geometry2;
37006: 1106 unit uPSI_MConnect;
37007: 1107 unit uPSI_ObjBrkr;
37008: 1108 unit uPSI_uMultiStr;
37009: 1109 unit uPSI_WinAPI.pas;
37010: 1110 unit uPSI_JvAVICapture;
37011: 1111 unit uPSI_JvExceptionForm;
37012: 1112 unit uPSI_JvConnectNetwork;
37013: 1113 unit uPSI_MTMainForm;
37014: 1114 unit uPSI_DdeMan;
37015: 1115 unit uPSI_DIUtils;
37016: 1116 unit uPSI_gnugettext;
37017: 1117 unit uPSI_Xmlxform;
37018: 1118 unit uPSI_SvrHTTPIndy;
37019: 1119 unit uPSI_CPortTrmSet;
37020: 1120 unit SvrLog; -----
37021: 1121 unit uPSI_IndySockTransport.pas (+IdHTTPHeaderInfo) //mX4
37022: 1122 unit uPSI_HTTPProd.pas
37023: 1123 unit uPSI_CppParser.pas
37024: 1124 unit uPSI_SynHighlighterCpp.pas

```

```
37025: 1125 unit uPSI_CodeCompletion.pas
37026: 1126 unit uPSI_U_IntList2.pas
37027: 1127 unit uPSI_SockHTTP.pas
37028: 1128 uPSI_SockAppNotify.pas
37029: 1129 uPSI_NSToIS.pas
37030: 1130 unit uPSI_DBOleCtl.pas
37031: 1131 unit uPSI_xercesxmldom;
37032: 1132 unit uPSI_xmldom;
37033: 1133 unit uPSI_Midas;
37034: 1134 unit uPSI_JclExprEval;
37035: 1135 uPSI_Gameboard;
37036: 1136 unit uPSI_ExtUtil;
37037: 1137 unit uPSI_FCGIApp;
37038: 1138 unit uPSI_ExtPascal;
37039: 1139 unit uPSI_PersistSettings;
37040: 1140 IdHTTPHeaderInfo.pas
37041: 1141 uPSI_SynEditAutoComplete;
37042: 1142 uPSI_SynEditTextBuffer.pas
37043: 1143 unit uPSI_JclPCRE;
37044: 1144 unit uPSI_ZConnection;
37045: 1145 unit uPSI_ZSequence;
37046: 1146 unit uPSI_ChessPrg;
37047: 1147 unit uPSI_ChessBrd;
37048: 1148 unit uPSI_Graph3D;
37049: 1149 uPSI_SysInfoCtrls2.pas
37050: 1150 unit uPSI_RegUtils;
37051: 1151 unit uPSI_VariantRtn;
37052: 1152 uPSI_StdFuncs;
37053: 1153 unit uPSI_SqlTxtRtns;
37054: 1154 unit uPSI_BSpectrum;
37055: 1155 unit IPAddressControl;
37056: 1156 unit uPSI_Paradox;
37057: 1157 unit uPSI_Environ;
37058: 1158 uPSI_GraphicsPrimitivesLibrary;
37059: 1159 uPSI_DrawFigures;
37060: 1160 unit uPSI_synabg;
37061: 1161 unit uPSI_BitStream;
37062: 1162 unit uPSI_xrtl_util_FileVersion;
37063: 1163 uPSI_XmlRpcCommon;
37064: 1164 unit uPSI_XmlRpcClient;
37065: 1165 unit uPSI_XmlRpcTypes;
37066: 1166 unit uPSI_XmlRpcServer;
37067: 1167 unit uPSI_SynAutoIndent;
37068: 1168 unit uPSI_synafpc;
37069: 1169 unit uPSI_RxNotify;
37070: 1170 unit uPSI_SynAutoCorrect;
37071: 1171 unit uPSI_rxOle2Auto;
37072: 1172 unit uPSI_Spring_Utilsmx;
37073: 1173 unit uPSI_ulogfileit;
37074: 1174 unit uPSI_HarmFade;
37075: 1175 unit uPSI_SynCompletionProposal;
37076: 1176 unit uPSI_rxAniFile;
37077: 1177 uPSI_ulinfitt;
37078: 1178 uPSI_usvdfitt;
37079: 1179 unit uPSI_JclStringLists;
37080: 1180 unit uPSI_ZLib;
37081: 1181 unit uPSI_MaxTokenizers; //WANT
37082: 1182 unit uPSI_MaxStrUtils;
37083: 1183 unit uPSI_MaxXMLUtils;
37084: 1184 unit uPSI_MaxUtils;
37085: 1185 unit uPSI_VListBox;
37086: 1186 unit uPSI_MaxDOM;
37087: 1187 unit uPSI_MaxDOMDictionary;
37088: 1188 unit uPSI_MaxDOMDictionary_Routines;
37089: 1189 unit uPSI_cASN1;
37090: 1190 unit uPSI_cX509Certificate;
37091: 1191 unit uPSI_uCiaXml;
37092: 1192 unit uPSI_StringsW;
37093: 1193 unit uPSI_FileStreamW; //WideString D7X
37094: 1194 unit Drawingutils;
37095: 1195 unit uPSI_InetUtilsUnified;
37096: 1196 unit uPSI_FileMask;
37097: 1197 unit uPSI_StrConv;
37098: 1198 unit uPSI_Simpat;
37099: 1199 unit uPSI_Tooltips.pas
37100: 1200 unit uPSI_StringGridLibrary.pas
37101: 1201 unit uPSI_ChronCheck.pas
37102: 1202 unit uPSI_REXX.pas
37103: 1203 uPSI_SysImg.pas
37104: 1204 unit uPSI_Tokens;
37105: 1205 unit uPSI_KFunctions;
37106: 1206 unit uPSI_KMessageBox;
37107: 1207 unit uPSI_CPUSpeed.pas
37108: 1208 uPSI_RoboTracker.pas
37109: 1209 unit uPSI_NamedPipesImpl.pas
37110: 1210 unit uPSI_KLog.pas
37111: 1211 unit uPSI_NamedPipeThreads.pas
37112: 1212 unit uPSI_MapFiles.pas //map stream of memory-mapped files
37113: 1213 unit uPSI_BKPwdGen, //Password Generator
```

```
37114: 1214 unit uPSI_Kronos, // big chrono date time library
37115: 1215 unit uPSI_TokenLibrary2;
37116: 1216 uPSI_KDialogs,
37117: 1217 uPSI_Numedit,
37118: 1218 unit uPSI_StSystem2;
37119: 1219 unit uPSI_KGraphics;
37120: 1220 uPSI_KGraphics_functions;
37121: 1221 uPSI_umaxPipes.pas
37122: 1222 unit uPSI_KControls;
37123: 1223 unit SysUtils_max2;
37124: 1224 uPSI_IdAntiFreeze.pas
37125: 1225 uPSI_IdLogStream.pas
37126: 1226 unit uPSI_IdThreadSafe;
37127: 1227 unit uPSI_IdThreadMgr;
37128: 1228 unit uPSI_IdAuthentication;
37129: 1229 unit uPSI_IdAuthenticationManager;
37130: 1230 uPSI_OverbyteIcsConApp
37131: 1231 unit uPSI_KMemo;
37132: 1232 unit uPSI_OverbyteIcsTicks64;
37133: 1233 unit uPSI_OverbyteIcsShal.pas
37134: 1234 unit uPSI_KEditCommon.pas
37135: 1235 unit uPSI_UtillsMax4.pas
37136: 1236 unit uPSI_IdNNTPServer;
37137: 1237 unit uPSI_UWANTUtils;
37138: 1238 unit uPSI_UtillsMax5.pas;
37139: 1239 unit uPSI_OverbyteIcsAsn1Utils;
37140: 1240 unit uPSI_IdHTTPHeaderInfo; //mX4 response headers
37141: 1241 uPSI_wmiserv.pas {uPSI_SimpleSFTP.pas}
37142: 1242 uPSI_WbemScripting_TLB.pas
37143: 1243 unit uPSI_uJSON;
37144: 1244 uPSI_RegSvrUtils.pas
37145: 1245 unit uPSI_osFileUtil;
37146: 1246 unit uPSI_SHDocVw; //TWebbrowser
37147: 1247 unit uPSI_SHDocVw_TLB
37148: 1248 uPSC_classes.pas V2
37149: 1249 uPSR_classes.pas V2
37150: 1250 uPSI_U_Oscilloscope4_2
37151: 1251 uPSI_xutils.pas
37152: 1252 uPSI_ietf.pas
37153: 1253 uPSI_iso3166.pas
37154: 1254 uPSI_dateutil_real.pas //Optima ISO 8601
37155: 1255 uPSI_dateext4.pas
37156: 1256 uPSI_locale.pas //ISO Formater & Tests
37157: 1257 file charset.inc //IANA Registered character sets
37158: 1258 unit uPSI_Strings;
37159: 1259 unit uPSI_crc_checks; //ISO 3309 and ITU-T-V42
37160: 1260 unit uPSI_extDOS;
37161: 1261 uPSI_uBild; //Steganography
37162: 1262 unit uPSI_SimpleTCP;
37163: 1263 unit uPSI_IdFTPList;
37164: 1264 uPSI_uTPLb_CryptographicLibrary.pas
37165: 1265 uPSI_uTPLb_RSA_Engine.pas
37166: 1266 uPSI_cHugeInt.pas
37167: 1267 unit uPSI_SimpleTCPServer;
37168: 1268 unit uPSI_xBase.pas
37169: 1269 unit uPSI_ImageHistogram.pas
37170: 1270 unit uP_PersistSettings2;
37171: 1271 uPSI_WDosDrivers.pas
37172: 1272 unit uPSI_cCipherRSA;
37173: 1273 uPSI_CromisStreams, (TStreamStorage)
37174: 1274 unit uPSI_Streams,
37175: 1275 uPSI_BitStream,
37176: 1276 uPSI_UJSONFunctions.pas
37177: 1277 uPSI_uTPLb_BinaryUtils.pas
37178: 1278 unit uPSI_USha256.pas //PascalCoin Blockchain
37179: 1279 uPSI_uTPLb_HashDsc.pas
37180: 1280 uPSI_uTPLb_Hash.pas
37181: 1281 SIRegister_Series(X); (uPSI_Series) //4.2.6.10
37182: 1282 unit uPSI_UTime; (UTime); uPSI_mimeinln2;
37183: 1283 uPSI_uTPLb_StreamCipher.pas
37184: 1284 uPSI_uTPLb_BlockCipher.pas
37185: 1285 uPSI_uTPLb_Asymetric.pas
37186: 1286 uPSI_uTPLb_CodecIntf.pas
37187: 1287 uPSI_uTPLb_Codec.pas
37188: 1288 uPSI_ADOInt.pas
37189: 1289 uPSI_MidasCon.pas
37190: 1290 uPSI_XMLDoc.pas
37191: 1291 uPSI_XMLIntf.pas
37192: 1292 uPSI_ProxyUtils.pas
37193: 1293 unit uPSI_maxXMLUtils2;
37194: 1294 unit_StDict_Routines(S: TPSExec);
37195: 1295 unit uPSI_Hashes2
37196: 1296 unit uPSI_IdCoderHeader;
37197: 1297 unit uPSI_BackgroundWorker2
37198: 1298 uPSI_uMRU.pas
37199: 1299 unit FANN.pas
37200: 1300 unit uPSI_FannNetwork.pas
37201: 1301 unit uPSI_RTLDateTimeplus.pas
37202: 1302 uPSI_ULog.pas
```



```

37203: 1303 uPSI_UThread.pas
37204: 1304 uPSI_UTCPIP.pas
37205: 1305 Synapse_OpenSSLv11
37206: 1306 PascalCoin configuration config.inc
37207: 1307 unit uPSI_statmach, {StateMachine} //46310
37208: 1308 uPSI_uTPLb_RSA_Primitives,
37209: 1309 unit uPSI_UMatrix,
37210: 1310 uPSI_DXUtil,
37211: 1311 uPSI_crlfParser,
37212: 1312 unit uPSI_DCPbase64;
37213: 1313 unit uPSI_FlyFilesUtils;
37214: 1314 uPSI_PJConsoleApp.pas
37215: 1315 uPSI_PJStreamWrapper.pas
37216: 1316 uPSI_LatLonDist, //DFF
37217: 1317 uPSI_cHash2.pas //Fundamentals
37218: 1318 uPSI_ZLib2.pas
37219: 1319 unit uPSI_commDriver
37220: 1320 unit uPSI_PXLNetComs.pas //PXL
37221: 1321 unit uPSI_PXLTiming.pas //PXL
37222: 1322 uPSI_Odometer.pas
37223: 1323 unit uPSI_UIntList2;
37224: 1324 uPSI_UIntegerpartition.pas
37225: 1325 unit uPSI_idPHPRunner.pas //prepare for PHP4D
37226: 1326 unit uPSI_idCGIRunner.pas
37227: 1327 uPSI_DrBobCGI, //4.7.1.20
37228: 1328 uPSI_OverbyteIcsLogger, / / n e w c o u n t 1 2 2 8 - - > 1 3 2 8
37229: 1329 uPSI_OverbyteIcsNntpCli, testset
37230: 1330 uPSI_OverbyteIcsCharsetUtils,
37231: 1331 uPSI_OverbyteIcsMimeUtils,
37232: 1332 uPSI_OverbyteIcsUrl(CL: TPSPascalCompiler);
37233: 1333 uPSI_uWebSocket.pas
37234: 1334 uPSI_KhFunction.pas
37235: 1335 uPSI_ALOpenOffice.pas
37236: 1336 unit uPSI_ALLibPhoneNumber
37237: 1337 unit uPSI_ALPhpRunner2;
37238: 1338 unit uPSI_ALWebSpider2;
37239: 1339 unit uPSI_ALFcnHTML2; // RunJavaScript2
37240: 1340 unit uPSI_ALExecute2.pas
37241: 1341 uPSI_ALIsapiHTTP.pas
37242: 1342 uPSI_ALOpenOffice_Routines
37243: 1343 unit uPSI_uUsb;
37244: 1344 uPSI_uWebcam.pas
37245: 1345 uPSI_PersistSettings.pas //fixing & refactoring
37246: 1346 uPSI_uTPLb_MemoryStreamPool.pas
37247: 1347 uPSI_uTPLb_Signatory.pas
37248: 1348 unit uPSI_uTPLb_Constants.pas //TurboPower
37249: 1349 uPSI_uTPLb_Random.pas
37250: 1350 unit uPSI_uTPLb_PointerArithmetic;
37251: 1351 unit uPSI_EwbCoreTools.pas
37252: 1352 unit uPSI_EwbUrl.pas
37253: 1353 unit uPSI_SendMail_For_Ewb.pas
37254: 1354 unit uPSI_MaskEdit.pas FLC
37255: 1355 unit uPSI_SimpleRSSTypes; BlueHippo
37256: 1356 unit uPSI_SimpleRSS; BlueHippo
37257: 1357 unit uPSI_psULib.pas Prometheus
37258: 1358 unit uPSI_psUFinancial; Prometheus
37259: 1359 uPSI_PsAPI_2.pas mX4
37260: 1360 uPSI_PersistSettings_2 mX4
37261: 1361 uPSI_rfc1213util2.pas IP
37262: 1362 uPSI_JTools.pas JCL
37263: 1363 unit uPSI_neuralbit.pas CAI
37264: 1364 unit uPSI_neuralab.pas CAI
37265: 1365 unit uPSI_winsvc2.pas TEK
37266: 1366 unit uPSI_wmiserv2.pas TEK
37267: 1367 uPSI_neuralcache.pas CAI
37268: 1368 uPSI_neuralbyteprediction CAI
37269: 1369 unit uPSI_USolarSystem; glscene.org
37270: 1370 uPSI_USearchAnagrams.pas DFF
37271: 1371 uPSI_JsonsUtilsEx.pas Randolph
37272: 1372 unit uPSI_Jsons.pas Randolph
37273: 1373 unit uPSI_HashUnit; DFF
37274: 1374 uPSI_U_Invertedtext.pas DFF
37275: 1375 unit uPSI_Bricks; Dendron
37276: 1376 unit uPSI_lifeblocks.pas Dendron
37277: 1377 unit uPSI_SystemsDiagram.pas Dendron
37278: 1378 unit uPSI_qsFoundation.pas Dendron
37279: 1379 uPSI_JclStringLists2 JCL
37280: 1380 uPSI_cInternetUtils2 FLC
37281: 1381 uPSI_cWindows.pas FLC
37282: 1382 uPSI_flcSysUtils.pas +TBytes utils
37283: 1383 unit uPSI_RotImg.pas DA
37284: 1384 uPSI_SimpleImageLoader.pas LAZ
37285: 1385 uPSI_HSLUtils.pas LAZ
37286: 1386 uPSI_GraphicsMathLibrary.pas EF
37287: 1387 unit uPSI_umodels.pas DMATH
37288: 1388 uPSI_flcStatistics.pas FLC5
37289: 1389 uPSI_flcMaths.pas FLC5
37290: 1390 uPSI_flcCharSet.pas FLC5
37291: 1391 uPSI_flcBits32.pas

```

```

37292: 1392 uPSI_flcTimers.pas
37293: 1393 uPSI_cBlaiseParserLexer.pas
37294: 1394 uPSI_flcRational.pas
37295: 1395 uPSI_flcComplex.pas
37296: 1396 unit uPSI_flcMatrix (uPSI_flcVectors.pas)
37297: 1397 unit uPSI_flcStringBuilder.pas
37298: 1398 unit PJResFile_Routines;
37299: 1399 uPSI_flcASCII.pas
37300: 1400 uPSI_flcStringPatternMatcher;
37301: 1401 unit uPSI_flcUnicodeChar.pas
37302: 1402 unit uXmlDates.pas;
37303: 1403 unit uPSI_SemaphorGrids;
37304: 1404 unit uXmlDates2;
37305: 1405 unit uPSI_JclTimeZones;
37306: 1406 unit uPSI_XmlDocRssParser.pas
37307: 1407 unit uPSI_RssParser.pas
37308: 1408 uPSI_SimpleParserRSS.pas
37309: 1409 unit uPSI_SimpleRSSUtils;
37310: 1410 unit uPSI_RssModel; _BlueHippo
37311: 1411 unit uPSI_StrUtil; _FIBPlus
37312: 1412 unit uPSI_TAChartUtils; _TEE
37313: 1413 unit uPSI_PythonEngine.pas _P4D_Beta
37314: 1414 unit uPSI_VclPythonGUIInputOutput;
37315: 1415 unit uPSI_VarPyth;
37316: 1416 unit JclUsesUtils;
37317: 1417 unit uPSI_cParameters;
37318: 1418 unit uPSI_WDCCMisc; (uPSI_cFileTemplates);
37319: 1419 uPSI_WDCColeVariantEnum.pas
37320: 1420 unit uPSI_WDCCWinInet.pas _WDCC
37321: 1421 uPSI_PythonVersions.pas _P4D_
37322: 1422 unit uPSI_PythonAction.pas _P4D_
37323: 1423 uPSI_SingleList.pas
37324: 1424 unit uPSI_AdMeter.pas; Async Professional
37325: 1425 unit uPSI_neuralplanbuilder; CAI
37326: 1426 unit uPSI_neuralvolume.pas; CAI
37327: 1427 unit uPSI_neuralvolumev.pas; CAI
37328: 1428 unit uPSI_DoubleList4; CapJack
37329: 1429 unit uPSI_ByteListClass; CapJack
37330: 1430 unit uPSI_flcVectors4; FLC5
37331: 1431 unit uPSI_flcMatrix4; FLC5
37332: 1432 uPSI_CurlHttpCodes.pas
37333: 1433 unit uPSI_NeuralNetwork.pas; CAI
37334: 1434 unit unit uPSI_neuralfit; CAI
37335: 1435 unit uPSI_neuraldatasets; CAI
37336: 1436 unit uPSI_neuraldatasetsv.pas CAI
37337: 1437 uPSI_flcFloats.pas FLC5
37338: 1438 unit UBigIntsForFloatV4.pas DFF
37339: 1439 unit uPSI_CustApp.pas Pas2js
37340: 1440 unit uPSI_NeuralNetworkCAI.pas CAI
37341: 1441 unit uPSI_neuralgeneric.pas; CAI
37342: 1442 unit uPSI_neuralthread.pas; CAI
37343: 1443 unit uPSI_uSysTools; TuO
37344: 1444 unit upsi_neuralsets; mX4
37345: 1445 unit uPSI_uWinNT.pas mX4
37346: 1446 unit uPSI_URungeKutta4.pas ICS
37347: 1447 unit uPSI_UrlConIcs.pas ICS
37348: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
37349: 1449 unit uPSI_Numedit2 mX4
37350: 1450 unit uPSI_PsAPI_3.pas mX4
37351: 1451 unit uPSI_SeSHA256.pas
37352: 1452 unit IdHashMessageDigest_max2;
37353: 1453 unit uPSI_BlocksUnit.pas
37354: 1454 unit uPSI_DelticsCommandLine.pas
37355: 1455 unit uPSI_DelticsStrUtils;
37356: 1456 unit uPSI_DelticsBitField;
37357: 1457 unit uPSI_DelticsSysUtils;
37358: 1458 unit uPSI_ALIniFiles2.pas
37359: 1459 unit uPSI_StarCalc2.pas
37360: 1460 unit uPSI_IdHashMessageDigest2.pas
37361: 1461 unit uPSI_U_Splines;
37362: 1462 unit uPSI_U_CoasterB.pas;
37363: 1463 U_SpringMass2.pas
37364: 1464 uPSI_MARSCoreUtils;
37365: 1465 unit uPSI_clJsonParser.pas
37366: 1466 unit uPSI_SynHighlighterPython.pas
37367: 1467 unit uPSI_DudsCommonDelphi;
37368: 1468 unit uPSI_AINNNeuron;
37369: 1469 unit uPSI_PJConsoleApp2; Console Application Runner
37370: 1470 unit uPSI_PJPipeFilters2; CAR
37371: 1471 unit uPSI_uHTMLBuilder;
37372: 1472 unit uPSI_PJPipe2;
37373: 1473 uPSI_WinApiDownload.pas
37374: 1474 uPSI_pxQRcode, //
37375: 1475 unit uPSI_neuralplanbuilder2
37376: 1476 unit uPSI_DelphiZXingQRcode;
37377: 1477 unit uPSI_RestJsonUtils;
37378: 1478 unit UtilsTimeCode;
37379: 1479 unit uPSC_classes2.pas; //TList
37380: 1480 unit uPSC_std2.pas

```

```
37381: 1481 unit uPSI_TBytes.pas
37382: 1482 unit uPSI_DelticsSysUtils2.pas
37383: 1483 unit uPSI_maxIniFiles.pas
37384: 1484 unit uROPSImports.pas
37385: 1485 unit uROPSServerLink.pas
37386: 1486 unit KLibVC_Redist.pas;
37387: 1487 unit HTTPApp2.pas;
37388: 1488 unit uPSI_XCollection2;
37389: 1489 unit uPSI_KLibWindows;
37390: 1490 unit KLibConstants;
37391: 1491 unit uPSI_AzuliasUtils.pas
37392: 1492 unit uPSI_ALHttpClient2;
37393: 1493 unit uPSI_ALWininetHttpClient2;
37394: 1494 unit uPSI_UtilsMax41.pas
37395: 1495 unit uPSI_JclSysUtils1;
37396: 1496 unit uPSI_RestUtils;
37397: 1497 unit uPSI_TeEngine2.pas
37398: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
37399: 1499 unit uPSI_PSResources.pas
37400: 1500 unit uPSI_TeCanvas2_1.pas
37401: 1501 unit uPSI_DataSetConverter4DUtil;
37402: 1502 unit uPSI_neuralfit2.pas;
37403: 1503 unit uPSI_SynCrtSock.pas
37404: 1504 uPSI_RunElevatedSupport.pas
37405: 1505 unit synTHttpRequest.pas;
37406: 1506 unit uPSI_VelthuisFloatUtils.pas
37407: 1507 unit HttpConnection.pas
37408: 1508 unit uPSI_HttpConnectionWinInet.pas
37409: 1509 unit UHexUtils.pas
37410: 1510 unit UExeFileType.pas
37411: 1511 unit uPSI_UConsoleApp.pas
37412: 1512 unit uPSI_CompilersURunner.pas
37413: 1513 unit uPSI_HttpConnection.pas
37414: 1514 unit uPSI_DataSetUtils.pas
37415: 1515 unit uPSI_HTTPSender.pas
37416: 1516 unit AES_Cryptobox4.pas
37417: 1517 unit uPSI_JsonConverter.pas
37418: 1518 unit uPSI_RestClient.pas
37419: 1519 unit JsonToDataSetConverter;
37420: 1520 unit JsonListAdapter; (superobject)
37421: 1521 unit uPSI_OpenApiUtils.pas;
37422: 1522 unit uPSI_WinHttp_TLB.pas;
37423: 1523 unit uPSI_NovusConsole;
37424: 1524 unit NovusShell.pas;
37425: 1525 unit NovusWebUtils.pas;
37426: 1526 unit NovusStreamUtils.pas;
37427: 1527 unit uPSI_dprocess; TProcess2
37428: 1528 unit uPSI_uXmlStorage.pas
37429: 1529 unit uPSI_AsphyreTimer.pas
37430: 1530 unit uPSI_Pas2JSUtils.pas
37431: 1531 unit uPSI_pacMain; (Formlpac: TFormlpac;)
37432: 1532 unit dwsWebUtils.pas; DWS
37433: 1533 unit uPSI_dwsWebUtils.pas; DWS
37434: 1534 unit uPSI_RestUtils2.pas;
37435: 1535 unit uPSI_Pas2jsFileUtils.pas; beta
37436: 1536 unit uPSI_JPerson.pas;
37437: 1537 unit uPSI_OldRttiMarshal;
37438: 1538 unit uPSI_superxmlparser;
37439: 1539 unit uPSI_superobject.pas; beta
37440: 1540 uPSI_NovusWindows.pas
37441: 1541 uPSI_NovusStringUtils.pas
37442: 1542 unit uPSI_NovusUtilities.pas;
37443: 1543 unit uPSI_NovusNumUtils;
37444: 1544 unit uPSI_NovusFileUtils;
37445: 1545 unit uPSI_NovusCURLUtils.pas; curl beta
37446: 1546 unit uPSI_uDM.pas;
37447: 1547 unit uPSI_dpipes.pas;
37448: 1548 unit uPSI_ShellAPI2;
37449: 1549 uPSI_NovusStringBuilder.pas
37450: 1550 unit NovusDateDiffUtil.pas
37451: 1551 unit NovusDateUtils;
37452: 1252 unit NovusDateStringUtils;
37453: 1553 unit uPSI_PJCBView;
37454: 1554 uPSI_NovusWinVersionUtils.pas
37455: 1555 unit uPSI_PJResFile.pas;
37456: 1556 unit PJResFile_Routines2;
37457: 1557 unit uPSI_JvCreateProcess2
37458: 1558 unit uPSI_JVCLHelpUtils.pas
37459: 1559 unit uPSI_ModuleLoader;
37460: 1560 unit uPSI_JvLogClasses;
37461: 1561 uPSI_uExporter.pas;
37462: 1562 unit uExporterDestinationCSV.pas;
37463: 1563 uPSI_uOptionParser.pas
37464: 1564 uPSI_TOptionDefs;
37465: 1565 unit uPSI_GUIUtils.pas
37466: 1566 unit uPSI_GUIAutomation.pas;
37467: 1567 unit uPSI_GUIActionRecorder; //dunit
37468: 1568 unit TypeHelpers.pas //dunit
37469: 1569 unit uPSI_API_base; //API
```

```
37470: 1570 unit uPSI_API_audio; //API
37471: 1571 unit uPSI_API_ledgrid;
37472: 1572 uPSI_API_graphics.pas;
37473: 1573 uPSI_API_files.pas;
37474: 1574 uPSI_API_tools.pas;
37475: 1575 unit uPSI_API_winprocess;
37476: 1576 unit API_strings.pas;
37477: 1577 unit uPSI_API_services.pas
37478:
37479: Total of Function Calls: 37372
37480: SHA1: 4.7.6.50 D047DBD5412C3E4A436089018B9C7FACF17A2EB5
37481: CRC32: 38562FA8 34.04 MB (35,697,944 bytes)
37482: Compilation Timestamp 2023-06-15 06:40:19 UTC Signtime 15 June 2023 08:42:33
37483: Entry Point 25484072 - Contained Sections 10
37484: sha1: d047dbd5412c3e4a436089018b9c7facf17a2eb5
37485: sha256: 193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7
37486: Docu: http://www.softwareschule.ch/maxbox_functions.txt
37487: ZIP maxbox4.zip SHA1: CEAB242D25E264FB95D5792EBC569C4EAD31B732
37488: https://www.hybrid-analysis.com/sample/193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7
37489:
37490:
37491: Total of Function Calls: 37121
37492: SHA1: 4.7.6.50 9E9D9D10762AE0D0BC8AFD747C2381EA5478AE49
37493: CRC32: 01B431EE 33.91 MB (35,558,168 bytes)
37494: Compilation Timestamp 2023-06-12 13:17:28 UTC Signtime 12 June 2023 15:20:10
37495: Entry Point 25377544 - Contained Sections 10
37496: sha1: 9e9d9d10762ae0d0bc8afd747c2381ea5478ae49
37497: sha256: fa0f30abf34292e91070a5bd4682040eb6af79a8f6c7f55111c9692153120988
37498: Docu: http://www.softwareschule.ch/maxbox_functions.txt
37499:
37500: Total of Function Calls: 37008
37501: SHA1: 4.7.6.50 FDA4E3CE0FEC5F86EF0F95278D5C35FEC42FDA1B
37502: fda4e3ce0fec5f86ef0f95278d5c35fec42fda1b
37503: CRC32: 46294349 33.81 MB (35,454,744 bytes)
37504: Compilation Timestamp 2023-06-08 17:00:33 UTC Signtime 08 June 2023 19:04:06
37505: Entry Point 25320128 - Contained Sections 10
37506: sha1: fda4e3ce0fec5f86ef0f95278d5c35fec42fda1b
37507: sha256: 882ddadb9b330318bfff73d8db66d1a33be575be4cedec3960374bfeb7c49c968
37508:
37509: Total of Function Calls: 36947
37510: SHA1: 4.7.6.50 C430FDBA9880317E31D4DA2F66C884799298FCC3
37511: CRC32: 5BD9839C 33.62 MB (35,257,624 bytes)
37512: Compilation Timestamp 2023-06-06 14:16:08 UTC Signtime 06 June 2023 16:22:14
37513: Entry Point 25287360 - Contained Sections 10
37514:
37515: Total of Function Calls: 36852
37516: SHA1: 4.7.6.50 D4FD4CACFD766EB8F78F2BB7B5EFDDEBB386597A
37517: CRC32: C1DCD693 33.50 MB (35,128,600 bytes)
37518: Compilation Timestamp 2023-05-31 12:44:17 UTC Signtime 31 May 2023 14:48:34
37519: Entry Point 25234104 - Contained Sections 10
37520: sha1: d4fd4cacfd766eb8f78f2bb7b5efdddebb386597a
37521: sha256: 4dfbada6765e47c72b7c2496f831419b3461fa7c4ac6e05e2a941b501e10e022
37522:
37523: Total of Function Calls: 36834
37524: SHA1: 4.7.6.50 EDE730F514834A85ECA6D0190DAC8CA6046C26EE
37525: ede730f514834a85eca6d0190dac8ca6046c26ee
37526: CRC32: 4E7C4A2B 33.49 MB (35,117,336 bytes)
37527: Compilation Timestamp 2023-05-31 08:25:23 UTC Signtime 31 May 2023 10:33:00
37528: Entry Point 25225912 - Contained Sections 10
37529: sha1: ede730f514834a85eca6d0190dac8ca6046c26ee
37530: sha256: 168f85cfffecdcfcc4e7614758bbdb333a38f4896f94b7056f3e53af8fd15a66b
37531:
37532: Total of Function Calls: 36798
37533: SHA1: 4.7.6.50 CAA09249B338ED2814B760FA549DAD47F6C789D2
37534: CRC32: 94780E06 33.41 MB (35,033,880)
37535: Compilation Timestamp 2023-05-30 13:17:50 UTC Signtime 30 May 2023 15:20:39
37536: Entry Point 25217720 - Contained Sections 10
37537: http://www.softwareschule.ch/maxbox_functions.txt
37538: sha1: caa09249b338ed2814b760fa549dad47f6c789d2
37539:
37540: Total of Function Calls: 36354
37541: SHA1: 4.7.6.20 F60338A77B77F2032061BF72A545AFB727F6395F
37542: CRC32: 48455EF8 32.8 MB (34,419,992 bytes)
37543: Compilation Timestamp 2023-01-26 15:36:15 UTC Signtime 26 Jan. 2023 16:41:42
37544: Entry Point 25033256 - Contained Sections 10
37545: ZIP maxbox4.zip SHA1: 7E6C1A422CCE02B90CC96A8AB994BDD235A4D02A
37546: http://www.softwareschule.ch/maxbox_functions.txt
37547:
37548: mX4 executed: 26/01/2023 17:27:09 Runtime: 0:15:27.415 Memload: 48% use
37549:
37550: Total of Function Calls: 35958
37551: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
37552: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
37553: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
37554: Entry Point 24783132 - Contained Sections 10
37555:
37556: *****
37557: Release Notes maXbox 4.7.6.20 January 2023 mX476
37558: *****
```

```
37559: Add 86 Units + 25 Tutorials
37560:
37561: 1441 unit uPSI_neuralgeneric.pas; CAI
37562: 1442 unit uPSI_neuralthread.pas; CAI
37563: 1443 unit uPSI_uSysTools; TuO
37564: 1444 unit upsi_neuralsets; mX4
37565: 1445 unit uPSI_uWinNT.pas mX4
37566: 1446 unit uPSI_URungeKutta4.pas ICS
37567: 1447 unit uPSI_UrlConIcs.pas ICS
37568: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
37569: 1449 unit uPSI_Numedit2 mX4
37570: 1450 unit uPSI_PsAPI_3.pas mX4
37571: 1451 unit uPSI_SeSHA256.pas
37572: 1452 unit IdHashMessageDigest_max2;
37573: 1453 unit uPSI_BlocksUnit.pas
37574: 1454 unit uPSI_DelticsCommandLine.pas
37575: 1455 unit uPSI_DelticsStrUtils;
37576: 1456 unit uPSI_DelticsBitField;
37577: 1457 unit uPSI_DelticsSysUtils;
37578: 1458 unit uPSI_ALIniFiles2.pas
37579: 1459 unit uPSI_StarCalc2.pas
37580: 1460 unit uPSI_IdHashMessageDigest2.pas
37581: 1461 unit uPSI_U_Splines;
37582: 1462 unit uPSI_U_CoasterB.pas;
37583: 1463 U_SpringMass2.pas
37584: 1464 uPSI_MARSCoreUtils;
37585: 1465 unit uPSI_clJsonParser.pas
37586: 1466 unit uPSI_SynHighlighterPython.pas
37587: 1467 unit uPSI_DudsCommonDelphi;
37588: 1468 unit uPSI_AINNeuron;
37589: 1469 unit uPSI_PJConsoleApp2;
37590: 1470 unit uPSI_PJPipeFilters2;
37591: 1471 unit uPSI_uHTMLBuilder;
37592: 1472 unit uPSI_PJPipe2;
37593: 1473 uPSI_WinApiDownload,
37594: 1474 uPSI_pxQRcode, //beta
37595: 1475 unit uPSI_neuralplanbuilder2
37596: 1476 unit uPSI_DelphiZXingQRCode;
37597: 1477 unit uPSI_RestJsonUtils;
37598: 1478 unit UtilsTimeCode;
37599: 1479 unit uPSC_classes2.pas; //TList
37600: 1480 unit uPSC_std2.pas
37601: 1481 unit uPSI_maxIniFiles.pas
37602: 1482 unit uROPSImports.pas
37603: 1483 unit uROPSServerLink.pas
37604: 1484 unit uPSI_KLibUtils;
37605: 1485 unit uPSI_PathFunc2; //inno setup
37606: 1486 unit KLibVC_Redist.pas;
37607: 1487 unit HTTPApp2.pas;
37608: 1488 unit uPSI_XCollection2;
37609: 1489 unit uPSI_KLibWindows;
37610: 1490 unit KlibConstants;
37611: 1491 unit uPSI_AzulialUtils.pas
37612: 1492 unit uPSI_ALHttpClient2;
37613: 1493 unit uPSI_ALWininetHttpClient2;
37614: 1494 unit uPSI_UtilsMax41.pas
37615: 1495 unit uPSI_JclSysUtils1;
37616: 1496 unit uPSI_RestUtils;
37617: 1497 unit uPSI_TeEngine2.pas
37618: 1498 unit uPSI_Chart2.pas; (uPSI_TeCanvas2.pas)
37619: 1499 unit uPSI_PSResources.pas
37620: 1500 unit uPSI_TeCanvas2_1.pas
37621: 1501 unit uPSI_DataSetConverter4DUtil;
37622: 1502 unit uPSI_neuralfit2.pas;
37623: 1503 unit uPSI_SynCrtSock.pas
37624: 1504 uPSI_RunElevatedSupport.pas
37625: 1505 unit synTHttpRequest.pas;
37626: 1506 unit uPSI_VelthuisFloatUtils.pas
37627: 1507 unit HttpConnection.pas
37628: 1508 unit uPSI_HttpConnectionWinInet.pas
37629: 1509 unit UHexUtils.pas
37630: 1510 unit UExeFileType.pas
37631: 1511 unit uPSI_UConsoleApp.pas
37632: 1512 unit uPSI_CompilersURunner.pas
37633: 1513 unit uPSI_HttpConnection.pas
37634: 1514 unit uPSI_DataSetUtils.pas
37635: 1515 unit uPSI_HTTPSender.pas
37636: 1516 unit AES_Cryptobox4.pas
37637: 1517 unit uPSI_JsonConverter.pas
37638: 1518 unit uPSI_RestClient.pas
37639: 1519 unit JsonToDataSetConverter;
37640: 1520 unit JsonListAdapter; (superobject)
37641: 1521 unit uPSI_OpenApiUtils.pas;
37642: 1522 unit uPSI_WinHttp_TLB.pas;
37643: 1523 unit uPSI_NovusConsole;
37644: 1524 unit NovusShell.pas;
37645: 1525 unit NovusWebUtils.pas;
37646: 1526 unit NovusStreamUtils.pas;
37647:
```



```

37648:
37649: *****
37650: Release Notes maXbox 4.7.6.10 November 2021 mX476
37651: *****
37652: Add 5 Units + 2 Tutorials
37653:
37654: 1441 unit uPSI_neuralgeneric.pas; CAI
37655: 1442 unit uPSI_neuralthread.pas; CAI
37656: 1443 unit uPSI_uSysTools; TuO
37657: 1444 unit upsi_neuralsets; mX4
37658: 1445 unit uPSI_uWinNT.pas mX4
37659: Total of Function Calls: 34880
37660: SHA1: of 4.7.6.10 CF939E3A8D4723DB1DEF383C5FC961E06728C58F
37661: CRC32: 38F88218 30.5 MB (32,022,344 bytes)
37662: V 4.7.5.90 VI
37663: Amount of Functions: 20549
37664: Amount of Procedures: 12307
37665: Amount of Constructors: 1937
37666: Amount of Destructors: 14
37667: Totals of Calls: 34807
37668: SHA1: of 4.7.5.90 96DCDE2028125E00B67E42A801721AC513A5EAFc
37669: CRC32: BBC3A7E5: 31974216 bytes
37670: 000 mX4 executed: 03/11/2021 15:25:47 Runtime: 0:12:33.807 Memload: 39% use
37671: Amount of Functions: 20994
37672: Amount of Procedures: 12492
37673: Amount of Constructors: 1984
37674: Amount of Destructors: 14
37675: Totals of Calls: 35484
37676: SHA1: of 4.7.6.10 EC20D0A371A57AA07862D6D45F7229A370549A1
37677: CRC32: 82E481A4: 33273160 bytes
37678: 000 mX4 executed: 14/03/2022 15:21:38 Runtime: 0:13:41.253 Memload: 42% use
37679: Amount of Functions: 21907
37680: Amount of Procedures: 12848
37681: Amount of Constructors: 2029
37682: Amount of Destructors: 14
37683: Totals of Calls: 36798
37684: SHA1: of 4.7.6.50 CAA09249B338ED2814B760FA549DAD47F6C789D2
37685: CRC32: 17A61798: 35033880 bytes
37686: 000 mX4 executed: 30/05/2023 15:46:15 Runtime: 0:16:22.226 Memload: 40% use
37687:
37688: // Project: State Machine
37689: // Module: statmach.pas
37690: // Description: Visual Finite State Machine.
37691: // Version: 2.2a - Release: 6
37692:
37693: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
37694: http://www.slideshare.net/maxkleiner1/codereview-topics
37695: ///////////////////////////////////////////////////////////////////
37696: //Form Template Library FTL
37697: ///////////////////////////////////////////////////////////////////
37698:
37699: FTL For Form Building Lib out of the Script, eg. 399 form_templates.txt
37700: 045 unit uPSI_VListView TFormListView;
37701: 263 unit uPSI_JvProfiler32; TProfReport
37702: 270 unit uPSI_ImgList; TCustomImageList
37703: 278 unit uPSI_JvImageWindow; TJvImageWindow
37704: 317 unit uPSI_JvParserForm; TJvHTMLParserForm
37705: 497 unit uPSI_DebugBox; TDebugBox
37706: 513 unit uPSI_ImageWin; TImageForm, TImageForm2
37707: 514 unit uPSI_CustomDrawTreeView; TCustomDrawForm
37708: 515 unit uPSI_GraphWin; TGraphWinForm
37709: 516 unit uPSI_actionMain; TActionForm
37710: 518 unit uPSI_CtlPanel; TAppletApplication
37711: 529 unit uPSI_MDIEdit; TEditForm //RichEditApp
37712: 535 unit uPSI_CoolMain; {browser} TWebMainForm
37713: 538 unit uPSI_frmExportMain; TSynexportForm
37714: 585 unit uPSI_usniffer; {PortScanForm} TSniffForm
37715: 600 unit uPSI_ThreadForm; TThreadSortForm;
37716: 618 unit uPSI_delphi_arduino_Unit1; TLEDForm
37717: 620 unit uPSI_fplotMain; TfplotForm1
37718: 660 unit uPSI_JvDBQueryParamsForm; TJvQueryParamsDialog
37719: 677 unit uPSI_OptionsFrm; TfrmOptions;
37720: 718 unit uPSI_MonForm; TMonitorForm
37721: 742 unit uPSI_SimplePortMain; TPortForm1
37722: 770 unit uPSI_ovccalc; TOvcCalculator //widget
37723: 810 unit uPSI_DbExcept; TDbEngineErrorDlg
37724: 812 unit uPSI_GL_actorUnit1; TglActorForm1 //OpenGL Robot
37725: 846 unit uPSI_synhttp_daemon; TTCPhTtpDaemon, TTCPhTtpThrd, TPingThread
37726: 867 unit uPSI_Debug; TfrmDebug
37727: 901 unit uPSI_JvAddPrinter TJvAddPrinter //JCL
37728: 904 unit uPSI_U_HexView; THexForm2
37729: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain) TOscfrmMain
37730: 959 unit uPSI_AdSelCom; TComSelectForm
37731: 1029 unit uPSI_GPSUDemo; TFDemo
37732: 1031 unit uPSI_ScreenThreeDLab; TFormLab3D
37733: 1051 unit uPSI_panUnit1; TPanForm1 //GLScene
37734: 1052 unit uPSI_DD83ul; {Arduino Tester Frm} TDD83f1
37735: 1059 unit uPSI_uconvMain; TfconvMain //PS
37736: 1076 unit uPSI_CustomizeDlg; TCustomizeDlg / TJvAddPrinterDialog;

```

```

37737: 1111 unit uPSI_JvExceptionForm;                                TjvErrorDialog //ShowException
37738: 1113 unit uPSI_MTMainForm;                                    TvtMainForm
37739: 1119 unit uPSI_CPortTrmSet;                                  TComTrmSetForm
37740: 1146 unit uPSI_ChessPrg;                                     TChessForm1
37741: 1216 uPSI_KDialogs,                                         TKBrowseFolderDialog
37742: 1217 eg,execStr('from delphivcl import *');                 LoadPyClass
37743: 1231 unit uPSI_pacMain; (Formlpac: TFormlpac;) TFormlpac;
37744:
37745: FormTemplates with <Ctrl> J
37746:
37747: [myformtemplate | formtemplate statement | Borland.EditOptions.Pascal] //with
37748: [myFastForm | class declaration | Borland.EditOptions.Pascal] //Dialog Form
37749: [myForm | class declaration | Borland.EditOptions.Pascal] //with Events
37750: [aForm | class declaration | Borland.EditOptions.Pascal] //single Form
37751: [getForm(x,y)] of object TForm [loadForm(x,y)]
37752: [ticker | ticker statement | Borland.EditOptions.Pascal]
37753: [except | exception statement | Borland.EditOptions.Pascal]
37754:
37755: Dark Color Style Darkcolor:
37756: memo1.color:= RGB2TColor(32,32,32) //20,18,18
37757: memo2.color:= RGB2TColor(20,18,18)
37758:
37759: Proc SIRegister_JvDesignUtils(CI: TPSPascalCompiler);
37760: begin
37761:   Func DesignClientToParent(const Apt: TPoint; AControl,AParent: TControl) : TPoint;
37762:   Func DesignMin(AA, AB : Int):Int;
37763:   Func DesignMax(AA, AB: Int): Int;
37764:   Func DesignRectWidth( const ARect : TRect) : Int;
37765:   Func DesignRectHeight( const ARect : TRect) : Int;
37766:   Func DesignValidateRect( const ARect : TRect) : TRect;
37767:   Func DesignNameIsUnique( AOwner : TComponent; const AName :Str) :Bool;
37768:   Func DesignUniqueName( AOwner : TComponent; const AClassName :Str) :Str;
37769:   Proc DesignPaintRubberbandRect(AContainer:TWinControl;ARect:TRect;APenStyle:TPenStyle);
37770:   Proc DesignPaintGrid(ACanvs:TCanvas;const ARect:TRect;ABackClr:TColor;AGridClr:TColor;ADivPixels:Int)
37771:   Proc DesignPaintRules(ACanvas:TCanvas;const ARect:TRect;ADivPixels:Int;ASubDivs:Bool;
37772:   Proc DesignSaveComponentToStream( AComp : TComponent; AStream : TStream);
37773:   Func DesignLoadComponentFromStream(AComp:TComponent;AStream:TStream;AOnError:TReaderError):TComponent;
37774:   Proc DesignSaveComponentToFile( AComp : TComponent; const AFileName :Str);
37775:   Proc DesignLoadComponentFromFile(AComp:TComp;const AFileName:str;AOnError:TReaderError);
37776: end;
37777:
37778: ex.:with TEditForm.create(self) do begin
37779:   caption:= 'Template Form Tester';
37780:   FormStyle:= fsStayOnTop;
37781:   with editor do begin
37782:     Lines.LoadFromFile(Exepath+'\\docs\\Readme_rus_mX2.rtf
37783:     SelStart:= 0; Modified:= False;
37784:   end;
37785: end;
37786: with TWebMainForm.create(self) do begin
37787:   URLs.Text:= 'http://www.kleiner.ch';
37788:   URLs.Click(self); Show;
37789: end;
37790: with TSynexportForm.create(self) do begin
37791:   Caption:= 'Synexport HTML RTF tester';
37792:   Show;
37793: end;
37794: with TThreadSortForm.create(self) do begin
37795:   showmodal; free;
37796: end;
37797: with TCustomDrawForm.create(self) do begin
37798:   width:=820; height:=820;
37799:   image1.height:= 600; //add properties
37800:   image1.picture.bitmap:= image2.picture.bitmap;
37801:   //SelectionBackground1Click(self) CustomDraw1Click(self);
37802:   Background1.click;
37803:   bitmap1.click; Tile1.click;
37804:   Showmodal;
37805:   Free;
37806: end;
37807: with TfplotForm1.Create(self) do begin
37808:   BtnPlotClick(self);
37809:   Showmodal; Free;
37810: end;
37811: with TOvcCalculator.create(self) do begin
37812:   parent:= aForm; //free;
37813:   setbounds(550,510,200,150);
37814:   displaystr:= 'maXcalc';
37815: end;
37816: with THexForm2.Create(self) do begin
37817:   ShowModal;
37818:   Free;
37819: end;
37820:
37821: Func CheckBox:Str;
37822: var idHTTP: TIDHTTP;
37823: begin
37824:   result:= 'version not found';
37825:   if IsInternet then begin

```

```

37826: idHTTP:= TIdHTTP.Create(NIL);
37827: try
37828:   result:= idHTTP.Get(MXVERSIONFILE2);
37829:   result:= result[1]+result[2]+result[3]+result[4]+result[5];
37830:   if result = MBVER2 then begin
37831:     //Speak(' A new Version '+vstr+' of max box is available! ;
37832:     result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
37833:   end; //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
37834: finally
37835:   idHTTP.Free
37836: end;
37837: end;
37838: end;
37839:
37840: charinset make your own charinset to prevent typematch:
37841:
37842: Func CharInSet4(C: AnsiChar; const CharSet: TSysCharSet):Bool;
37843: begin
37844:   Result:= C in CharSet;
37845: end;
37846:
37847: Func CharInSet5(C: AnsiChar; const CharSet: CharSet):Bool;
37848: begin
37849:   Result:= C in CharSet;
37850: end;
37851:
37852: //Runtime Spec Functions Edition 190 and more code blocks
37853: Func ApWinExecAndWait32(FileName:PChar;CommandLine:PChar; Visibility:Int):Int;
37854: Func KillTask(ExeFileName:Str): Int;
37855: Proc KillProcess(hWindowHandle: HWND);
37856: Func FindWindowByTitle(WindowTitle:Str): Hwnd;
37857: Func OpenIE(aURL:Str):Bool;
37858: Func XRTLIIsInMainThread:Bool;
37859: Func IsInMainThread:Bool;
37860: Func IntToFloat(i: Int): double;
37861: Func AddThousandSeparator(S:Str; myChr: Char):Str;
37862: Func mciSendString(cmd: PChar; ret: PChar; len: Int; callback: Int):Card;
37863: Proc FormAnimation(Sender: TObject; adelay: Int);
37864: Proc LoadResourceFile2(aFile:Str; ms:TMemoryStream);
37865: Func putBinResTo(binresname: pchar; newpath:Str):Bool;
37866: Proc ExecuteHyperlink(Sender:TObject;HyperLinkClick:TJvHyperLinkClickEvent;const LinkName:Str);
37867: Func IsHyperLink(Canvas:TCanvas;Rect:TRect;const Text:Str;MouseX,MouseY:Int;var HyperLink:Str):Bool
37868: Func GetWindowThreadProcessId( hWnd : HWND; var dwProcessId : DWORD) : DWORD;;
37869: Func GetWindowTask( hWnd : HWND) : THandle;
37870: Func LoadBitmap( hInstance : HINST; lpBitmapName : PChar) : HBITMAP;
37871: Func GetCommConfig(hCommDev: THandle; var lpCC: TCommConfig; var lpdwSize: DWORD): BOOL;
37872: Func WinExecAndWait32(FileName:Str; Visibility: Int): Longword;
37873: Func MakeHash( const s : TbtString) : Longint;
37874: Func GetUsedUnitList( list : TStringlist) :Str;
37875: Func ConsoleCapture(const _dirName, _exeName, _cmdLine:Str; amemo: TStringlist): Bool;
37876: Func ConsoleCaptureDOS(const _dirName, _exeName, _cmdLine:Str; amemo: TStringlist): Bool;
37877:   srlist:= TStringlist.create;
37878:   ConsoleCapture('C:\', 'cmd.exe', '/c dir *.*',srlist);
37879:   writeln(srlist.text); srlist.Free;
37880: Func RunCaptured(const _dirName, _exeName, _cmdLine:Str; amemo:TStringlist):Bool;;
37881: Func SamePropTypeName( const Name1, Name2 : ShortString) :Bool;
37882: Func FloatToStrEx( Value : Extended) :Str;
37883: Func StrToFloatEx( const S :Str) : Extended;
37884: Proc PerformanceDelayMS(ams: Int); //microsecond resolution delay!
37885: //http://www.swissdelphicenter.ch/en/showcode.php?id=2179
37886: Func ExecuteProcess(FileName:Str;Visibility:Int; BitMask:Int; Synch:Boolean):Longword;
37887: Func ExecuteMultiProcessor(FileName:Str;Visibility:Int;BitMask:Int;Synch:Bool):Longword;
37888:   if ExecuteMultiProcessor('notepad.exe', SW_SHOW, 2, true) = 0 then
37889:     writeln('Multiprocessing Runs on CPU 2;
37890: Proc StartServiceAfterInstall(aserv: TService);
37891: Func GetDllVersion2(DllName:Str; var DLLVersionInfo: TDLLVersionInfo):Bool;
37892: Proc SendCopyMessage(amess,astation:Str); //communicate process-spanned with WM_COPYDATA
37893: Func BrowseComputer2(DialogTitle:Str; var CompName:Str; bNewStyle: Bool): Bool;
37894: Func ChangeAlphaTo(input:Str; aoffset: byte):Str;;
37895: Func CheckIBAN(iban:Str):Bool;;
37896: Func CreateIDStack; //instance to IdStack of CreateStack
37897: Func GetRecordCount(DataSet: TBDEDataSet): Longint;;
37898: Func CountPos(const subtxt:Str; Text:Str): Int;
37899: Proc HTMLtoRTF(html:Str; var rtf: TRichedit);
37900: Proc ReversePlay(const szFileName:Str);
37901: Func ADOConnectionString(ParentHandle:THandle;InitialStr:WideString;out NewString:Str):Bool;
37902: Proc ShowEOleException(AExc: EOleException; Query:Str);
37903: Func UpdateBlob(Connection:TADOConnection;Spalte:Str;Tabelle:Str;Where:Str;var
ms:TMemoryStream):Bool;..save HTML pages as MHTML(HTML Archiv Format)
37904: http://www.swissdelphicenter.ch/en/showcode.php?id=2300
37905: Func SaveToMHT(const AUrl,AFileName:Str; AShowErrorMessage:Boolean = False):Bool;
37906: Func FileType2MimeType(const AFileName:Str):Str;
37907: Func DownloadURL_NOCache(const aUrl:Str; var s:Str):Bool;
37908: Func IsCOMObjectActive(Classname:Str):Bool;
37909: Proc CopyHTMLToClipboard(const str:Str; const htmlStr:Str = '
37910: Func CheckCreditCard(c:Str): Int;
37911: 0: Card is invalid or unknown 1: is a AmEx 2: is a Visa 3: is a valid MasterCard
37912: Func GetNewGUID:Str;
37913: Func FormatGUID(const GUID:Str):Str;

```

```

37914: Func GetNewFormattedGUID:Str;
37915: Func getFormRes(classname:Str):Str; //shows DFM Res of Exe!
37916: Proc OutputDebugString(PChar(Format('%s[%s]s', [aCaption, GetFormatDT(StartDT), aText])));
37917: Proc ScanNetworkResources(ResourceType, DisplayType: DWord; List: TStringList);
37918: Func PrepareConstraint(Src:Tstrings):str;
37919: Proc DeleteEmptyStr(Src:Tstrings);
37920: Func NormalizeSQLText(const SQL:Str;MacroChar:Char):Str;
37921: Func CountSelect(const SrcSQL:str):str;
37922: Func GetModifyTable(const SQLText:str;AlreadyNormal:boolean):str;
37923: Func GetCharFromVKey(vkey: Word):Str;
37924: Func Xls_To_StringGrid(AGrid: TStringGrid; AXLSFile:Str):Bool;
37925: Func IsObjectActive(Classname:Str):Bool;
37926: Func GetActiveObject(ClassID:TGUID; anil:TObject; aUnknown:IUnknown):HRESULT;;
37927: Func RegisterOCX(FileName:Str):Bool;
37928: Func UnRegisterOCX(FileName:Str):Bool;
37929: Func RegisterServer2(const aDllFileName:Str; aRegister:Bool):Bool;
37930: Proc mIRCDE(Service, Topic, Cmd:Str); //mIRCDE('mIRC', 'COMMAND', '/say Hallo von SwissDelphiCenter.ch;
37931: Func OpenIE(aURL:Str):Bool;
37932: Func XRTLIIsInMainThread:Bool;
37933: Func IsInMainThread:Bool;
37934: TryConvertStrToDateTime(const s, format:str; out value: TDateTime):Boolean;;
37935: ConvertStrToDateTime(const s, format:Str): TDateTime;;
37936: Func CreateNotifyThread2(const FolderName:Str;
WatchSubtree:Bool;Filter:TFileChangeFilters2):TNotifyThread;
37937: Proc DetectImage(const InputFileName:Str; BM: TBitmap);
37938: Func BitmapToString(Bitmap: TBitmap):Str;
37939: Func StringToBitmap(S:Str): TBitmap;
37940: Func RemoveChar(CONST s:Str; CONST c: CHAR):Str;
37941: Proc SecureClearStr(var S: Ansistr);
37942: Proc movestring(const Source:str; var Destination:str; CopyCount : Int );
37943: Proc movestringJV(const Source:str; var Destination:str; CopyCount : Int );
37944: Proc moveint(const Source:Int; var destination: Int; CopyCount : Int );
37945: Proc movefloat(const Source:double; var destination: double; CopyCount : Int );
37946: Proc moveextended(const Source:extended; var Destination:extended; CopyCount:Int);
37947: Proc ShowFilePropertiesSH(Files: TStringList; aWnd: HWND);
37948: //ScanNetworkResources(RESOURCE_TYPE_DISK, RESOURCE_DISPLAYTYPE_SERVER, ListBox1.Items);
37949: Func GrabLine(const s:Str; ALine: Int):Str;
37950: Func GrabLineFast(const s:Str; ALine: Int):Str;
37951: Func IsTextFile(const sFile: TFileName):Bool;
37952: Func getODBC: TStringList;
37953: Func getODBCString:Str;
37954: Proc GetJPGSize(const sFile:Str; var wWidth, wHeight: Word);
37955: Proc GetPNGSize(const sFile:Str; var wWidth, wHeight: Word);
37956: function GetPNGSize2(const FileName: string): TSize;
37957: Proc GetGIFSize(const sGIFFile:Str; var wWidth, wHeight: Word);
37958: // note: using TForm's BorderIcons, etc. is slow (form blinks) and not reliable (for some
37959: // reason it causes TListView.Items to lose all associated objects and other things happen).
37960: Proc ChangeWindowState(const Form: HWND; Style: DWord; AddIt:Bool);
37961: Func VarByteArrayOf(const s:Str): OleVariant;
37962: type TView = (normal, Scientific);
37963: Func bigdiv2(aval:Str; aval2: integer):Str;
37964: Func modbig(aval:Str; amod: integer): integer;
37965: Func bigmod(aval:Str; amod: integer): integer;
37966: Func modPowBig3(aval, apow, amod:Str):Str;;
37967: Func BigPowMod(aval, apow, amod:Str):Str;;
37968: Function BigLog(atwo:String; sig: integer):String;
37969: Func RSAEncrypt(aval, apow, amod:Str):Str;;
37970: Func RSADecrypt(aval, apow, amod:Str):Str;;
37971: Func SwitchToThread : BOOL;
37972: Func SetThreadDesktop( hDesktop : HDESK) : BOOL;
37973: Func CloseDesktop( hDesktop : HDESK) : BOOL;
37974: Func GetThreadDesktop( dwThreadId : DWORD) : HDESK;
37975: Func SetSyscallHook():Bool;;
37976: Func SetSwapcontextHook():Bool;;
37977: Func UnhookAll():Bool;;
37978: Func getWorld:Str;;
37979: Func getIPConfigAll:Str;;
37980: Func getIPConfig:Str;;
37981: Func WinsockEnabled:Bool;
37982: Func HTTPEncode2(const AStr:Str):Str;
37983: Func ComputePEChecksum(FileName:Str): DWord;
37984:
37985: if not DynamicDllCallName(user32, 'LockWorkStation',true,returned,parameters) then begin
37986: Func DynamicDllCallNames(Dll:Str; const Name:Str; HasResult:Bool; var Returned:Card; const Parameters:
array of string):Bool;;
37987: Proc GetMIDASAppServerList(List: TStringList; const RegCheck :Str);
37988: Proc SQLDropField(dbName, tblName, fldName:Str); {Field Name to Drop}
37989: type TCastType = (ctSmallInt, ctInt, ctDecimal, ctNumeric, ctFloat,
ctChar, ctVarChar, ctDate, ctBoolean, ctBLOB, ctTime,
ctTimeStamp, ctMoney, ctAutoInc, ctBytes);
37990: {Blob definition type 1 = Memo, 2 = Binary, 3 = Formatted Memo, 4 = OLE Object, 5 = Graphic}
37991:
37992: Proc SQLAddField(dbName,tblName,fldName:str;fldType:TCastType;fldLength,precisOrBlobLen,
scaleOrBlobType:Int);
37993:
37994: const
37995: UrlGeoLookupInfo ='http://ipinfodb.com/ip_query.php?timezone=true&ip=%s';
37996: UrlGeoLookupInfo2 ='http://backup.ipinfodb.com/ip_query.php?timezone=true&ip=%s'; //backup
37997: http://ip-api.com/csv/
37998:
37999:

```

```

38000: Proc GetGeoInfo(const IPAddress:str;var GeoInfo:TGeoInfo;const UrlGeoLookupInfo:str);
38001:   TGeoInfo,'record status:str; countrycode : '
38002:   +string; countryname:str; regioncode:Str;city:Str;zippostalcode:Str;
      latitude:Str;longitude:str;timezonename:str;gmtoffset:str;isdst:str; end;
38003:
38004: Proc SIRegister_ubigFloatV3(CL: TPSPascalCompiler);
38005: begin
38006:   TMaxSig', 'Int;
38007:   TView', '( normal, Scientific );
38008:   SIRegister_TFloatInt(CL); SIRegister_TBigFloat(CL);
38009: end;
38010:
38011: Proc SIRegister_UBigIntsV4(CL: TPSPascalCompiler);
38012: begin TDigits', 'array of int64; SIRegister_TInt(CL);
38013: Proc SetBaseVal( const newbase : Int);
38014: Func GetBasePower : Int;
38015: Func GetBase : Int;
38016: Proc SetThreadSafe( newval :Bool);
38017: Func bigdiv2(aval1:Str; aval2: integer):Str;
38018: Func modbig(aval:Str; amod: integer): integer; ;
38019: Func bigmod(aval:Str; amod: integer): integer; ;
38020: Func modPowBig3(aval, apow, amod:Str):Str;;
38021: Func BigPowMod(aval, apow, amod:Str):Str;;
38022: Func BigLog(atwo:String; sig: integer):String; //UBigFloat3
38023: Func RSAEncrypt(aval, apow, amod:Str):Str;;
38024: Func RSADecrypt(aval, apow, amod:Str):Str;;
38025: Func SwitchToThread : BOOL;
38026: end;
38027:
38028: Func BigDiv(aone, atwo:Str):Str;
38029: var tbig1, tbig2: TInt;
38030: begin
38031:   tbig1:= TInt.create(10);
38032:   tbig2:= TInt.create(10);
38033:   try
38034:     tbig1.assign2(atwo)
38035:     tbig2.assign2(aone)
38036:     tbig2.Divide(tbig1)
38037:   finally
38038:     result:= tbig2.ConvertToDecimalString(false)
38039:     tbig1.Free;
38040:     tbig2.free;
38041:   end;
38042: end;
38043:
38044: Function BigLog(atwo:String; sig: integer):String;
38045: var tbig2: TBigFloat;
38046: begin
38047:   tbig2:= TBigFloat.create;
38048:   try
38049:     tbig2.assign8(atwo)
38050:     tbig2.log(sig)
38051:   finally
38052:     result:= tbig2.toString(normal)
38053:     tbig2.free;
38054:   end;
38055: end;
38056:
38057: Proc SIRegister_UDict(CL: TPSPascalCompiler);
38058: begin 'dichighletter','String').SetString( 'z;
38059:   SIRegister_TDicForm(CL);
38060:   SIRegister_TDic(CL);
38061: end;
38062:
38063: Proc SetArrayLength2Char2(var arr: T2CharArray; asize1, asize2: Int);
38064: var i: Int;
38065: begin setlength(arr, asize1);
38066:   for i:= 0 to asize1-1 do SetLength(arr[i], asize2);
38067: end;
38068:
38069: Example of dynamic array of panels:
38070: Proc SetArrayLength2Panels(var arr: array of array of TPanel;
38071:   asize1, asize2: Integer);
38072: var i: Integer;
38073: begin setlength(arr, asize1);
38074:   for i:= 0 to asize1-1 do SetLength(arr[i], asize2);
38075: end;
38076:
38077: Proc TMyFormInitialisePanels(aform: TForm; RowCount,ColCount: Integer);
38078: var aLeft,aTop,aWidth,aHeight, row,col: Integer;
38079:   Panel: TPanel;
38080:   FPanels: array of array of TPanel;
38081: begin
38082:   //SetLength(FPanels, RowCount, ColCount);
38083:   SetArrayLength2Panels(FPanels, RowCount, ColCount)
38084:   aTop:= 0;
38085:   for Row:= 0 to RowCount-1 do begin
38086:     aLeft:= 0;
38087:     aHeight:= (aform.ClientHeight-aTop) div (RowCount-Row);

```



```

38088:   for Col:= 0 to ColCount-1 do begin
38089:     Panel:= TPanel.Create(Self);
38090:     FPanels[Row][Col]:= Panel;
38091:     Panel.Parent:= aform; //Self;
38092:     //panel.parentcolor:= false;
38093:     panel.ParentBackground:= false;
38094:     panel.color:= random(clred)
38095:     aWidth:= (aform.ClientWidth-aLeft) div (ColCount-Col);
38096:     Panel.SetBounds(aLeft, aTop, aWidth, aHeight);
38097:     inc2(aLeft, aWidth);
38098:   end;
38099:   inc2(aTop, aHeight);
38100: end;
38101: end;
38102: aform:= TForm.create(self)
38103: aform.setbounds(10,10,600,500);
38104: TMyFormInitialisePanels(aform, 26, 19);
38105: aform.show;
38106:
38107: Proc SIRegister_UP10Build(CL: TPSPascalCompiler);
38108: begin Proc ParseFunction(FunctionString:str; Variables:TStringlist; FunctionOne,
FunctionTwo:TStringList;UsePascalNumbers:bool;var FirstOP:{TObject}POperation;var Error:bool);
38109: end;
38110:
38111: Proc SIRegister_ComObj2 (CL: TPSPascalCompiler);
38112: begin
38113:   ClassN(CL.FindClass('TOBJECT'),'TComObjectFactory;
38114:   SIRegister_TComServerObject(CL);
38115:   SIRegister_ADOConst(CL);
38116:   FieldTypeNames:= array[0..41] of string = (
38117:     Unknown, 'String', 'SmallInt', 'Int', 'Word', 'Boolean', 'Float',
38118:     'Currency', 'BCD', 'Date', 'Time', 'DateTime', 'Bytes', 'VarBytes',
38119:     'AutoInc', 'Blob', 'Memo', 'Graphic', 'FmtMemo', 'ParadoxOle',
38120:     'dBaseOle', 'TypedBinary', 'Cursor', 'FixedChar', 'WideString',
38121:     'LargeInt', 'ADT', 'Array', 'Reference', 'DataSet', 'HugeBlob', 'HugeClob',
38122:     'Variant', 'Interface', 'Dispatch', 'Guid', 'SQLTimeStamp', 'FMTBcdField',
38123:     FixedWideChar, 'WideMemo', 'SQLTimeStamp', 'String';
38124:   TFactoryProc, 'Proc (Factory : TComObjectFactory);
38125:   TCallingConvention', (ccRegister, ccCdecl, ccPascal, ccStdCall, ccSafeCall);
38126:   SIRegister_TComClassManager(CL);
38127:   SIRegister_IServerExceptionHandler(CL); SIRegister_TComObject(CL);
38128:   //TComClass, 'class of TComObject;
38129:   TClassInstantiating', (ciInternal, ciSingleInstance, ciMultiInstance );
38130:   TThreadingModel', (tmSingle, tmApartment, tmFree, tmBoth, tmNeutral );
38131:   SIRegister_TComObjectFactory(CL); SIRegister_TTypedComObject(CL);
38132:   //TTypedComClass, 'class of TTypedComObject;
38133:   SIRegister_TTypedComObjectFactory(CL);
38134:   TConnectEvent2', 'Procedure (const Sink : IUnknown; Connecting :Bool);
38135:   ClassN(CL.FindClass('TOBJECT'),'TAutoObjectFactory;
38136:   SIRegister_TAUTOObject(CL); //TAUTOObject2 ?? in OleAuto and ComObj
38137:   //TAUTOClass, 'class of TAutoObject;
38138:   SIRegister_TAUTOObjectFactory(CL); SIRegister_TAUTOIntfObject(CL);
38139:   //ClassN(CL.FindClass('TOBJECT'),'EOleError;
38140:   ClassN(CL.FindClass('Exception'),'EOleError;
38141:   SIRegister_EOleSysError(CL); SIRegister_EOleException(CL);
38142:   ClassN(CL.FindClass('TOBJECT'),'EOleRegistrationError;
38143:   //Proc DispatchInvoke(const
Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Result: PVariant);
38144:   //Proc DispatchInvokeError( Status : Int; const ExcepInfo : TExcepInfo);
38145:   //Func HandleSafeCallException(ExceptObject:TObject;ExceptAddr:Pointer;const ErrorIID TGUID;const ProgID,
HelpFileName:WideString):HResult;
38146:   Func CreateComObject( const ClassID : TGUID) : IUnknown;
38147:   Func CreateRemoteComObject(const MachineName: WideString;const ClassID:TGUID):IUnknown;
38148:   //Func CreateOleObject(const ClassName :Str): IDispatch;
38149:   //Func GetActiveOleObject(const ClassName:str): IDispatch;
38150:   Proc OleError2( ErrorCode : HResult);
38151:   Proc OleCheck2( Result : HResult);
38152:   Func StringToGUID2( const S :Str) : TGUID;
38153:   Func GUIDToString2( const ClassID : TGUID) :Str;
38154:   Func ProgIDToClassID2( const ProgID :Str) : TGUID;
38155:   Func ClassIDToProgID2( const ClassID : TGUID) :Str;
38156:   Proc CreateRegKey(const Key, ValueName, Value:str;RootKey:DWord);
38157:   Proc DeleteRegKey(const Key :Str; RootKey : DWord);
38158:   Func GetRegStringValue( const Key, ValueName :Str; RootKey : DWord) :Str;
38159:   //Func StringToLPOLESTR( const Source :Str) : PoleStr;
38160:   Proc RegisterComServer( const DLLName :Str);
38161:   Proc RegisterAsService( const ClassID, ServiceName :Str);
38162:   Func CreateClassID2:Str;
38163:   Proc InterfaceConnect(const Srce:IUnknown;const IID:TIID;const Sink:IUnknown;var Connectin:Longint);
38164:   Proc InterfaceDisconnect(const Source:IUnknown;const IID:TIID;var Connection:Longint);
38165:   Func GetDispatchPropValue(Disp: IDispatch; DispID : Int) : OleVariant;;
38166:   Func GetDispatchPropValue1(Disp: IDispatch; Name : WideString) : OleVariant;;
38167:   Proc SetDispatchPropValue2(Disp:IDispatch;DispID:Int;const Value:OleVariant);
38168:   Proc SetDispatchPropValue3(Disp: IDispatch; Name:WideString; const Value:OleVariant);;
38169:   Func ComClassManager : TComClassManager;
38170:   // from ADO DB OLE Utils
38171:   TOleEnum, 'LongWord; //DataTypeEnum = TOleEnum;
38172:   DataTypeEnum, 'TOleEnum;
38173:   Func CreateADOObject( const ClassID : TGUID) : IUnknown;

```

```

38174: Func ADOTypeToFieldType(const ADOType:DataTypeEnum; EnableBCD:Boolean):TFieldType;
38175: Func FieldTypeToADOType(const FieldType: TFieldType):DataTypeEnum;
38176: Func StringToVarArray( const Value :Str) : OleVariant;
38177: Func VarDataSize( const Value : OleVariant) : Int;
38178: Func OleEnumToOrd( OleEnumArray : array of TOleEnum; Value : TOleEnum) : Int;
38179: Func GetStates( State : Int) : TObjectStates;
38180: Func ExecuteOptionsToOrd(ExecuteOptions:TExecuteOptions): Int;
38181: Func OrdToExecuteOptions(Options: Int) : TExecuteOptions;
38182: Func ExtractFieldName( const Fields : WideString; var Pos : Int) : WideString;
38183: Func GetFilterStr( Field : TField; Value : Variant; Partial :Bool) : WideString;
38184: Func FieldListChecksum( DataSet : TDataSet) : Int;
38185: Func GlobalAllocString(s: Ansistr): HGlobal;
38186: Func ScanTime(const S:Str; var Pos: Int; var Time: TDateTime):Bool;
38187: Func ScanChar(const S:Str; var Pos: Int; Ch: Char):Bool;
38188: Func ScanNumber(const S:Str; var Pos: Int; var Number: Word):Bool;
38189: Func ScanString(const S:Str; var Pos: Int; const Symbol:Str):Bool;
38190: Proc LV_InsertFiles(strPath:Str; ListView: TListView; ImageList: TImageList);
38191: Func GetPasteLinkInfo(var Service:Str; var Topic:Str; var Item:Str):Boolean;
38192: Func IPToHostName(const IP:Str):Str;
38193: Proc GetZoneIcon(IconPath:Str; var Icon: TIcon);
38194: Func GetZoneAttributes(const URL:Str): TZoneAttributes;
38195: //unit uPSI_PsAPI;
38196: Proc CGITester;
38197: //CGI will take name and email address from command line and place it into HTML
38198: Proc CreateBrowserOnForm(aform: TForm; aurl:Str);
38199: Proc WebOnForm(aform: TForm; aurl:Str);
38200: Proc WebToForm(aform: TForm; aurl:Str);
38201: Proc SearchAndHighlightWebText(aform: TForm; aurl:Str; aText:Str);
38202: Proc SaveImagesOnWeb(aurl, apath:Str);
38203: Func GetProcessNameFromWnd(Wnd: HWND):Str; //get EXE path from window handle
38204: Func GetAllEvents(aform: TForm): TStringlist;
38205: Proc GetKLList(List: TStrings);
38206: Proc GetKeyboardList(List: TStrings);
38207: Func SetSuspendState(Hibernate:Boolean;ForceCritical:Boolean;DisableWakeEvent:Bool):bool;
38208: call: SetSuspendState(True, False, False);
38209: Func ServiceRunning(sMachine, sService: PChar):Bool;
38210: Func isServiceRunning(sMachine, sService: PChar):Bool;
38211: Proc CloseOpenSockets( Sockets : array of TIdStackSocketHandle);
38212: //SIRegister_SvrHTTPIndy - Linux
38213: Proc TransformError( const Msg :Str);
38214: Proc StringToFile2( const S, FileName :Str);
38215: Func GetXMLData( DataSet : TClientDataSet) :Str;
38216: Proc EditComTerminal( ComTerminal : TCustomComTerminal); //TComTrmSetForm
38217: Func search_adapter_key_networkcard:Str; //at registry
38218: Func getNetworkCard:Str;
38219: Func GetMacAddresses2(const Machine:Str; const Addresses: TStrings): Int;
38220: Func ConnectDrive(_drvLetter:Str;_netPath:Str;_showError:Bool;_reconnect:Bool):DWORD;
38221: Func ConnectPrinterDevice(_lptPort:Str;_netPath:Str;_showError:Bool;_reconnect:Bool):DWORD;
38222: Func DisconnectNetDrive(_locDrive:Str;_showError:Bool;_force:Bool;_save:Bool): DWORD;
38223: //ConnectDrive('k:', '\\Servername\C', True, True);
38224: //DisconnectNetDrive('k:', True, True, True);
38225: Func GetConnectionKind(var strKind:Str):Bool; MODEM=1; LAN=2; PROXY=4; BUSY= 8;
38226: Func DownloadJPGToBitmap(const URL :Str; ABitmap: TBitmap):Bool;
38227: Proc GetImageLinks(AURL:Str; AList: TStrings);
38228: Func GetCharEncoding( alias :Str; var _name :Str) : Int;
38229: Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
38230: Func MicrosoftLangageCodeToISOCode( langcode : Int) :Str;
38231: Proc CopyHTMLToClipboard(const str:Str; const htmlStr:Str = '');
38232: Func RFC1123ToDateTime(Date:Str): TDateTime;
38233: Func DateTimeToRFC1123(aDate: TDateTime):Str;
38234: Proc CopyHTMLToClipboard(const str:Str; const htmlStr:Str);
38235: Proc DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);
38236: Proc DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);
38237: Proc DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);
38238:
38239: IPToHostName
38240: //from ADOInt.pas // TOleEnum = type LongWord;
38241: TypeS('CursorOptionEnum', 'TOleEnum;
38242: Interface(CL.FindInterface('IUNKNOWN'),_Recordset, '_Recordset;
38243: //TypeS('_RecordsetDisp', 'dispinterface;
38244: Interface(CL.FindInterface('IUNKNOWN'),_Command, '_Command;
38245: Interface(CL.FindInterface('IUNKNOWN'),_Connection, '_Connection;
38246: // SIRegister_Recordset(CL); // SIRegister_Command(CL);
38247: Const IID_Recordset20,'TGUID').SetString('{0000054F-0000-0010-8000-00AA006D2EA4}');
38248: 'IID_Recordset','string').SetString( '00000555-0000-0010-8000-00AA006D2EA4;
38249: //test with stringToGUID
38250: // 'CLASS_Command','TGUID').SetString( '{00000507-0000-0010-8000-00AA006D2EA4};
38251: // 'CLASS_Recordset','TGUID').SetString( '{00000535-0000-0010-8000-00AA006D2EA4};
38252: { TypeS('Connection', '_Connection;
38253: TypeS('Command', '_Command; TypeS('Recordset', '_Recordset;
38254: TypeS('Parameter', '_Parameter; TypeS('DataSpace', '_IDataspace;
38255: TypeS('SearchDirection', 'SearchDirectionEnum; }
38256: //TypeS('Command', '_Command; //TypeS('Recordset', '_Recordset;
38257: end;
38258:
38259: Proc SIRegister_DIUtils(CL: TPSPascalCompiler);
38260: begin
38261: // 'CRLF','string').SetString( '$0D$0A);
38262: 'REPLACEMENT_CHARACTER','LongWord').SetUInt( $FFFD);

```

```

38263: 'HANGUL_SBase','LongWord').SetUInt( $AC00);
38264: 'HANGUL_LBase','LongWord').SetUInt( $1100);
38265: 'HANGUL_VBase','LongWord').SetUInt( $1161);
38266: 'HANGUL_TBase','LongWord').SetUInt( $11A7);
38267: 'HANGUL_LCount','LongInt').SetInt( 19);
38268: 'HANGUL_VCount','LongInt').SetInt( 21);
38269: 'HANGUL_TCount','LongInt').SetInt( 28);
38270: 'KEY_WOW64_32KEY','LongWord').SetUInt( $0200);
38271: 'KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
38272: 'KEY_WOW64_RES','LongWord').SetUInt( $0300);
38273: TypeS('TAnsiCharSet', 'set of AnsiChar;
38274: TypeS('TISODate', 'Cardinal;
38275: TypeS('TJulianDate', 'Int;
38276: //TypeS('PJulianDate', '^TJulianDate // will not work;
38277: TypeS('TValidateCharFunc', 'function(const c: Char):Bool;;
38278: // TValidateCharFunc = function( const c: Char):Bool;
38279: TypeS('TProcedureEvent', 'Procedure;
38280: 'MT19937_N','LongInt').SetInt( 624);
38281: 'MT19937_M','LongInt').SetInt( 397);
38282: SIRegister_TMT19937(CL);
38283: //SIRegister_TWIDEStrBuf(CL);
38284: TypeS('TDITextLineBreakStyle', '( tlbsLF, tlbsCRLF, tlbsCR );
38285: Func AdjustLineBreaksW(const s:UnicodeStr;const Style:TDITextLineBreakStyle):UnicodeStr;
38286: Func BrightenColor( const Color : Int; const amount : Byte) : Int;
38287: Func BSwap4( const Value :Card) :Card;;
38288: Func BSwap5( const Value : Int) : Int;;
38289: //Func BufCompNumIW(p1:PWideChar; l1:Int; p2 : PWideChar; l2 : Int) : Int;
38290: Func BufSameA( p1, p2 : PChar; l :Card) :Bool;
38291: // Func BufSameW( p1, p2 : PWideChar; l :Card) :Bool;
38292: Func BufSameIA( p1, p2 : PChar; l :Card) :Bool;
38293: //Func BufSameIW( p1, p2 : PWideChar; l :Card) :Bool;
38294: Func BufPosCharA(const Buf:PChar;l:Card;const c:AnsiChar;const Start:Card):Int;
38295: Func BufPosCharsA(const Buf:PChar;l:Card;const Search:TAnsiCharSet;const Start:Card):Int;
38296: Func BufStrSame(const Buf:PChar;const BufCharCount:Card;const s:Str):Bool;
38297: Func BufStrSameA(const Buf:PChar; const BufCharCount:Card;const s:RawByteString):Bool;
38298: Func BufStrSameI(const Buf:PChar; const BufCharCount:Card; const s :Str):Bool;
38299: Func BufStrSameIA(const Buf:PChar;const BufCharCount:Card;const s:RawByteString): Bool;
38300: Func diChangeFileExt(const FileName,Extension:Str):Str;
38301: Func ChangeFileExtA( const FileName, Extension : Ansistr) : Ansistr;
38302: Func ChangeFileExtW( const FileName, Extension : UnicodeString) : UnicodeString;
38303: //Func CharDecomposeCanonicalW( const c : WideChar) : PCharDecompositionW;
38304: Func CharDecomposeCanonicalStrW( const c : WideChar) : UnicodeString;
38305: //Func CharDecomposeCompatibleW( const c : WideChar) : PCharDecompositionW;
38306: Func CharDecomposeCompatibleStrW( const c : WideChar) : UnicodeString;
38307: Func CharIn8( const c, t1, t2 : WideChar) :Bool;;
38308: Func CharIn9( const c, t1, t2, t3 : WideChar) :Bool;;
38309: Proc ConCatBuf(const Buffer: PChar;const CharCount:Card;var d:str;var InUse:Card);
38310: Proc ConCatBufA(const Buff:PChar;const AnsiCharCount:Card;var d:RawByteStr;var InUse:Card);
38311: Proc ConCatChar( const c : Char; var d :Str; var InUse :Card);
38312: Proc ConCatCharA( const c : AnsiChar; var d : RawByteString; var InUse :Card);
38313: // Proc ConCatCharW( const c : WideChar; var d : UnicodeString; var InUse :Card);
38314: Proc ConCatStr( const s :Str; var d :Str; var InUse :Card);
38315: Proc ConCatStrA( const s : RawByteString; var d : RawByteString; var InUse :Card);
38316: //Proc ConCatStrW(const w:UnicodeString; var d : UnicodeString; var InUse :Card);
38317: Func diCountBitsSet( const Value : Int) : Byte;
38318: //Func Crc32OfStrA( const s : RawByteString) : TCrc32;
38319: //Func Crc32OfStrW( const s : UnicodeString) : TCrc32;
38320: Func CurrentDay : Word;
38321: Func CurrentJulianDate : TJulianDate;
38322: Func CurrentMonth : Word;
38323: Func CurrentQuarter : Word;
38324: Func diCurrentYear : Int;
38325: Func DarkenColor( const Color : Int; const amount : Byte) : Int;
38326: Func diDeleteFile( const FileName :Str) :Bool;
38327: Func DeleteFileA( const FileName : Ansistr) :Bool;
38328: //Func DeleteFileW( const FileName : UnicodeString) :Bool;
38329: Func diDirectoryExists( const Dir :Str) :Bool;
38330: Func DirectoryExistsA( const Dir : Ansistr) :Bool;
38331: //Func DirectoryExistsW( const Dir : UnicodeString) :Bool;
38332: Func diDiskFree( const Dir :Str) : Int64;
38333: Func DiskFreeA( const Dir : Ansistr) : Int64;
38334: //Func DiskFreeW( const Dir : UnicodeString) : Int64;
38335: Func diExpandFileName( const FileName :Str) :Str;
38336: Func ExpandFileNameA(const FileName: Ansistr): Ansistr;
38337: //Func ExpandFileNameW( const FileName : UnicodeString) : UnicodeString;
38338: Proc diExcludeTrailingPathDelimiter( var s :Str);
38339: Proc ExcludeTrailingPathDelimiterA( var s : RawByteString);
38340: //Proc ExcludeTrailingPathDelimiterW( var s : UnicodeString);
38341: Func diExtractFileDrive( const FileName :Str) :Str;
38342: Func ExtractFileDriveA( const FileName : RawByteString) : RawByteString;
38343: //Func ExtractFileDriveW( const FileName : UnicodeString) : UnicodeString;
38344: Func diExtractFileExt( const FileName :Str) :Str;
38345: Func ExtractFileExtA( const FileName : RawByteString) : RawByteString;
38346: //Func ExtractFileExtW( const FileName : UnicodeString) : UnicodeString;
38347: Func diExtractFileName( const FileName :Str) :Str;
38348: Func ExtractFileNameA(const FileName: Ansistr): Ansistr;
38349: //Func ExtractFileNameW( const FileName : UnicodeString) : UnicodeString;
38350: Func diExtractFilePath( const FileName :Str) :Str;
38351: Func ExtractFilePathA( const FileName : RawByteString) : RawByteString;

```

```

38352: //Func ExtractFilePathW( const FileName : UnicodeString) : UnicodeString;
38353: Func ExtractNextWord10(const s:str;const ADelimiter:Char;var AStartIndex:Int):str;;
38354: Func ExtractNextWordAll(const s:RawByteStr;const ADelimiter:AnsiChar;var AStartIndex:Int): RawByteStr;
38355: Func ExtractNextWordW12(const s:UnicodeStr;const ADelimiter:WideChar;var AStartIndex:Int): UnicodeStr;
38356: Func ExtractNextWord13(const s:str;const ADelims:TAnsiCharSet;var AStartIndex:Int): str;
38357: Func ExtractNextWordA14(const s:RawByteString;const ADelimits:TAnsiCharSet;var
AStartIndex:Int):RawByteString;
38358: Func diExtractWord(const Number:Card;const s:RawByteString;const Delimiters:TAnsiCharSet): RawByteString;
38359: Func ExtractWordA const Numb:Card;const s RawByteStr;const Delimiters:TAnsiCharSet): RawByteStr;
38360: Func ExtractWordStartsA(const s:RawByteString;const MaxCharCount:Card;const WordSeparators:TAnsiCharSet):
RawByteString;
38361: Func diFileExists( const FileName :Str) :Bool;
38362: Func FileExistsA( const FileName : Ansistr) :Bool;
38363: //Func FileExistsW( const FileName : UnicodeString) :Bool;
38364: Func diGCD( x, y :Card) :Card;
38365: Func diGetTempFolder :Str;
38366: Func GetTempFolderA : Ansistr;
38367: //Func GetTempFolderW : UnicodeString;
38368: Func diGetUserName( out UserName :Str) :Bool;
38369: Func GetUserNameA( out UserName : Ansistr) :Bool;
38370: //Func GetUserNameW( out UserName : UnicodeString) :Bool;
38371: Func HexCodePointToInt( const c :Card) : Int;
38372: Func diHexToInt( const s :Str) : Int;
38373: Func HexToIntA( const s : RawByteString) : Int;
38374: //Func HexToIntW( const s : UnicodeString) : Int;
38375: Func BufHexToInt( p : PChar; l :Card) : Int;
38376: Func BufHexToIntA( p : PChar; l :Card) : Int;
38377: //Func BufHexToIntW( p : PWideChar; l :Card) : Int;
38378: Proc IncludeTrailingPathDelimiterByRef( var s :Str);
38379: Proc IncludeTrailingPathDelimiterByRefA(var s:RawByteString);
38380: //Proc IncludeTrailingPathDelimiterByRefW(var w:UnicodeString);
38381: Func IntToHex16(const Value: Int; const Digits : NativeInt):Str;;
38382: Func IntToHex17(const Value: Int64; const Digits : NativeInt) :Str;;
38383: Func IntToHex18(const Value: UInt64; const Digits : NativeInt) :Str;;
38384: Func IntToHexA( Value : UInt64; const Digits : NativeInt) : RawByteString;
38385: //Func IntToHexW(Value : UInt64; const Digits : NativeInt) : UnicodeString;
38386: Func IntToStrA19(const i: Int) : RawByteString;;
38387: Func IntToStrW20(const i: Int) : UnicodeString;;
38388: Func IntToStrA21(const i: Int64) : RawByteString;;
38389: Func IntToStrW22(const i: Int64) : UnicodeString;;
38390: Func CharDecomposeHangulW(const c: WideChar) : UnicodeString;
38391: Func diIsPathDelimiter( const s :Str; const Index :Card) :Bool;
38392: Func IsPathDelimiterA( const s : RawByteString; const Index :Card) :Bool;
38393: //Func IsPathDelimiterW( const s : UnicodeString; const Index :Card) :Bool;
38394: Func IsPointInRect( const Point : TPoint; const Rect : TRect) :Bool;
38395: Func JulianDateToIsoDateStr(const Julian: TJulianDate):str;
38396: Func JulianDateToIsoDateStrA( const Julian : TJulianDate) : RawByteString;
38397: //Func JulianDateToIsoDateStrW( const Julian : TJulianDate) : UnicodeString;
38398: Func LeftMostBit( Value :Card) : ShortInt;;
38399: Func LeftMostBit2( Value : UInt64) : ShortInt;;
38400: //Func MakeMethod( const AData, ACode : Pointer) : TMethod;
38401: Func StrIsEmpty( const s :Str) :Bool;
38402: Func StrIsEmptyA( const s : RawByteString) :Bool;
38403: //Func StrIsEmptyW( const s : UnicodeString) :Bool;
38404: Func PadLeftA(const Source:RawByteString;const Count:Card;const c:AnsiChar):RawByteStr;
38405: //Func PadLeftW(const Source:UnicodeString; const Count :Card; const c : WideChar): UnicodeString;
38406: Func PadRightA(const Source:RawByteString;const Count:Card;const c:AnsiChar):RawByteStr;
38407: //Func PadRightW(const Source:UnicodeString;const Count:Card;const c WideChar):UnicodeStr;
38408: Func ProperCase( const s :Str) :Str;
38409: Func ProperCaseA( const s : RawByteString) : RawByteString;
38410: //Func ProperCaseW( const s : UnicodeString) : UnicodeString;
38411: Proc ProperCaseByRefA( var s : RawByteString);
38412: //Proc ProperCaseByRefW( var s : UnicodeString);
38413: Func RegReadRegisteredOrganization( const Access : REGSAM) :Str;
38414: Func RegReadRegisteredOrganizationA( const Access : REGSAM) : Ansistr;
38415: //Func RegReadRegisteredOrganizationW( const Access : REGSAM) : UnicodeString;
38416: Func RegReadRegisteredOwner( const Access : REGSAM) :Str;
38417: Func RegReadRegisteredOwnerA( const Access : REGSAM) : Ansistr;
38418: //Func RegReadRegisteredOwnerW( const Access : REGSAM) : UnicodeString;
38419: Func RegReadStrDef(const Key:HKEY; const SubKey:Str;const ValueName:Str;const Default:str;const
Access:REGSAM):Str;
38420: Func RegReadStrDefA(const Key:HKEY;const SubKey:Ansistr;const ValueName:Ansistr;const Default:Ansistr;
const Access:REGSAM):Ansistr;
38421: Func StrDecodeUrlA( const Value : RawByteString) : RawByteString;
38422: Func StrEncodeUrlA( const Value : RawByteString) : RawByteString;
38423: Func diStrEnd( const s : PChar) : PChar;
38424: Func StrEndA( const s : PChar) : PChar;
38425: //Func StrEndW( const s : PWideChar) : PWideChar;
38426: Proc StrIncludeTrailingChar( var s :Str; const c : Char);
38427: Proc StrIncludeTrailingCharA( var s : RawByteString; const c : AnsiChar);
38428: //Proc StrIncludeTrailingCharW( var s : UnicodeString; const c : WideChar);
38429: Func diStrLen( const s : PChar) : NativeUInt;
38430: Func StrLenA( const s : PChar) : NativeUInt;
38431: //Func StrLenW( const s : PWideChar) : NativeUInt;
38432: Func StrRandom(const ASeed:RawByteStr;const ACharacters:str;const ALength:Card):str;;
38433: Func StrRandomA(const ASeed:RawByteString;const ACharacters:RawByteString;const ALength:Card):RawByteStr;
38434: Proc StrRemoveFromToIA(var Source:RawByteString;const FromString, ToString:RawByteString);
38435: //Proc StrRemoveFromToIW(var Source:UnicodeString;const FromString, ToString : UnicodeString);
38436: Proc StrRemoveSpacingA(var s:RawByteStr;const SpaceChars:TAnsiCharSet;const ReplaceChar:AnsiChar);

```



```

38437: Proc diStrReplaceChar(var Source:Str;const SearchChar,ReplaceChar:Char);
38438: Proc StrReplaceChar8(var s:Utf8String;const SearchChar,ReplaceChar:AnsiChar);
38439: Proc StrReplaceCharA(var s:RawByteString;const SearchChar,ReplaceChar: AnsiChar);
38440: //Proc StrReplaceCharW(var s:UnicodeString; const SearchChar, ReplaceChar : WideChar);
38441: Func diStrReplace( const Source, Search, Replace :Str) :Str;
38442: Func StrReplaceA( const Source, Search, Replace : RawByteString) : RawByteString;
38443: //Func StrReplaceW( const Source, Search, Replace : UnicodeString) : UnicodeString;
38444: Func StrReplaceI( const Source, Search, Replace :Str) :Str;
38445: Func StrReplaceIA( const Source, Search, Replace : RawByteString) : RawByteString;
38446: //Func StrReplaceIW( const Source, Search, Replace : UnicodeString) : UnicodeString;
38447: Func StrReplaceLoopA( const Source, Search, Replace : RawByteString) : RawByteString;
38448: //7Func StrReplaceLoopW( const Source, Search, Replace: UnicodeString) : UnicodeString;
38449: Func StrReplaceLoopIA( const Source, Search, Replace : RawByteString) : RawByteString;
38450: //Func StrReplaceLoopIW( const Source, Search, Replace: UnicodeString) : UnicodeString;
38451: Func RightMostBit( const Value :Card) : ShortInt;;
38452: Func RightMostBit2( const Value : UInt64) : ShortInt;;
38453: Func LoadStrFromFile(const FileName:str; var s:RawByteString):Boolean;;
38454: Func FileToStr( const FileName :Str; var s :Str) :Bool;;
38455: Func LoadStrAFromFileA(const FileName:Ansistr;var s:RawByteString):Boolean;
38456: //Func LoadStrAFromFileW(const FileName:UnicodeString; var s: RawByteString) :Bool;
38457: Func LoadStrWFromFile28(const FileName:str;var s:UnicodeString):Boolean;;
38458: Func LoadStrWFromFileA(const FileName: Ansistr;var s:UnicodeString):Bool;
38459: //Func LoadStrWFromFileW(const FileName:UnicodeString;var s:UnicodeString):Bool;
38460: Func QuotedStrW( const s : UnicodeString; const Quote : WideChar) : UnicodeString;
38461: Func SaveStrToFile( const s :Str; const FileName :Str) :Bool;
38462: Func StrToFile( const s :Str; const FileName :Str) :Bool;
38463: Func SaveStrAToFile(const s:RawByteString;const FileName :Str) :Bool;
38464: Func SaveStrAToFileA( const s : RawByteString; const FileName : Ansistr) :Bool;
38465: //Func SaveStrAToFileW(const s:RawByteString; const FileName: UnicodeString) :Bool;
38466: Func SaveStrWToFile(const s:UnicodeString;const FileName :Str):Boolean;
38467: Func SaveStrWToFileA(const s: UnicodeString;const FileName:Ansistr) : Bool;
38468: //Func SaveStrWToFileW(const s:UnicodeString; const FileName : UnicodeString):Bool;
38469: Func StrPosChar(const Source:str; const c:Char;const Start:Card):Card;
38470: Func StrPosCharA(const Source:RawByteString;const c:AnsiChar;const Start:Card):Card;
38471: //Func StrPosCharW(const Source:UnicodeStr;const c:WideChar;const Start:Cardi):Card;
38472: Func StrPosCharBack(const Source:Str;const c:Char;const Start:Card):Card;
38473: Func StrPosCharBackA(const Source:RawByteStr;const c:AnsiChar;const Start:Cardi):Card);
38474: Func StrPosCharsA(const Source:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
38475: Func StrPosCharsBackA(const Src:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
38476: Func StrPosNotCharsA(const Src:RawByteStr;const Search:TAnsiCharSet;const Start:Cardi):Cardi;
38477: Func StrPosNotCharsBackA(const Source:RawByteString;const Search:TAnsiCharSet; const Start:Card):Card;
38478: Func SetFileDate(const FileHandle:THandle;const Year:Int;const Month,Day:Word):Bool);
38479: Func SetFileDate2( const FileName :Str; const JulianDate : TJulianDate) :Bool;
38480: Func SetFileDateA( const FileName: Ansistr; const JulianDate:TJulianDate) :Bool;
38481: //Func SetFileDateW(const FileName:UnicodeString;const JulianDate:TJulianDate):Boolean;
38482: Func SetFileDateYmd(const FileName:str;const Year:Int;const Month,Day:Word):Bool;
38483: Func SetFileDateYmdA(const FileName:Ansistr;const Year:Int;const Month,Day:Word):Bool;
38484: Func SetFileDateYmdW(const FileName:UnicodeStr;const Year:Int;const Month,Day:Word):Bool;
38485: Func StrContainsChar(const s:str; const c:Char; const Start :Card) :Bool;
38486: Func StrContainsCharA(const s:RawByteString;const c:AnsiChar;const Start:Card):Bool;
38487: //Func StrContainsCharW(const s:UnicodeString;const c:WideChar;const Start:Cardi):Bool;
38488: Func StrContainsCharsA(const s:RawByteStr;const Chars:TAnsiCharSet;const Start:Card):Bool;
38489: Func diStrSame( const s1, s2 :Str) :Bool;
38490: Func StrSameA( const s1, s2 : RawByteString) :Bool;
38491: //Func StrSameW( const s1, s2 : UnicodeString) :Bool;
38492: Func StrSameI( const s1, s2 :Str) :Bool;
38493: Func StrSameIA( const s1, s2 : RawByteString) :Bool;
38494: //Func StrSameIW( const s1, s2 : UnicodeString) :Bool;
38495: Func StrSameStart( const s1, s2 :Str) :Bool;
38496: Func StrSameStartA( const s1, s2 : RawByteString) :Bool;
38497: //Func StrSameStartW( const s1, s2 : UnicodeString) :Bool;
38498: Func StrSameStartI( const s1, s2 :Str) :Bool;
38499: Func StrSameStartIA( const s1, s2 : RawByteString) :Bool;
38500: //Func StrSameStartIW( const s1, s2 : UnicodeString) :Bool;
38501: Func diStrComp( const s1, s2 :Str) : Int;
38502: Func StrCompA( const s1, s2 : RawByteString) : Int;
38503: //Func StrCompW( const s1, s2 : UnicodeString) : Int;
38504: Func StrCompI( const s1, s2 :Str) : Int;
38505: Func StrCompIA( const s1, s2 : RawByteString) : Int;
38506: //Func StrCompIW( const s1, s2 : UnicodeString) : Int;
38507: Func StrCompNum( const s1, s2 :Str) : Int;
38508: Func StrCompNumA( const s1, s2 : RawByteString) : Int;
38509: // Func StrCompNumW( const s1, s2 : UnicodeString) : Int;
38510: Func StrCompNumI( const s1, s2 :Str) : Int;
38511: Func StrCompNumIA( const s1, s2 : RawByteString) : Int;
38512: // Func StrCompNumIW( const s1, s2 : UnicodeString) : Int;
38513: Func StrContains(const Search,Source:str; const Start:Card):Bool;
38514: Func StrContainsA(const Search,Source:RawByteString;const Start:Card):Bool;
38515: //Func StrContainsW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Bool;
38516: Func StrContainsI(const Search, Source :Str; const Start :Card) :Bool;
38517: Func StrContainsIA(const Search,Source:RawByteString; const Start:Card):Bool;
38518: //Func StrContainsIW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Bool;
38519: Func StrCountChar(const ASource:str;const c:Char;const AStartIdx:Card):Card;
38520: Func StrCountCharA(const ASource:RawByteStr;const c:AnsiChar;const AStartIdx:Cardi):Card;
38521: Func StrMatchesA(const Search,Source:RawByteString; const AStartIdx:Card):Bool;
38522: Func StrMatchesIA(const Search,Source:RawByteString;const AStartIdx:Card):Bool;
38523: Func StrMatchWild(const Source,Mask:str;const WildChar:Char;const MaskChar:Char):Bool;
38524: Func StrMatchWildA(const Src,Mask:RawByteString;const WildChr:AnsiChar;const MaskChr:AnsiChar):Bool;
38525: Func StrMatchWildI(const Source,Mask:str;const WildChar:Char;const MaskChar:Char):Bool;

```



```

38526: Func StrMatchWildIA(const Source,Mask:RawByteString;const WildChar:AnsiChar;const MaskChar:AnsiChar):Bool;
38527: Func diStrPos( const ASearch, ASource :Str; const AStartPos :Card) :Card;
38528: Func StrPosA(const ASearch,ASource:RawByteString;const AStartPos:Card):Card;
38529: //Func StrPosW(const ASearch,ASource:UnicodeString;const AStartPos:Card):Card;
38530: Func StrPosI( const ASearch, ASource :Str; const AStartPos :Card) :Card;
38531: Func StrPosIA(const ASearch,ASource:RawByteString; const AStartPos:Card):Card;
38532: //Func StrPosIW(const ASearch, ASource:UnicodeString;const AStartPos:Card):Card;
38533: Func StrPosBackA( const ASearch, ASource: RawByteString; AStart :Card) :Card;
38534: Func StrPosBackIA(const ASearch, ASource: RawByteString; AStart :Card) :Card;
38535: //Func StrToIntDefW( const w : UnicodeString; const Default : Int) : Int;
38536: Func StrToInt64DefW( const w : UnicodeString; const Default : Int64) : Int64;
38537: Func StrToUpper( const s :Str) :Str;
38538: Func StrToUpperA( const s : RawByteString) : RawByteString;
38539: //Func StrToUpperW( const s : UnicodeString) : UnicodeString;
38540: Proc StrToUpperInPlace( var s :Str);
38541: Proc StrToUpperInPlaceA( var s : Ansistr);
38542: //Proc StrToUpperInPlaceW31( var s : WideString);;
38543: //Proc StrToUpperInPlaceW32( var s : UnicodeString);;
38544: Func StrToLower( const s :Str) :Str;
38545: Func StrToLowerA( const s : RawByteString) : RawByteString;
38546: //Func StrToLowerW( const s : UnicodeString) : UnicodeString;
38547: Proc StrToLowerInPlace( var s :Str);
38548: Proc StrToLowerInPlaceA( var s : Ansistr);
38549: //Proc StrToLowerInPlaceW33( var s : WideString);;
38550: //Proc StrToLowerInPlaceW34( var s : UnicodeString);;
38551: Proc StrTimUriFragmentA( var Value : RawByteString);
38552: //Proc StrTrimUriFragmentW( var Value : UnicodeString);
38553: //Func StrExtractUriFragmentW( var Value : UnicodeString): UnicodeString;
38554: Func StrCountUtf8Chars(const AValue: Utf8String) :Card;
38555: Func StrDecodeUtf8( const AValue: Utf8String): UnicodeString;
38556: Func StrEncodeUtf8( const AValue: UnicodeString): Utf8String;
38557: Func diSysErrorMessage( const MessageID :Card) :Str;
38558: Func SysErrorMessageA(const MessageID:Card): Ansistr;
38559: //Func SysErrorMessageW( const MessageID :Card) : UnicodeString;
38560: Func TextExtentW( const DC : HDC; const Text : UnicodeString) : TSize;
38561: Func TextHeightW( const DC : HDC; const Text : UnicodeString) : Int;
38562: Func TextWidthW( const DC : HDC; const Text : UnicodeString) : Int;
38563: Func diStrTrim( const Source :Str) :Str;
38564: Func StrTrimA( const Source : RawByteString) : RawByteString;
38565: //Func StrTrimW( const w : UnicodeString) : UnicodeString;
38566: Func StrTrimCharA(const Source:RawByteString;const CharToTrim:AnsiChar): RawByteString;
38567: Func StrTrimCharsA(const Source:RawByteStr;const CharsToTrim:TAnsiCharSet):RawByteString;
38568: //Func StrTrimCharsW(const s:UnicodeStr;const IsCharToTrim:TValidateCharFuncW):UnicodeStr;
38569: Proc TrimLeftByRefA( var s : RawByteString; const Chars : TAnsiCharSet);
38570: Func TrimRightA( const Source : RawByteString; const s : TAnsiCharSet) : RawByteString;
38571: Proc TrimRightByRefA( var Source : RawByteString; const s : TAnsiCharSet);
38572: Proc StrTrimCompressA(var s:RawByteString;const TrimCompressChars:TAnsiCharSet;const
ReplaceChar:AnsiChar);
38573: Func TryStrToIntW( const w : UnicodeString; out Value : Int) :Bool;
38574: Func TryStrToInt64W( const w : UnicodeString; out Value : Int64) :Bool;
38575: Func ValInt( const p : PChar; const l : Int; out Code : Int) : Int;;
38576: Func ValIntA36( p : PChar; l : Int; out Code : Int) : Int;;
38577: //Func ValIntW37( p : PWideChar; l : Int; out Code : Int) : Int;;
38578: Func ValInt2(const s:Str; out Code : Int) : Int;;
38579: Func ValIntA39( const s : RawByteString; out Code : Int) : Int;;
38580: Func ValIntW40( const s : UnicodeString; out Code : Int) : Int;;
38581: Func ValInt64A41( p : PChar; l : Int; out Code : Int) : Int64;;
38582: //Func ValInt64W42( p : PWideChar; l : Int; out Code : Int) : Int64;;
38583: Func ValInt64A43( const s : RawByteString; out Code : Int) : Int64;;
38584: Func ValInt64W44( const s : UnicodeString; out Code : Int) : Int64;;
38585: Func YmdToIsoDateStr(constYear:Int;const Month:Word;const Day:Word):str;
38586: Func YmdToIsoDateStrA(const Year:Int;const Month:Word; const Day:Word):RawByteString;
38587: Func YmdToIsoDateStrW(const Year:Int;const Month:Word; const Day:Word):UnicodeString;
38588: Func CharIsLetterW( const c : WideChar) :Bool;
38589: Func CharIsLetterCommonW( const c : WideChar) :Bool;
38590: Func CharIsLetterUpperCaseW(const c : WideChar) :Bool;
38591: Func CharIsLetterLowerCaseW(const c : WideChar) :Bool;
38592: Func CharIsLetterTitleCaseW(const c : WideChar) :Bool;
38593: Func CharIsLetterModifierW(const c : WideChar) :Bool;
38594: Func CharIsLetterOtherW(const c : WideChar) :Bool;
38595: Func CharIsMarkW( const c : WideChar) :Bool;
38596: Func CharIsMarkNon_SpacingW( const c : WideChar) :Bool;
38597: Func CharIsMarkSpacing_CombinedW(const c: WideChar):Bool;
38598: Func CharIsMarkEnclosingW( const c : WideChar) :Bool;
38599: Func CharIsNumberW( const c : WideChar) :Bool;
38600: Func CharIsNumber_DecimalW( const c : WideChar) :Bool;
38601: Func CharIsNumber_LetterW( const c : WideChar) :Bool;
38602: Func CharIsNumber_OtherW( const c : WideChar) :Bool;
38603: Func CharIsPunctuationW( const c : WideChar) :Bool;
38604: Func CharIsPunctuation_ConnectorW( const c : WideChar) :Bool;
38605: Func CharIsPunctuation_DashW( const c : WideChar) :Bool;
38606: Func CharIsPunctuation_OpenW( const c : WideChar) :Bool;
38607: Func CharIsPunctuation_CloseW( const c : WideChar) :Bool;
38608: Func CharIsPunctuation_InitialQuoteW( const c : WideChar) :Bool;
38609: Func CharIsPunctuation_FinalQuoteW( const c : WideChar) :Bool;
38610: Func CharIsPunctuation_OtherW( const c : WideChar) :Bool;
38611: Func CharIsSymbolW( const c : WideChar) :Bool;
38612: Func CharIsSymbolMathW( const c : WideChar) :Bool;
38613: Func CharIsSymbolCurrencyW( const c : WideChar) :Bool;

```

```

38614: Func CharIsSymbolModifierW( const c : WideChar ) :Bool;
38615: Func CharIsSymbolOtherW( const c : WideChar ) :Bool;
38616: Func CharIsSeparatorW( const c : WideChar ) :Bool;
38617: Func CharIsSeparatorSpaceW( const c : WideChar ) :Bool;
38618: Func CharIsSeparatorLineW( const c : WideChar ) :Bool;
38619: Func CharIsSeparatorParagraphW( const c : WideChar ) :Bool;
38620: Func CharIsOtherW( const c : WideChar ) :Bool;
38621: Func CharIsOtherControlW( const c : WideChar ) :Bool;
38622: Func CharIsOtherFormatW( const c : WideChar ) :Bool;
38623: Func CharIsOtherSurrogateW( const c : WideChar ) :Bool;
38624: Func CharIsOtherPrivateUseW( const c : WideChar ) :Bool;
38625: Func BitClear( const Bits, BitNo : Int ) : Int;
38626: Func BitSet( const Bits, BitIndex : Int ) : Int;
38627: Func BitSetTo( const Bits, BitIndex : Int, const Value :Bool ) : Int;
38628: Func BitTest( const Bits, BitIndex : Int ) :Bool;
38629: Func CharCanonicalCombiningClassW( const Char : WideChar ) :Card;
38630: Func CharIsAlphaW( const c : WideChar ) :Bool;
38631: Func CharIsAlphaNumW( const c : WideChar ) :Bool;
38632: Func CharIsCrLf( const c : Char ) :Bool;
38633: Func CharIsCrLfA( const c : AnsiChar ) :Bool;
38634: //Func CharIsCrLfW( const c : WideChar ) :Bool;
38635: Func diCharIsDigit( const c : Char ) :Bool;
38636: Func CharIsDigitA( const c : AnsiChar ) :Bool;
38637: //Func CharIsDigitW( const c : WideChar ) :Bool;
38638: Func CharIsHangulW( const Char : WideChar ) :Bool;
38639: Func CharIsHexDigitW( const c : WideChar ) :Bool;
38640: Func CharIsWhiteSpaceW( const c : WideChar ) :Bool;
38641: Func CharToCaseFoldW( const Char : WideChar ) : WideChar;
38642: Func CharToLowerW( const Char : WideChar ) : WideChar;
38643: Func CharToUpperW( const Char : WideChar ) : WideChar;
38644: Func CharToTitleW( const Char : WideChar ) : WideChar;
38645: Func DayOfJulianDate( const JulianDate : TJulianDate ) : Word;
38646: Func diDayOfWeek( const JulianDate : TJulianDate ) : Word;
38647: Func DayOfWeekYmd( const Year : Int, const Month, Day : Word ) : Word;
38648: Func diDaysInMonth( const JulianDate : TJulianDate ) : Word;
38649: Func DaysInMonthYm( const Year : Int, const Month : Word ) : Word;
38650: Proc DecDay( var Year : Int, var Month, Day : Word);
38651: Proc DecDays( var Year : Int, var Month, Day : Word, const Days : Int);
38652: Func diDeleteDirectory( const Dir :Str, const DeleteItself :Bool ) :Bool;
38653: Func DeleteDirectoryA( Dir : Ansistr, const DeleteItself :Bool ) :Bool;
38654: //Func DeleteDirectoryW( Dir : UnicodeString, const DeleteItself :Bool ) :Bool;
38655: Func diEasterSunday( const Year : Int ) : TJulianDate;
38656: Proc EasterSundayYmd( const Year : Int, out Month, Day : Word);
38657: Func diFirstDayOfWeek( const JulianDate : TJulianDate ) : TJulianDate;
38658: Proc FirstDayOfWeekYmd( var Year : Int, var Month, Day : Word);
38659: Func diFirstDayOfMonth( const Julian : TJulianDate ) : TJulianDate;
38660: Proc FirstDayOfMonthYmd( const Year : Int, const Month : Word, out Day : Word);
38661: Func diForceDirectories( const Dir :Str ) :Bool;
38662: Func ForceDirectoriesA( Dir : Ansistr ) :Bool;
38663: //Func ForceDirectoriesW( Dir : UnicodeString ) :Bool;
38664: Proc FreeMemAndNil( var Ptr: TObject);
38665: Func diGetCurrentFolder :Str;
38666: Func GetCurrentFolderA : Ansistr;
38667: //Func GetCurrentFolderW : UnicodeString;
38668: Proc SetCurrentFolder( const NewFolder :Str);
38669: Proc SetCurrentFolderA( const NewFolder : Ansistr);
38670: //Proc SetCurrentFolderW( const NewFolder : UnicodeString);
38671: Func diGetDesktopFolder :Str;
38672: Func GetDesktopFolderA : Ansistr;
38673: //Func GetDesktopFolderW : UnicodeString;
38674: Func diGetFileSize( const AFileName :Str ) : Int64;
38675: Func GetFileSizeA( const AFileName : Ansistr ) : Int64;
38676: //Func GetFileSizeW( const AFileName : UnicodeString ) : Int64;
38677: Func diGetDesktopDirectoryFolder :Str;
38678: Func GetDesktopDirectoryFolderA : Ansistr;
38679: //Func GetDesktopDirectoryFolderW : UnicodeString;
38680: Func GetFileLastWriteTime(const FileName:Str; out FileTime : TFileTime) :Bool;
38681: Func GetFileLastWriteTimeA(const FileName:Ansistr; out FileTime:TFileTime):Boolean;
38682: //Func GetFileLastWriteTimeW(const FileName:UnicodeString;out FileTime:TFileTime):Bool;
38683: Func diGetPersonalFolder( const PersonalFolder : Int ) :Str;
38684: Func GetPersonalFolderA : Ansistr;
38685: //Func GetPersonalFolderW : UnicodeString;
38686: Func GetSpecialFolder( const SpecialFolder : Int ) :Str;
38687: Func GetSpecialFolderA( const SpecialFolder : Int ) : Ansistr;
38688: //Func GetSpecialFolderW( const SpecialFolder : Int ) : UnicodeString;
38689: Proc diIncMonth( var Year : Int, var Month, Day : Word);
38690: Proc diIncMonths(var Year :Int, var Month, Day:Word; const NumberOfMonths:Int);
38691: Proc diIncDay( var Year : Int, var Month, Day : Word);
38692: Proc IncDays( var Year : Int, var Month, Day : Word, const Days : Int);
38693: Func IsDateValid( const Year : Int, const Month, Day : Word ) :Bool;
38694: Func IsHolidayInGermany( const Julian : TJulianDate ) :Bool;
38695: Func IsHolidayInGermanyYmd( const Year : Int, const Month, Day : Word ) :Bool;
38696: Func diIsLeapYear( const Year : Int ) :Bool;
38697: Func ISODateToJulianDate( const ISODate : TISODate ) : TJulianDate;
38698: Proc ISODateToYmd(const ISODate: TISODate; out Year : Int, out Month, Day : Word);
38699: Func IsCharLowLineW( const c : WideChar ) :Bool;
38700: Func IsCharQuoteW( const c : WideChar ) :Bool;
38701: Func IsShiftKeyDown :Bool;
38702: Func IsCharWhiteSpaceOrAmpersandW( const c : WideChar ) :Bool;

```

```

38703: Func IsCharWhiteSpaceOrNoBreakSpaceW( const c : WideChar) :Bool;
38704: Func IsCharWhiteSpaceOrColonW( const c : WideChar) :Bool;
38705: Func CharIsWhiteSpaceGtW( const c : WideChar) :Bool;
38706: Func CharIsWhiteSpaceLtW( const c : WideChar) :Bool;
38707: Func CharIsWhiteSpaceHyphenW( const c : WideChar) :Bool;
38708: Func CharIsWhiteSpaceHyphenGtW( const c : WideChar) :Bool;
38709: Func IsCharWordSeparatorW( const c : WideChar) :Bool;
38710: Func diISOWeekNumber( const JulianDate : TJulianDate) : Word;
38711: Func ISOWeekNumberYmd( const Year : Int; const Month, Day : Word) : Word;
38712: Func ISOWeekToJulianDate( const Year: Int; const WeekOfYear: Word; const DayOfWeek: Word) : TJulianDate;
38713: Func JulianDateIsWeekDay( const JulianDate : TJulianDate) : Bool;
38714: Func JulianDateToIsoDate( const Julian : TJulianDate) : TISODate;
38715: Proc JulianDateToYmd( const JulianDate: TJulianDate; out Year: Int; out Month, Day: Word);
38716: Func LastDayOfMonth( const JulianDate : TJulianDate) : TJulianDate;
38717: Proc LastDayOfMonthYmd( const Year : Int; const Month : Word; out Day : Word);
38718: Func LastDayOfWeek( const JulianDate : TJulianDate) : TJulianDate;
38719: Proc LastDayOfWeekYmd( var Year : Int; var Month, Day : Word);
38720: Func LastSysErrorMessage :Str;
38721: Func LastSysErrorMessageA : Ansistr;
38722: Func LastSysErrorMessageW : UnicodeString;
38723: Func diMax( const a : Int; const b : Int) : Int;;
38724: Func diMax3( const a, b, c : Int) : Int;
38725: Func MaxCard( const a :Card; const b :Card) :Card;;
38726: Func MaxCard3( const a :Card; const b :Card; const c :Card) :Card;;
38727: Func diMaxint64( const a : Int64; const b : Int64) : Int64;;
38728: Func diMaxint643( const a : Int64; const b : Int64; const c : Int64) : Int64;;
38729: Func diMin( const a, b : Int) : Int;;
38730: Func diMin3( const a, b, c : Int) : Int;
38731: Func MinCard( const a, b :Card) :Card;;
38732: Func MinCard3( const a, b, c :Card) :Card;;
38733: Func diMinint64( const a, b : Int64) : Int64;;
38734: Func diMinint643( const a, b, c : Int64) : Int64;;
38735: Func diMinint64U( const a, b : UInt64) : UInt64;;
38736: Func diMinint64U3( const a, b, c : UInt64) : UInt64;;
38737: Func MonthOfJulianDate( const JulianDate : TJulianDate) : Word;
38738: Func YearOfJulianDate( const JulianDate : TJulianDate) : Int;
38739: Func YmdToIsoDate( const Year : Int; const Month, Day : Word) : TISODate;
38740: Func YmdToJulianDate( const Year : Int; const Month, Day : Word) : TJulianDate;
38741: end;
38742:
38743: Func FiboMaxE2(n: Int): Extended;
38744: begin
38745:   result:= (pow((1+SQRT5)/2,n)-pow((1-SQRT5)/2,n))/SQRT5
38746: end;
38747: TDLLVersionInfo=Record
38748:   cbSize, // Size of the structure, in bytes.
38749:   dwMajorVersion, // Major version of DLL
38750:   dwMinorVersion, // Minor version of DLL
38751:   dwBuildNumber, // Build number of DLL
38752:   dwPlatformID: DWord; //Identifies the platform for which the DLL was built
38753: end;
38754:
38755: {$IFDEF MSWINDOWS}
38756: Proc TIdAntiFreeze_Process;
38757: var Msg: TMsg;
38758: ApplicationHasPriority:Bool;
38759: begin
38760:   if ApplicationHasPriority then begin
38761:     Application.ProcessMessages;
38762:   end else begin
38763:     // This guarantees it will not ever call Application.Idle
38764:     if PeekMessage(Msg, 0, 0, 0, PM_NOREMOVE) then begin
38765:       Application.HandleMessage;
38766:     end;
38767:   end;
38768: end;
38769: {$ENDIF}
38770:
38771: with lockbox you do shall with strings and hex:
38772: writeln(SHA1ofStr(loadstringj(exepath+'maxbox4.exe')))
38773: // ¢'BRL'mf0iY$1f0ai
38774: tempstr:= SHA1ofStr(loadstringj(exepath+'maxbox4.exe'))
38775: writeln(' ')
38776: for i:= 1 to 20 do
38777:   Write(charToHexStr(tempstr[i]));
38778:   A291DE524C1B916DCDD502EFDDA76C0BCED3F0EE
38779:
38780: Example with WMI:
38781: Func TestWMIOS2:Str;
38782: var wmiService, computerName, objectsList: OLEVariant;
38783: sysID:Str;
38784: isloc: ISWBemLocator; issuer: ISWBemServices;
38785: iset: ISWBemObjectSet; Enum: IEnumVariant;
38786: tempObj: OleVariant;
38787: begin
38788:   computerName := '.';
38789:   isloc:= WMIStart;
38790:   issuer:= WMIConnect2(isloc, '\root\cimv2','maxbox10','max',''); //: ISWBemServices;
38791:   //Exception: incorrect credentials. WMI connection failed.

```

```

38792: //Set wmiService = GetObject("WinMgmts:{impersonationLevel=impersonate, \"%_
38793: // \"(SystemTime, Shutdown)}!\" & computerName & "\root\cimv2")
38794: //Set objectsList = wmiService.InstancesOf("Win32_OperatingSystem")
38795: //For Each ObjService in objectsList
38796:
38797: iset:= WMIDeleteQuery(isser,'Select * from Win32_OperatingSystem')
38798: result:=(botoStr(WMIRowFindFirst(iset, ENum, tempObj)));
38799: repeat
38800:     sysid:= ('OSid: '+vartoStr(tempobj.csname)+
38801:         ' - '+vartoStr(tempobj.name)+
38802:         ' - '+vartoStr(tempobj.caption))
38803:     until not WMIRowFindNext(ENum, tempobj); //}
38804:     tempObj:= unassigned;
38805:     result:= result+' '+sysID;
38806: //Call Log.Message("SysInfo", sysID)
38807: End; // Sub
38808:
38809: >>> Sysid2: TRUE OSid: MAXBOX10 - Microsoft Windows 10 Home|C:\WINDOWS\Device\Harddisk0\Partition4 -
Microsoft Windows 10 Home
38810: command1:= 'play "'+songpath+'maxbox.wav"'; command2:= 'play "'+songpath+'moon.wav"';
38811: SendMCICCommand('open waveaudio shareable; //parallels
38812: SendMCICCommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\maxbox.wav";
38813: SendMCICCommand('play "G:\sonysavefeb2014\maxbox\maxbox3_back\examples\moon.wav";
38814: SendMCICCommand('close waveaudio;
38815: //MakeSound(Frequency{Hz},Duration{mSec}:Int;Volume:TVolumeLevel;savefilePath:str);
38816: if Not fileexists(exepath+'mytest1.wav') then begin
38817:     Writeln('Generate 2 sounds with 440 and 443 Hz');
38818:     MakeSound(440,5000, 80, Exepath+'mytest1.wav');
38819:     MakeSound(443,5000, 80, Exepath+'mytest2.wav');
38820: end else begin
38821:     Writeln('Play Beat of 2 sounds together with 3 Hz Amp Mod:');
38822:     mciSendString('PLAY "'+exepath+'mytest1.wav'+"', 'Nil', 0, 0);
38823:     mciSendString('PLAY "'+exepath+'mytest2.wav'+"', 'Nil', 0, 0);
38824: end;
38825:
38826: Strings (and other non ordinals) in Case Statements
38827: This has to be most requested feature by far in history of D imho, sure to make many long time D fans
happy.}
38828: writeln(floattostr(pow(256,200000)))
38829: //writeln(itoa(length(bigpow(256,200000)))) //481648
38830: sr:= sword;
38831: case sr of
38832:     'hello' : xi := 1;
38833:     'goodbye' : xi := 2;
38834:     sWorld : xi := 3; //sWorld is a string constant
38835: end;
38836: writeln('res of case '+itoa(xi));
38837:
38838: initialization XRaise
38839: type
38840:     EPointsArrayIsNotAssigned = Exception;
38841: writeln(itoa(length(aptarr)));
38842: setlength(aptarr, 2)
38843: writeln(itoa(length(aptarr)));
38844: if length(aptarr) = 0 then begin
38845:     Xraise(EPointsArrayIsNotAssigned.Create('Points array is not assigned...'));
38846: //Exit;
38847: end;
38848:
38849: maXbox SSL fox471 for
38850: The first step the following components, maybe we will add further in the future:
38851: 1. Orpheus (Win32 and Win64)
38852: 2. Abbrevia (Win32, Win64, MacOSX, iOS and Android)
38853: 3. Virtual Tree (Win32 and Win64)
38854: 4. SynEdit (Win32 and Win64)
38855: 5. LockBox (Win32, Win64, MacOSX, iOS and Android)
38856: 6. Async Professional (Win32)
38857: 7. PowerPDF (Win32 and Win64)
38858:
38859: //////////////////////////////////////
38860: All maXbox Tutorials Table of Content 2014/2015/2016/2017/2018/2019/2020/2021/2022/2023
38861: //////////////////////////////////////
38862: Tutorial 00 Function-Coding (Blix the Programmer)
38863: Tutorial 01 Procedural-Coding
38864: Tutorial 02 OO-Programming
38865: Tutorial 03 Modular Coding
38866: Tutorial 04 UML Use Case Coding
38867: Tutorial 05 Internet Coding
38868: Tutorial 06 Network Coding
38869: Tutorial 07 Game Graphics Coding
38870: Tutorial 08 Operating System Coding
38871: Tutorial 09 Database Coding
38872: Tutorial 10 Statistic Coding
38873: Tutorial 10_1 Probability Coding
38874: Tutorial 11 Forms Coding
38875: Tutorial 12 SQL DB Coding
38876: Tutorial 13 Crypto Coding
38877: Tutorial 14 Parallel Coding
38878: Tutorial 15 Serial RS232 Coding

```

38879: Tutorial 16 Event Driven Coding
38880: Tutorial 17 Web Server Coding
38881: Tutorial 18 **Arduino** System Coding
38882: Tutorial 18_3 RGB LED System Coding
38883: Tutorial 19 WinCOM /**Arduino** Coding
38884: Tutorial 20 Regular Expressions **RegEx**
38885: Tutorial 21 **Android** Coding (coming 2013)
38886: Tutorial 22 Services Programming
38887: Tutorial 23 Real Time Systems
38888: Tutorial 24 Clean Code
38889: Tutorial 25 **maXbox** Configuration I+II
38890: Tutorial 26 Socket Programming **with** TCP
38891: Tutorial 27 XML & TreeView
38892: Tutorial 28 DLL Coding (available)
38893: Tutorial 29 **UML** Scripting (2014)
38894: Tutorial 30 Web **of** Things (2014)
38895: Tutorial 31 Closures (2014)
38896: Tutorial 32 SQL Firebird (2014)
38897: Tutorial 33 Oscilloscope (available)
38898: Tutorial 34 GPS Navigation (2014)
38899: Tutorial 35 Web Box (available)
38900: Tutorial 36 **Unit** Testing (avail)
38901: Tutorial 37 API Coding (avail)
38902: Tutorial 38 3D Coding (coming 2015)
38903: Tutorial 39 GEO Map Coding (available)
38904: Tutorial 39_1 GEO Map Layers Coding (available)
38905: Tutorial 40 REST Coding (coming 2015)
38906: Tutorial 41 Big Numbers Coding (2015)
38907: Tutorial 42 Parallel Processing (2015)
38908: Tutorial 43 Code Metrics: June2016
38909: Tutorial 44 IDE Extensions
38910: Tutorial 45 Robotics: July2016
38911: Tutorial 46 WineHQ: Dez2016
38912: Tutor 47 RSA Crypto Jan2017
38913: Tutor 48 Microservice Jan2017
38914: http://www.softwareschule.ch/download/maxbox_starter48.pdf
38915: Tutorial 49 Refactoring: March 2017
38916: Tutorial 50 Big Numbers II: April 2017
38917: Tutorial 51 5 Use Cases April 2017
38918: Tutorial 52 Work **with** WMI Mai 2017
38919: Tutorial 53 Real Time **UML** 2017
38920: Tutor 54 Microservice II Nov 2017
38921: Tutor 55 ASCII Talk Dez 2017
38922: Tutor 56 Neural Network 2018
38923: Tutor 57 Neural Network II
38924: Tutor 58 Data Science
38925: Tutor 59 Big Data Feb 2018
38926: Tutor 60 Machine Learning March 2018
38927: Tutor 60.1 Sentiment Analysis
38928: Tutor 60.2 ML II
38929: Tutor 63 Machine Games
38930: Tutor 64 Install Routines
38931: Tutor 65 Machine Learning III
38932: Tutor 66 Machine Learning IV
38933: Tutor 67 Machine Learning V
38934: Tutor 68 Machine Learning VI
38935: Tutor 69 Machine Learning VII
38936: Tutor 70 No GUI Shell
38937: Tutor 71 CGI **and** WebSocket
38938: Tutor 72 Multilanguage
38939: Tutor 73 EKON 24 Edition
38940: Tutor 74 BASTA 2020 Vision
38941: Tutor 75 **Object** Detection
38942: Tutor 76 Image Classifier CAI
38943: Tutor 77 Image Classifier Matrix
38944: Tutor 78 Portable pixmap format (PPM)
38945: Tutor 79 **Unit** Testing **with** Asserts
38946: Tutor 80 My Tips & Tricks
38947: Tutor 81 RSS Feeds **of** BBC News - Digital Forensic March 2021
38948: Tutorial 82 JSON Code API
38949: Tutorial 82_2 Sentiment NLTK API Code
38950: #Tutorial 83 Unicode + Bayes Theorem & Confusion Matrix April 2021
38951: #Tutorial 84 JSON Primer Mai 2021
38952: Tutorial 83 Machine Learning XI Classification April 2021
38953: Tutorial 84 Machine Learning XII Baseline Mai 2021
38954: Tutorial 85 JSON4Delphi Automation Mai 2021
38955: Tutorial 86 Python4Delphi P4D June 2021 (Part 1-4)
38956: Tutorial 87 Image Detection **with** Lazarus
38957: Tutorial 88 Performance Tuning
38958: Tutorial 89 CAI mX4
38959: Tutorial 89_1 CAI mX4 model define
38960: Tutorial 89_2 CAI mX4 model fits
38961: Tutorial 90 Python CheatSheet Nov. 2021
38962: Tutorial 91 CAI CheatSheet Faker SynDat Dez. 2021
38963: Tutorial 92 Python CAI CheatSheet Jan. 2022
38964: Tutorial 92_1 VCL4Python Feb. 2022
38965: Tutorial 93 Geocoding March 2022
38966: Tutorial 94 Post API Services April 2022
38967: Tutorial 95 Language Services April 2022


```

38968: Tutorial 96 CNN Pipeline June 2022
38969: Tutor 97 Operating System Services OSS
38970: Tutor 98 Python Integration
38971: Tutorial 99 DataScience API October 2022
38972: Tutorial 100 Data Science Story November 2022
38973: Tutorial 101 Data Science Story2 Dezember 2022
38974: Tutorial 102 Interpreter versus Compiler December 2022
38975: Tutorial 103 Image to Text API January 2023
38976: Tutorial 104 Restcountries API January 2023
38977: Tutorial 105 Classify CNN February 2023
38978: Tutorial 106 Air Distance and Bearing, March 2023
38979: Tutorial 107 pas2js, March 2023
38980: Tutorial 108 Hacking your Märklin, March 2023
38981: Tutorial 109 Google Translate API, April 2023
38982: Tutorial 109_1 OpenAI ChatGPT API, April 2023
38983: Tutorial 110 Code Overview Samples, April 2023
38984:
38985: https://linuxschweizag.wordpress.com/2023/04/06/tutorials/
38986:
38987: Doc ref Docu for all Type Class and Const in maxBox_types.pdf
38988: using Docu for this file is maxbox_functions_all.pdf
38989: PEP - Pascal Education Program Low Lib Lab ShellHell in UDEMY
38990:
38991: https://bitbucket.org/max_kleiner/maxbox3/wiki/maxBox%20Tutorials
38992: http://stackoverflow.com/tags/pascalscript/hot
38993: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
38994: http://sourceforge.net/projects/maxBox #locs:62000
38995: http://sourceforge.net/apps/mediawiki/maxBox
38996: http://www.blaisepascal.eu/
38997: https://github.com/maxkleiner/maxBox3.git
38998: https://github.com/maxkleiner/maxBox4.git
38999: http://www.heise.de/download/maxbox-1176464.html
39000: http://www.softpedia.com/get/Programming/Other-Programming-Files/maxBox.shtm1
39001: https://www.facebook.com/pages/Programming-maxBox/166844836691703
39002: http://www.softwareschule.ch/arduino_training.pdf
39003: http://www.delphiarea.com
39004: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
39005: http://entwickler-konferenz.de/2014/speakers/max-kleiner
39006: http://www.heise.de/download/maxbox-1176464.html
39007: https://www.udemy.com/learn-coding-from-the-scratch
39008: http://www.slideshare.net/maxkleiner1/codesign-2015
39009: https://www.dropbox.com/s/yolconwmg4oqta4/Blaise_2and3_SP_Total.pdf?dl=0
39010: http://www.softwareschule.ch/download/maxbox_promo.png
39011: http://max.kleiner.com/maxbox_functions_all.htm
39012: http://max.kleiner.com/boxart.htm
39013: http://www.jurgott.org/linkage/util.htm
39014: http://www.swissdelphicenter.ch/en/niklauswirth.php
39015: http://www.softwareschule.ch/images/maxbox_20years_delphi.jpg
39016: https://softwareschule.code.blog/
39017: https://repl.it/@MaxKleiner/machinelearning4#main.py
39018: https://github.com/project-jedi/jvcl/tree/master/jvcl/examples/JvChartDemo
39019: https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas
39020:
39021: {$ All maxBox 4 Examples List Exampleslist }
39022: https://github.com/maxkleiner/maxBox3/releases
39023: https://github.com/maxkleiner/maxBox4/releases
39024: http://www.softwareschule.ch/download/exampleedition2016.zip
39025:
39026: *****
39027: 000_pas_baseconvert.txt 282_fadengraphik.txt
39028: 000_pas_baseconvert.txt_encrypt 283_SQL_API_messagetimeout.txt
39029: 000_pas_baseconvert.txt_decrypt 284_SysTools4.txt
39030: 001_1_pas_funcstest - Kopie.txt 285_MineForm_GR32.TXT
39031: 001_1_pas_funcstest.txt 285_MineForm_GR32main.TXT
39032: 001_1_pas_funcstest2.txt 285_MineForm_GR32mainsolution.TXT
39033: 001_1_pas_funcstest_clx2.txt 285_MineForm_propas.TXT
39034: 001_1_pas_funcstest_clx2_2.txt 285_MineForm_propas2.TXT
39035: 001_1_pas_funcstest_openarray.txt 285_minesweeper2.TXT
39036: 001_pas_lottogen.txt 285_Patterns_process.txt
39037: 001_pas_lottogen_template.txt 286_colormixer_jpeg_charcounter.txt
39038: 001_pas_lottogen.txtcopy 286_colormixer_jpeg_charcounter2.txt
39039: 002_pas_russianroulette.txt 287_eventhandling.txt
39040: 002_pas_russianroulette.txtcopy 287_eventhandling2.txt
39041: 002_pas_russianroulette.txtcopy_decrypt 287_eventhandling2_negpower.txt
39042: 002_pas_russianroulette.txtcopy_encrypt 288_bitblt.txt
39043: 003_pas_motion.txt 288_bitblt_resize.txt
39044: 003_pas_motion.txtcopy 289_regression.txt
39045: 004_pas_search.txt 289_regression2.txt
39046: 004_pas_search_replace.txt 290_bestofbox.txt
39047: 004_search_replace_allfunctionlist.txt 290_bestofbox2.txt
39048: 005_pas_oodesign.txt 290_bestofbox3.txt
39049: 005_pas_shelllink.txt 291_3sort_visual_thread.txt
39050: 006_pas_oobatch.txt 292_refactoring2.txt
39051: 007_pas_streamcopy.txt 293_bold_utils.txt
39052: 008_EINMALEINS_FUNC.TXT 293_ib_utils.txt
39053: 008_explanation.txt 293_ib_utils_timetest.txt
39054: 008_pas_verwechselt.txt 294_maxcalc_demo.txt
39055: 008_pas_verwechselt_ibz_bern_func.txt 294_maxcalc_demo2.txt
39056: 008_stack_ibz.TXT 295_easter_calendar.txt

```

```

39057: 009_pas_umlranner.txt
39058: 009_pas_umlranner_all.txt
39059: 009_pas_umlranner_componenttest.txt
39060: 009_pas_umlranner_solution.txt
39061: 009_pas_umlranner_solution_2step.txt
39062: 010_pas_oodesign_solution.txt
39063: 011_pas_puzzlepas_defect.txt
39064: 012_pas_umlranner_solution.txt
39065: 012_pas_umlranner_solution2.txt
39066: 013_pas_linenummer.txt
39067: 014_pas_primetest.txt
39068: 014_pas_primetest_first.txt
39069: 014_pas_primetest_sync.txt
39070: 015_pas_designbycontract.txt
39071: 015_pas_designbycontract_solution.txt
39072: 016_pas_searchrec.txt
39073: 017_chartgen.txt
39074: 018_data_simulator.txt
39075: 019_dez_to_bin.txt
39076: 019_dez_to_bin_grenzwert_ibz.txt
39077: 020_proc_feedback.txt
39078: 021_pas_symkey.txt
39079: 021_pas_symkey_solution.txt
39080: 022_pas_filestreams.txt
39081: 023_pas_find_searchrec.txt
39082: 023_pas_pathfind.txt
39083: 024_pas_TFileStream_records.txt
39084: 025_prime_direct.txt
39085: 026_pas_memorystream.txt
39086: 027_pas_shellexecute_beta.txt
39087: 027_pas_shellexecute_solution.txt
39088: 028_pas_dataset.txt
39089: 029_pas_assignfile.txt
39090: 029_pas_assignfile_dragndropexe.txt
39091: 030_palindrome_2.txt
39092: 030_palindrome_tester.txt
39093: 030_pas_recursion.txt
39094: 030_pas_recursion2.txt
39095: 031_pas_hashcode.txt
39096: 032_pas_crc_const.txt
39097: 033_pas_cipher.txt
39098: 033_pas_cipher_def.txt
39099: 033_pas_cipher_file_2_solution.txt
39100: 034_pas_soundbox.txt
39101: 035_pas_crcscript.txt
39102: 035_pas_CRCscript_modbus.txt
39103: 036_pas_includetest.txt
39104: 036_pas_includetest_basta.txt
39105: 037_pas_define_demo32.txt
39106: 038_pas_box_demonstrator.txt
39107: 039_pas_dllcall.txt
39108: 040_paspointer.txt
39109: 040_paspointer_old.txt
39110: 041_pasplotter.txt
39111: 041_pasplotter_plus.txt
39112: 042_pas_kgv_ggt.txt
39113: 043_pas_proceduretype.txt
39114: 044_pas_14queens_solwith14.txt
39115: 044_pas_8queens.txt
39116: 044_pas_8queens_sol2.txt
39117: 044_pas_8queens_solutions.txt
39118: 044_queens_performer.txt
39119: 044_queens_performer2.txt
39120: 044_queens_performer2tester.txt
39121: 045_pas_lishtandling.txt
39122: 046_pas_records.txt
39123: 047_pas_modula10.txt
39124: 048_pas_romans.txt
39125: 049_pas_ifdemo.txt
39126: 049_pas_ifdemo_BROKER.txt
39127: 050_pas_primetest2.txt
39128: 050_pas_primetest2_thieves.txt
39129: 050_program_starter.txt
39130: 050_program_starter_performance.txt
39131: 051_pas_findtext_solution.txt
39132: 052_pas_text_as_stream.txt
39133: 052_pas_text_as_stream_include.txt
39134: 053_pas_singleton.txt
39135: 054_pas_speakpassword.txt
39136: 054_pas_speakpassword2.txt
39137: 054_pas_speakpassword_searchtest.txt
39138: 055_pas_factorylist.txt
39139: 056_pas_demeter.txt
39140: 057_pas_dirfinder.txt
39141: 058_pas_filefinder.txt
39142: 058_pas_filefinder_pdf.txt
39143: 058_pas_filefinder_screview.txt
39144: 058_pas_filefinder_screview2.txt
39145: 058_pas_filefinder_screview3.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_realtimelock_arduino.txt
299_realtimelock_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
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

```

```

39146: 059_pas_timertest.txt
39147: 059_pas_timertest_2.txt
39148: 059_pas_timertest_time_solution.txt
39149: 059_timerobject_starter2.txt
39150: 059_timerobject_starter2_ibz2_async.txt
39151: 059_timerobject_starter2_uuml.txt
39152: 059_timerobject_starter2_uuml_main.txt
39153: 059_timerobject_starter4_ibz.txt
39154: 060_pas_datefind.txt
39155: 060_pas_datefind_exceptions2.txt
39156: 060_pas_datefind_exceptions_CHECKTEST.txt
39157: 060_pas_datefind_fulltext.txt
39158: 060_pas_datefind_plus.txt
39159: 060_pas_datefind_plus_mydate.txt
39160: 061_pas_randomwalk.txt
39161: 061_pas_randomwalk_plus.txt
39162: 062_paskorrelation.txt
39163: 063_pas_calculateform.txt
39164: 063_pas_calculateform_2list.txt
39165: 064_pas_timetest.txt
39166: 065_pas_bitcounter.txt
39167: 066_pas_eliza.txt
39168: 066_pas_eliza_include_sol.txt
39169: 067_pas_morse.txt
39170: 068_pas_piezo_sound.txt
39171: 069_LED_Matrix_R1_3_6_NV_PSchaeer.TXT
39172: 069_my_LEDBOX.TXT
39173: 069_pas_ledmatrix.txt
39174: 069_pas_LEDMATRIX_Alphabet.txt
39175: 069_pas_LEDMATRIX_Alphabet_run.txt
39176: 069_pas_LEDMATRIX_Alphabet_tester.txt
39177: 069_PAS_LEDMATRIX_COLOR.TXT
39178: 069_pas_ledmatrix_fixedit.txt
39179: 069_pas_LEDMATRIX_soundbox.txt
39180: 069_pas_LEDMATRIX_soundbox2.txt
39181: 069_Richter_MATRIX.TXT
39182: 070_pas_functionplot.txt
39183: 070_pas_functionplotter2.txt
39184: 070_pas_functionplotter2_mx4.txt
39185: 070_pas_functionplotter2_tester.txt
39186: 070_pas_functionplotter3.txt
39187: 070_pas_functionplotter4.txt
39188: 070_pas_functionplotter_digital.txt
39189: 070_pas_functionplotter_elliptic.txt
39190: 070_pas_function_helmholtz.txt
39191: 070_pas_textcheck_experimental.txt
39192: 071_pas_graphics.txt
39193: 071_pas_graphics_drawsym.txt
39194: 071_pas_graphics_drawsym_save.txt
39195: 071_pas_graphics_random.txt
39196: 072_pas_fractals.txt
39197: 072_pas_fractals_2.txt
39198: 072_pas_fractals_blackhole.txt
39199: 072_pas_fractals_performace.txt
39200: 072_pas_fractals_performace_new.txt
39201: 072_pas_fractals_performance_sharp.txt
39202: 072_pas_fractals_performance.txt
39203: 072_pas_fractals_performance_mX4.txt
39204: 073_pas_forms.txt
39205: 074_pas_chartgenerator.txt
39206: 074_pas_chartgenerator_solution.txt
39207: 074_pas_chartgenerator_solution_back.txt
39208: 074_pas_charts.txt
39209: 075_bitmap_Artwork2.txt
39210: 075_pas_bitmappuzzle.txt
39211: 075_pas_bitmappuzzle24.prod.txt
39212: 075_pas_bitmappuzzle2_prod.txt
39213: 075_pas_bitmappuzzle3.txt
39214: 075_pas_bitmapsolve.txt
39215: 075_pas_bitmap_Artwork.txt
39216: 075_pas_puzzlepas_solution.txt
39217: 076_pas_3dcube.txt
39218: 076_pas_circle.txt
39219: 077_pas_mmshow.txt
39220: 078_pas_pi.txt
39221: 079_pas_3dcube_animation.txt
39222: 079_pas_3dcube_animation4.txt
39223: 079_pas_3dcube_plus.txt
39224: 080_pas_hanoi.txt
39225: 080_pas_hanoi2.txt
39226: 080_pas_hanoi2_file.txt
39227: 080_pas_hanoi2_sol.txt
39228: 080_pas_hanoi2_tester.txt
39229: 080_pas_hanoi2_tester_fast.txt
39230: 080_pas_hanoi3.txt
39231: 081_pas_chartist2.txt
39232: 082_pas_biorythmus.txt
39233: 082_pas_biorythmus_solution.txt
39234: 082_pas_biorythmus_solution_3.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
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
372_stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt

```

```

39235: 082_pas_biorythmus_test.txt
39236: 083_pas_GITARRE.txt
39237: 083_pas_soundbox_tones.txt
39238: 084_pas_waves.txt
39239: 085_mxsinus_logo.txt
39240: 085_sinus_plot_waves.txt
39241: 086_pas_graph_arrow_heart.txt
39242: 087_bitmap_loader.txt
39243: 087_pas_bitmap_solution.txt
39244: 087_pas_bitmap_solution2.txt
39245: 087_pas_bitmap_subimage.txt
39246: 087_pas_bitmap_test.txt
39247: 088_pas_soundbox2_mp3.txt
39248: 088_pas_soundbox_mp3.txt
39249: 088_pas_sphere_2.txt
39250: 089_pas_gradient.txt
39251: 089_pas_maxland2.txt
39252: 090_pas_sudoku4.txt
39253: 090_pas_sudoku4_2.txt
39254: 091_pas_cube4.txt
39255: 092_pas_statistics4.txt
39256: 093_variance.txt
39257: 093_variance_debug.txt
39258: 094_pas_daysold.txt
39259: 094_pas_stat_date.txt
39260: 095_pas_ki_simulation.txt
39261: 096_pas_geisen_problem.txt
39262: 096_pas_montyhall_problem.txt
39263: 097_lotto_proofofconcept.txt
39264: 097_pas_lottocombinations_beat_plus.txt
39265: 097_pas_lottocombinations_beat_plus2.txt
39266: 097_pas_lottocombinations_universal.txt
39267: 097_pas_lottosimulation.txt
39268: 098_pas_chartgenerator_plus.txt
39269: 099_pas_3D_show.txt
39270: 200_big_numbers.txt
39271: 200_big_numbers2.txt
39272: 201_streamload_xml.txt
39273: 202_systemcheck.txt
39274: 203_webservice_simple_intftester.txt
39275: 204_webservice_simple.txt
39276: 205_future_value_service.txt
39277: 206_DTD_string_functions.txt
39278: 207_ibz2_async_process.txt
39279: 208_crc32_hash.txt
39280: 209_cryptohash.txt
39281: 210_public_private.txt
39282: 210_public_private_cryptosystem5_ibz_spektrum.txt
39283: 210_public_private_cryptosystem.txt
39284: 211_wipe_pattern.txt
39285: 211_wipe_pattern2.txt
39286: 211_wipe_pattern_solution.txt
39287: 212_pas_statisticmodule4.TXT
39288: 212_pas_statisticmodule4.txt.TXT
39289: 212_statisticmodule4.txt
39290: 213_pas_BBP_Algo.txt
39291: 214_mxdocudemo.txt
39292: 214_mxdocudemo2.txt
39293: 214_mxdocudemo3.txt
39294: 215_hints_test.TXT
39295: 216_warnings_test.TXT
39296: 217_pas_heartbeat.txt
39297: 218_biorythmus_studio.txt
39298: 219_cipherbox.txt
39299: 219_crypt_source_comtest_solution.TXT
39300: 220_cipherbox_form.txt
39301: 220_cipherbox_form2.txt
39302: 221_bcd_explain.txt
39303: 222_memoform.txt
39304: 223_directorybox.txt
39305: 224_dialogs.txt
39306: 225_sprite_animation.txt
39307: 226_ASCII_Grid2.TXT
39308: 227_animation.txt
39309: 227_animation2.txt
39310: 228_android_calendar.txt
39311: 229_android_game.txt
39312: 229_android_game_tester.txt
39313: 230_DataProvider.txt
39314: 230_DataSetProvider.txt
39315: 230_DataSetXMLBackupScholz.txt
39316: 231_DBGGrid_access.txt
39317: 231_DBGGrid_XMLaccess.txt
39318: 231_DBGGrid_XMLaccess2.txt
39319: 231_DBGGrid_XMLaccess_locatetester.txt
39320: 231_DBGGrid_XML_CDS_local.txt
39321: 232_outline.txt
39322: 232_outline_2.txt
39323: 233_modular_form.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_timefunc_testerfilemon.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
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_bitbox.txt

```

```

39324: 234_debugoutform.txt
39325: 235_fastform.TXT
39326: 236_componentpower.txt
39327: 236_componentpower_back.txt
39328: 237_pas_4forms.txt
39329: 238_lottogen_form.txt
39330: 239_pas_sierpinski.txt
39331: 239_pas_sierpinski2.txt
39332: 240_unitGlobal_tester.txt
39333: 241_db3_sql_tutorial.txt
39334: 241_db3_sql_tutorial2.txt
39335: 241_db3_sql_tutorial2fix.txt
39336: 241_db3_sql_tutorial3.txt
39337: 241_db3_sql_tutorial3connect.txt
39338: 241_db3_sql_tutorial3_fpctest.txt
39339: 241_RTL_SET2.txt
39340: 241_RTL_SET2_tester.txt
39341: 242_Component_Control.txt
39342: 243_tutorial_loader.txt
39343: 244_script_loader_loop.txt
39344: 245_formapp2.txt
39345: 245_formapp2_tester.txt
39346: 245_formapp2_testerX.txt
39347: 246_httpapp.txt
39348: 247_datecalendar.txt
39349: 248_ASCII_Grid2_sorted.TXT
39350: 249_picture_grid.TXT
39351: 250_tipsandtricks2.txt
39352: 250_tipsandtricks3.txt
39353: 250_tipsandtricks3api.txt
39354: 250_tipsandtricks3_admin_elevation.txt
39355: 250_tipsandtricks3_tester.txt
39356: 250_tipsandtricks4_tester.txt
39357: 250_tipsandtricks4_tester2.txt
39358: 251_compare_noise_gauss.txt
39359: 251_whitenoise.txt
39360: 251_whitenoise2.txt
39361: 252_hilbert_turtle.txt
39362: 252_pas_hilbert.txt
39363: 253_opearatingsystem3.txt
39364: 254_dynarrays.txt
39365: 255_einstein.txt
39366: 256_findconsts_of_EXE.txt
39367: 256_findfunctions2_of_EXE.txt
39368: 256_findfunctions2_of_EXEeverp.txt
39369: 256_findfunctions2_of_EXEspec.txt
39370: 256_findfunctions3.txt
39371: 256_findfunctions_of_EXE.txt
39372: 257_AES_Cipher.txt
39373: 258_AES_cryptobox.txt
39374: 258_AES_cryptobox2.txt
39375: 258_AES_cryptobox2_passdlg.txt
39376: 259_AES_crypt_directory.txt
39377: 260_sendmessage_2.TXT
39378: 260_sendmessage_beta.TXT
39379: 261_probability.txt
39380: 262_mxoutputdemo4.txt
39381: 263_async_sound.txt
39382: 264_vclutils.txt
39383: 264_VCL_utils2.txt
39384: 265_timer_API.txt
39385: 266_serial_interface.txt
39386: 266_serial_interface2.txt
39387: 266_serial_interface3.txt
39388: 267_ackermann_rec.txt
39389: 267_ackermann_variants.txt
39390: 268_DBGGrid_tree.txt
39391: 268_record_grid.TXT
39392: 270_Jedi_FunctionPower.txt
39393: 270_Jedi_FunctionPowertestter.txt
39394: 271_closures_study.txt
39395: 271_closures_study_workingset2.txt
39396: 272_pas_function_show.txt
39397: 273_pas_function_show2.txt
39398: 274_library_functions.txt
39399: 275_turtle_language.txt
39400: 275_turtle_languagehouse2_time2.txt
39401: 275_turtle_language_save.txt
39402: 276_save_algo.txt
39403: 276_save_algo2.txt
39404: 277_functionsfor39.txt
39405: 278_DB_Dialogs.TXT
39406: 279_hexer2.TXT
39407: 279_hexer2macro.TXT
39408: 279_hexer2macroback.TXT
39409: 280_UML_process.txt
39410: 280_UML_process_knabe2.txt
39411: 280_UML_process_knabe3.txt
39412: 280_UML_process_TIM_Botzenhardt.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
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

```



```

39413: 280_UML_TIM_Seitz.txt
39414: 281_picturepuzzle.txt
39415: 281_picturepuzzle2.txt
39416: 281_picturepuzzle3.txt
39417: 281_picturepuzzle4.txt
39418: 479_inputquery.txt
39419: 480_regex_pathfinder2.txt
39420: 482_processPipe.txt
39421: 483_PathFuncTest_mX.txt
39422: 485_InnoFunc.txt
39423: 487_asyncKeyState.txt
39424: 489_simpleComport.txt
39425: 491_analogmeter.txt
39426: 493_gadgets.txt
39427: 496_InstallX.txt
39428: 498_UnitTesting.txt
39429: 500_diceoflives.txt
39430: 502_findalldocs.txt
39431: 504_fileclass.txt
39432: 506_colormatrix.txt
39433: 508_simplecomportmorse.txt
39434: 509_509_GEOMap2_SReverse.TXT
39435: 511_LEDLabel.txt
39436: 513_StreamIntegration.txt
39437: 515_ledclock3.txt
39438: 517_animation7.txt
39439: 519_powtils.txt
39440: 521_iputils2.txt
39441: 523_NMEA.txt
39442: 525_GEO84s.txt
39443: 527_GPSDemo.txt
39444: 529_profilerTest.txt
39445: 531_profilerTest.txt
39446: 533_syncasyn_demo.txt
39447: 535_Battleship3.pas
39448: 537_iniplus.TXT
39449: 539_timeturtle123.txt
39450: 541_webserver_arduino_motorturtle.txt
39451: 543_MATH_TurboP.PAS
39452: 545_strips.TXT
39453: 547_regexmaster.TXT
39454: 549_3D_Panorama.txt
39455: 551_ArduinoTester.txt
39456: 553_ArduinoCockBit3.txt
39457: 556_stringlistrandom.TXT
39458: 558_highrestimer.TXT -559_highthrestimer2
39459: 561_newfunctions399160.txt
39460: 563_moonpaper.txt
39461: 565_ConsoleCapture.txt
39462: 567_SquareWordGrids2.txt
39463: 569_keylog.txt - 569_ServiceMgr2.TXT
39464: 571_myPing.txt
39465: 573_modbusfrm_Main.pas
39466: 575_TARTARUGA_Desktop.txt
39467: 577_listbox2list.txt
39468: 579_numbersystems_sort.txt
39469: 581_stringstream.txt
39470: 583_VirtualConstructor_savereport.txt
39471: 585_fulltextfinder_cleancode_override.txt
39472: 587_one_function.txt
39473: 589_avi_animate.txt
39474: 591_emailattach.txt
39475: 593_round_time.txt
39476: 595_check_memory.txt
39477: 597_ole_commands.txt
39478: 599_bug.txt
39479: 601_PECheckSum.txt
39480: 603_cupids_arrow.TXT
39481: 605_maxonmotor3DTage2BASTA2015.TXT
39482: 607_DataSetProvider_CDS_ADO.txt
39483: 609_ScriptExecutor (beta)
39484: 611_Arduino_COMOutputs.txt
39485: 611_Arduino_COMOutputs_TMP36_3.txt
39486: 613_uPSI_DIUtils_test.pas
39487: 615_SONAR_51_Starter.txt
39488: (589_AVI_Throbber.res.txt)
39489: 616_ComTerminalDlg.pas
39490:
39491: 617_API_coding_tut37.txt
39492: 618_bayes_filter.txt
39493: 618_bayes_filter_dic2.txt
39494: 619_crypto_package_demo_policy.txt
39495: 620_serialtimer.txt
39496: 622_netsh_master_restore4.psb
39497: 623_exspawnu.pas
39498: 624_MiscTest_lexscanner.pas
39499: 625_binomial_bigint3_testcase.txt
39500: 627_4gewinnt_main3.txt
39501: 629_Word_FORMULA.PAS
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_datanap_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
608_ColorMixer_Arduino2.txt
610_3D_DLL.txt
612_MTerminal.pas
614_inbrowserock.txt
616_ComTerminalDlg.pas
589_avi_animate.txt
//end for 3.9.9.195
617_API_coding_tut39.txt
618_bayes_filter_dic.txt
619_crypto_package_demo.txt
619_crypto_packr.txt
621_bruteforceattack_dic.txt
622_netsh_master_sonar.pas
623_exspawnu_bufferoverflow.pas
624_MiscTest_lexscanner2.pas
626_lorentztzTransformation.TXT
628_FormatFloat.pas
630_multikernel2.TXT

```

```

39502: 630_multikernel3.TXT
39503: 632_safeCALL12.txt
39504: 632_safecheck2_regex_basta.txt
39505: 634_briefcase_XMLDB.xml
39506: 634_maxbase3server.mdb
39507: 636_rest_apilexamples.txt
39508: 637_psnconvert4.TXT
39509: 638_Mathslib_prime.pas.txt
39510: 640_rest_geocode.txt
39511: 640_weather_cockpit5.TXT
39512: 641_voicecommand.TXT
39513: 643_EKON19_Tester3_OLE_Reg.TXT
39514: 645_age_guess_REX.pas
39515: 646_pi_evil2.TXT
39516: 648_SYNEdit_ExportDemo.TXT
39517: 649_StringlistHTMLtestBasta.TXT
39518: 650_http_server2tester.txt
39519: 650_dbblobber.txt
39520: 651_StrUtilmaxbase.pas
39521: 653_Pdoxtstbox.pas
39522: 655_arduino_chess.txt
39523: 656_XML_RpcCommon.pas
39524: 657_PEM_Cert.txt
39525: 659_VirtListBox.txt
39526: 660_InetUtils_Rest.txt
39527: 662_VisualChronForm.pas
39528: 664_DataAnnotationsValid.pas.pas
39529: 666_SysInfoCtrls.pas.pas
39530: 667_URobo2.pas.pas
39531: 669_uptime.txt.pas
39532: 670_interesting_birthday_paradox.txt
39533: 672_regex_ask_task.txt
39534: 674_pipe_graphics.txt
39535: 676_run_as_admin.txt
39536: 678_fmTraceRouteMainU.pas
39537: 680_gravity_waves.txt
39538: 682_ScreenMatrixToBitmap.pas
39539: 684_mathe4.pas
39540: 686_namedpipe_demo.txt
39541: 687_memorymapped_filestream.txt
39542: 689_proctype_alias.txt
39543: 691_httpservermain3.txt
39544: 692_imageserv_fClient33.pas
39545: 694_OverbyteIcsAsn1Utils.pas
39546: 696_kmemo_demo_Main.pas
39547: 697_OverbyteIcsTicks64.pas
39548: 698_new_classes_tester.pas
39549: 698_Bitcoin_blockchaintester.pas
39550: 700_mX422_functions_demo2.txt
39551: 700_new_function_snippets.pas
39552: 701_all_fibonacci.txt
39553: 702_X509_Cert_OpenSSL_1.txt
39554: 703_pc_sensors.txt
39555: 705_magsubsl_functions.pas
39556: 706_population_count.txt
39557: 708_XML_DOM_ReaderTutor.txt
39558: 710_enigma_riddler.txt
39559: 710_geo_distance_form_mapbox.txt
39560: 710_RegSvrUtils_mX4.pas
39561: 712_rootsfunction.txt
39562: 713_SnakeA2.pas
39563: 714_astroids_GAME3.PAS
39564: 715_shape_calculator.PAS
39565: 717_AdLog.PAS
39566: 719_rootsfunction.txt
39567: 720_HTML_TableExport_CSS.pas
39568: 721_DOSCapture_netstat2.TXT
39569: 723_remote_notepad_message.TXT
39570: 725_CRC32_live.txt
39571: 725_725_Cholesky.pas
39572: 725_blockchain_sensors3.txt
39573: 726_DBGGrid_XML_CDataSet.txt
39574: 728_mci_macros.TXT
39575: 729_PrivateKey_mX.pas
39576: 730_modular_exponentiation_IBZ.txt
39577: 730_ternary_logice.txt
39578: 731_DWS_Client4_EKON20.pas
39579: 732_hailstonesequences_delphi.txt
39580: 733_jensensdevice.txt
39581: 735_Linear_congruential_generator.txt
39582: 735_Euler_phi17_openSSL_scripts.txt
39583: 737_xmastree2.txt
39584: 738_mX4_exact_fishtest2.pas
39585: 739_GIFPAL.PAS
39586: 740_Globalreplacetext.txt
39587: 740_msgpump1.txt
39588: 740_uFTPServermX2_listtest.pas
39589: 740_RosettaIsaac_Cryptocode_Delphi
39590: 741_DrawYinAndYang.txt
631_dmath_testfunc_mx3.pas
632_safecheck2.txt
633_gamma_func.txt
634_BriefcaseMain2.pas
635_bitcount_bytecodestudy.txt
637_psnconvert3.TXT
637_startscript_psnconvert2.TXT
639_prime_factors.txt
640_rest_weather_report.txt
640_weather_cockpit6.TXT
642_BDE_export.PAS
644_Synapse_Mimetest_regEx3.txt
646_pi_evil.TXT
647_StringtoHTMLBasta.TXT
649_dayio.pas
650_sched.pas
650_time_routines.txt
650_drawdice2016.txt
652_graph3DMainUnit2.pas
654_spectrum_dice2016_3.txt
655_arduino_chess2.txt
656_XML_RpcCommon2.pas
658_ASN1_Cert.txt
660_X509_Cert.pas
661_REXX.pas
663_Tokens.pas
665_keditcommon.pas.pas
667_URobol.pas.pas
668_U_SoundGen3.pas.pas
700_interestingX.txt
671_replacedigit.txt //File
673_pipe_singapore.txt //LAB
675_bitcoin_doublehash.txt
677_pingtest4.txt
679_messagefMain.pas
681_kronos_3DLab.TXT
683_mx422_functions_demo.txt
685_leanfitmath.txt
686_namedpipe_demo2.txt
688_KGraphics_Call.txt
690_calc_pi_ex.pas
692_fservermain3.pas
693_OverbyteIcsUtils_tester.pas
695_IdAntiFreeze_tester.pas
696_kmemo_demo.pas
697_OverbyteIcsTicks64Demo1.pas
698_OverbyteIcsDnsQuerytester.pas
699_OverbyteIcsShal_tester.pas
700_mx422_functions_demo3.txt
700_function_snippets3_metrics.inc
702_pascal_fibonacci.txt
702_X509_Cert_OpenSSL_2.txt
704_entropy_test.txt
705_parser_demo_case.txt
707_GrayCode.txt
709_BarcodeReader.txt
710_EnumRegKeys.txt
710_geo_distance_dialog.txt
711_geo_satellite_mapbox.txt
712_towerofhanoi_animation.pas
713_internet_traces_mx_screenshot2.png
714_GAME_spaceinvaders.PAS
716_webplan.PAS
718_AdPerformanceCounter.PAS
720_euler_project.txt
720_selfdescribing_number.txt
722_refererHeader_monitor.TXT
724_spyware_monitor_test.TXT
725_Install_Routines.TXT
725_Voice3.TXT
725_X509_SHA1_TestCert.txt
727_Canvas_Artist.TXT
728_mci_macros_imagetest.TXT
729_U_BigIntTest_mX.pas
730_topological4.txt
730_ternary_logice2.txt
731_DWS_Server4_EKON20.pas
732_hailstonesequences.txt
734_boxthecompass.txt
735_mx_screenshot_duesseldorf.png
736_hiddenstreams_ads.txt
737_xmastree3.txt
738_mX4_exact_fishtest3.pas
739_mXbinom.pas
740_Globalreplacetext_test.txt
740_uFTPServermX2.pas
740_RosettaIsaac_cryptocode.txt
740_RosettaIsaac_cryptocode2.txt
741_DrawYinAndYang_console.txt

```

```

39591: 742_VectorProducts.txt
39592: 744_Web_scraping_UTCTime.txt
39593: 745_crypto_memory.txt
39594: 746_uText.pas
39595: 747_uSteganos.pas
39596: 749_helloWebServer.txt
39597: 749_helloWebServer3.txt
39598: 750_EchoServer3.txt
39599: 210_RSA_crypto_complete8hybrid.txt
39600: 750_ibz_cryptomem_RSA_proof_64.txt
39601: 750_SimpleTCMain.pas
39602: 751_Elevator_Simulator.pas
39603: 752_BitmapHistogram.txt
39604: 752_U_AstroDemo_mX4_2.pas
39605: 753_xbase_tester.pas
39606: 755_science_ibz_spektrum.txt
39607: 755_sedol2.txt
39608: 756_shellscripthell_javascript.txt
39609: 756_shellscripthell_javascript2.txt
39610: 667_URobo2EKON_FUN_Tracking2.pas
39611: 757_tower_of_hanoiX.txt
39612: 758_speechmachine.txt
39613: 760_Normal_distribution.txt
39614: 760_XML_DOM_OLB.TXT
39615: 760_systeminfo_tester.txt
39616: 766_wmi_GetServiceStatus.txt
39617: 760_systeminfo_tester2.txt
39618: 761_BoulesBillard3.txt
39619: 760_systeminfo_tester2.txt
39620: 761_BoulesBillard3.txt
39621: 762_regex_analytica.txt
39622: 762_matlab.txt
39623: 764_reverse.txt
39624: 765_codesign.txt
39625: 766_wmi_management.txt
39626: 767_mint_geo_map.txt
39627: 769_tzutil.pas
39628: 770_pascal_coin_api_ascii_talk.txt
39629: 770_U_ParsingDemo3_ASCIIITalk.pas
39630: 770_3 hires_benchmarktimer3.txt
39631: 771_fibonacci_first.txt
39632: 773_streamfibonacci_first.txt
39633: 774_findwindow_calc.txt
39634: 776_fiboseq_state_android_pi.txt
39635: 777_netportscan.txt
39636: 777_tictactoe2.txt
39637: 777_tictac4picboard_16F887_arduino.txt
39638: 778_advapi32_dll_SHA256.txt
39639: 779_drivemanager.txt
39640: 780_mandelbrotset_max.txt
39641: 780_set_enumeration.txt
39642: 780_newfunctions_calctest_42610.txt
39643: 780_abc_problem.txt
39644: 780_mandelbrotset_max2.txt
39645: 780_thingspeak_mathworks.txt
39646: 780_JSONDemo.txt
39647: 780_ppk_xmltransform.txt
39648: 780_xrt1_net_URIUtils.pas
39649: 781_ProxyUtils.txt
39650: 781_ProxyUtils2performance.txt
39651: 782_XML_DOM_ADO_API_REST_demo.txt
39652: 783_turtle_sign.txt
39653: 785_Matrix_Transpose.txt
39654: 787_cologne_Bitmap21.txt
39655: 789_DOM_XML.txt
39656: 791_uFTPServer.pas
39657: 793_uOneInstance.pas
39658: 795_uOutput.pas
39659: 797_SynEditCodeFolding_routines.pas
39660: 799_list_duplicates.txt
39661: 800_isodatettime.txt
39662: 802_indystack_geo2_picture.txt
39663: 803_textanalysis2.txt
39664: 805_webspider_Mainconsole.pas
39665: 805_webspider_Mainconsole2_2.pas
39666: 807_FANN_XorSample.pas
39667: 807_FANN_XorSample2.pas
39668: 810_bulldog_mastermind.txt
39669: 810_neuralnetwork2.txt
39670: 810_neuralnetwork21.txt
39671: 811_mXpctest_dmath_datascience2.pas
39672: 813_PCA_datascience_iris.txt
39673: 814_bigmulu.txt
39674: 813_PCA_datascience_iris3.txt
39675: 815_look_and_say.txt
39676: 816_SnowballProgram.pas
39677: 818_spider_traversaldemo.txt
39678: 819_ES.Utils.pas
39679: 820_U_ProbabilityDist.pas
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_SimpleTCMainForm.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_OLB_bestoffbox4.TXT
751_Elevator_Simulator2.pas
751_Elevator_Simulator4.pas
760_systeminfo_tester3.txt
maxbox_billard2.png
760_systeminfo_tester3.txt
761_maxbox_billard2.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_3 hires_benchmarktimer2.txt
770_U_ParsingDemo3_ASCIIITalk_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_FONTVIEW1mX4.PAS
788_WMI_Transpose.txt
790_Start_Elasticsearch.pas
792_uMRU.pas
794_uFind.pas
796_uUtils.pas
798_turtle_python_time2.txt
800_wraplines.txt
801_algorithms_Devices.txt
803_textanalysis.txt
804_BackgroundWorker.pas
805_webspider_Mainconsole2.pas
806_primes_threadMain.pas
808_FANN_XorSample_traindata.pas
809_FANN_XorSample_traindata.pas
810_python_shell.txt
810_python_shell2.txt
811_mXpctest_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_traversaldemo2.txt
820_eigenvalue.txt
820_U_ProbabilityDist2.pas

```

```

39680: 820_U_ProbabilityDist2.psb
39681: 820_bostondataset_socks4test.pas
39682: 820_bostondataset_socks4test_irisset.pas
39683: 820_bostondataset_irisset_clean.pas
39684: 821_Narcissistic.txt
39685: 822_perceptron1_2.htm
39686: 822_perceptron1_2.txt
39687: 822_perceptron1_3.htm
39688: 807_FANN_PerceptronSample.pas
39689: 822_perceptron_for_pascal2.txt
39690: 822_perceptron_for_pascal2regression.txt
39691: 824_catamorphism.txt
39692: 825_uTPLb_Hash_tester.pas
39693: 826_sentiment_api2_1.txt
39694: 826_sentiment_api2_2.txt
39695: 820_bytecodetest.txt
39696: 826_reglin_regression_fit2.pas
39697: 827_save_UTF8.txt
39698: 828_install_routine2.txt
39699: 829_save_JSON.txt
39700: 830_RSA_cryptosystem_wiki.txt
39701: 831_regex_light.txt
39702: 833_findfactors.txt
39703: 835_fibonsteps_sequence.txt
39704: 837_penneysgame.txt
39705: 838_endofuniverse_explain.txt
39706: 840_maxboxmandel2.png
39707: 840_URungeKutta4test.pas
39708: 842_U_Keno2.pas
39709: 842_U_JPGClock2.pas
39710: 844_OpenSSL_synapsetester.txt
39711: 845_Gray_Code.txt
39712: 845_TestPermutations2.txt
39713: 846_longmultiplication.pas
39714: 848_xml_intfn.pas
39715: 849_pi3.pas
39716: 810_neuralnetwork2lpascal_tutor66.txt
39717: 810_neuralnetwork2lpascal_tutor66predict.txt
39718: 854_U_Invertedtext.pas
39719: 855_dll_from_maxbox.pas
39720: 856_snake_maxbox4_4.png
39721: 856_snake_maxbox4_5.png
39722: 856_snake_maxbox4_6.png
39723: 856_SnakeAmX42_tab.pas
39724: 859_maputils_client.pas
39725: 860_virtualbox_overview.png
39726: 826_similarity_api2.txt
39727: 860_similarity_api3.txt
39728: 860_sexy_primes2performance.txt
39729: 860_sexy_primes2performance.psb
39730: 860_sexy_primes.txt
39731: 860_PascalMatrix_Out.txt
39732: 860_graph3.png
39733: 826_similarity_api2.txt
39734: 860_virtualbox_overview.png
39735: 859_maputils_client.pas
39736: 861_stringtokenizer.txt
39737: 861_stringtokenizer2.txt
39738: 862_benfordslaw_synedit.txt
39739: 863_stack_tcp_chatroom.TXT
39740: 865_caesar_cipher.txt
39741: 865_winservices2.txt
39742: 866_tutor70consolepowershell2.png
39743: 780_abc_problem_delphi_synapse2.txt
39744: 781_LatLonDistmX.pas
39745: 781_JLatLon_JFK-ZRH.JPG
39746: 782_U_MakeCityLocations2mX4.pas
39747: 784_checkers.txt
39748: 786_web_scrap.txt
39749: 788_PETRA.MID
39750: 789_totient_primes_tester.txt
39751: 790_sha512.txt
39752: 867_metadefener_api.txt
39753: 869_factsum.txt
39754: 871_checkemptyfolders.txt
39755: 872_RSA_codes.txt
39756: 873_Unittester_mX47.txt
39757: 873_Unittester_mX47_2.psb
39758: 874_LinkGrabUnit1.pas
39759: 875_filehistounit2app_forensic.txt
39760: 876_console_output.txt
39761: 877_flowpanel2.pas
39762: 879_psvCGI.pas
39763: 880_visualstatemachine_main.pas
39764: 880_cgiMain_server2.pas
39765: 881_odometer_uMain2.pas
39766: 883_create_dataset.txt
39767: 884_google_chart_api2.TXT
39768: 886_AmicablePairs.TXT
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_neuralnetwork2lpython_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_neuralnetwork2l_levenshtein_dist.txt
860_similarity_api3.txt
860_brownian_tree.pas
860_brownian_tree.pas
860_graph.png
827_save_UTF8_xml_olddb.txt
861_stringtokenizer.htm
862_socialmention_api3.txt
863_stack_tcp_chatroom_client.TXT
864_form_canvas_font_client.TXT
865_winservices.txt
866_native_console.txt
780_abc_problem_delphi_synapse.txt
780_abc_problem_delphi_synapse3.txt
781u_LatLonDistanceTestmX.pas
781u_LatLonDistanceTestmX2.pas
783_replaceall.txt
785_reviews.txt
787_Excel_ImportExport.txt
788_pcu_mX4.pas
789_totient_primes_tester.psb
790_googlemaps.txt
868_6MainFmnetworkmX47.pas
870_totient_primes_tester_471_functions.txt
496_installX.txt
872_RSA_codes2.txt
873_Unittester_mX47_2.txt
873_httpclient.png
875_filehistounit2app.txt
875_codetester_GUI_maxbox4exe.png
877_flowpanel.pas
878_code_reflection.txt
880_cgiMain_server.pas
880_ModemEmulator.pas
640_API_LayerDetection_EKON23.TXT
882_bitmap_brusher.txt
884_google_chart_api.TXT
885_IP_Geolocation.TXT
887_Anti_Primes.TXT

```

```

39769: 888_Sundial.TXT
39770: 890_Jensens_Device.TXT
39771: 892_microwaves.TXT
39772: 891_universal_turing_machine3.TXT
39773: 893_montecarlo_PI.TXT
39774: 894_csv_to_html.TXT
39775: 896_tickets_tree.png
39776: 896_all_tickets.csv
39777: 897_openoffice_ole_automation.TXT
39778: 899_remote_control_key.TXT
39779: 900_XML.TXT
39780: 901_turtle_purple_circle.txt
39781: 902_shal_explicit_aes.TXT
39782: 904_Unit1NEMESIS.pas
39783: 905_rfcl213util.pas
39784: 905_PlotCompCode.pas
39785: 905_net_ugraph_formplot.pas
39786: 905_rfcl213sys.pas
39787: 907_OverbyteIcsSHA1Test1.pas
39788: 880_cgiMain_server3.pas
39789: 880_CGI1.exe
39790: 908_Netlister.pas
39791: 909_sinewave_circle.pas
39792: 880_cgiMain_server4.pas
39793: 910_kh_function_plot_demo2.pas
39794: 910_tictactoe_form.png
39795: 911_gapfulnumbers2ibz.pas
39796: 912_TTCBlockSocket_Tester3.txt
39797: 729_U_BigIntTest_mx4.pas
39798: 913_al_web_spider_test.pas
39799: 915_OpenOffice_API.pas
39800: 916_fmTraceRouteMainU.pas
39801: 916_CGIMailerfMain.pas
39802: 918_TIdDateTimeStamp_main.pas
39803: 920_TURL_class_test.pas
39804: 920_TURL_class_test2.pas
39805: 919_uLockBox_Signatory_TestCasesmx471.pas
39806: 913_al_web_spider34form_trivial.pas
39807: 921_object_filehandling.pas
39808: 922_u_dir.pas
39809: 915_OpenOffice_API_2.pas
39810: 923_indybasicclientserverfMain.pas
39811: 925_U_DoubletsX3_2.pas
39812: 925_U_CountPhrases_form.pas
39813: 926_singlesamplepredict.py
39814: 927_Emirp.pas
39815: 927_Emirp2.pas
39816: 928_cd_catalog2.xml
39817: 927_Emirp2.pas
39818: 928_cd_catalog2.xml
39819: 930_service_tools.pas
39820: 930_service_tools2.pdf
39821: 930_service_tools2.htm
39822: 931_spiralmatrix2.txt
39823: 932_unixdict.txt
39824: 932_ordered_words42.txt
39825: 932_correlation.txt
39826: 932_correlation_regression.txt
39827: 548_STExpressions_Services2.TXT
39828: 749_helloWebServer3refresh2.txt
39829: 938_xmldemo_out.xml
39830: 937_xml_input.pas
39831: 935_Primality_by_trial_division.pas
39832: 933_draw_sphere.pas
39833: 940_primefactors.psb
39834: 941_picturebox.txt
39835: 942_gaussen.txt
39836: 942_gaussen_data_science.txt
39837: 942_statpackage2.png
39838: 942_statpackage2.png
39839: 942_gauss_jordan.pas
39840: 942_gauss_jordan2.pas
39841: 944_Perl_Regex.txt
39842: 945_Many_Regex.txt
39843: 946_sudokuform.png
39844: 945_registrysearch_processlist2.psb
39845: 947_TWinControl_Canvas.txt
39846: 949_setbit.txt
39847: 950_uniqueinstance_pythonversions.pas
39848: 950_saint_source.txt
39849: 185_memorymax3_internet.txt
39850: 942_gaussen_data_science3.txt
39851: 952_psUFinancial.pas
39852: 953_Res2BmpUtils.pas
39853: 953_JTools.pas
39854: 955_logregclassifier.py
39855: 957_blobstream.txt
39856: 958_webcam.txt
39857: 959_convert_temperature2.txt

889_Jaro_distance.TXT
891_uturing_machine.TXT
891_uturing_machine2.TXT
892_microwaves2.TXT
893_montecarlo_PI.psb
895_screenpixel.TXT
896_textclassification.py
690_calc_pi_ex_stat4.pas
898_exit_procedure.TXT
569_ServiceMgr3.TXT
XML.TXT
902_shal_explicit.TXT
903_ODBC_list.TXT
905_uipertools.pas
905_uipservices.pas
905_net_ugraph.pas
903_microtonal_csound.TXT
906_OverbyteIcsLogger.pas
170_4gewinnt_main3.txt
880_psvCGI.pas
880_CGI1.pas 880_DrBobCGI.pas
909_MandelBrot.pas
910_tictac.pas 910_tictac_form.pas
880_CGI1.exe
910_kh_plot.png
910_tictac_form2.pas
912_Primestest_AKS_2.pas
912_ssl_blksock_tester.pas
913_al_php_runner_test.pas
913_ALOpenOffice_Tester.pas
897_openoffice_ole_automation2.TXT
914_run_javascript_471.pas
917_EwbCoreTools_experimental.pas
919_uLockBox_HugeCardinalTestCases.pas
919_uLockBox_RSA_TestCases.pas
920_TimeServer_TickClock.pas
919_umfmMakeSampleKey.pas
913_al_web_spider34form.pas
922_u_stringlist.pas
922_u_c_url.pas
915_OpenOffice_API_21.pas
924_arrays.pas
925_U_CountPhrases.pas
925_phrasecount2.png
927_primetrial.pas
927_Emirp.psb
928_wave_async_parallel.pas
929_xmlstream.pas
928_wave_async_parallel.pas
929_xmlstream.pas
930_service_tools2.rtf
930_service_tools2.pas
931_spiralmatrix.txt
932_ordered_words4.txt
932_ordered_words4.psb
932_getgithub2.txt
932_correlationplot2.png
932_correlation_regression_light.txt
932_getgithub22.txt
938_xmldemo.xml
938_xml_output.pas
936_md5.pas
934_datamanipulation.pas
939_soundex.pas
940_primefactors.pas
941_picturebox2.txt
942_gaussen_data_science.psb
942_statpackage.png
942_gauss_jordan2.pas
942_gaussjordan.png
942_gaussjordan2.png
943_TEEChart.txt
945_Many_Regex.psb
946_sudokuUnit1Form.pas
945_registrysearch_processlist.txt
945_registrysearch_processlist2.txt
948_dynamic_page_control.txt
950_uniqueinstance.txt
950.ParserUtils.pas
950_saint_source.psb
185_memorymax4_internet.txt
951_dclublib.pas
952_psULib.pas
953_JConvertUtils.pas
954_jump_to_registrykey.txt
956_jscript.txt
958_foto2webcam.png
959_convert_temperature.txt
960_getsounddevices.txt

```



```

39858: 960_newjapan.mp3
39859: 960_neuralbitmx2.pas
39860: 960_Legendre_Gaussintegration.txt
39861: 950_saint_source2.txt
39862: 963_movie_reviewd_words42.txt
39863: 963_moviereviewswordlist2listboxsort.txt
39864: 861_stringtokenizer2primes_moviesreview2.txt
39865: 363_compress_services3.txt
39866: 964_geo_map_zoom.txt
39867: 965_RSA_IBZ_PrivatePublicKey_Form.pas
39868: 965_RSA_IBZ_PrivatePublicKey_Form3p.pas
39869: 966_GUItester2.png
39870: 967_U_RSADemo2.pas
39871: 967_CryptoGUItester4.png
39872: 967_CryptoGUItester1.png
39873: 968_recordobject.txt
39874: 969_gapfulnumbers.txt
39875: 970_integrate.txt
39876: 970_integrate_vandercorput_sequence.txt
39877: 970_U_wordladder_mX4.pas
39878: 970_book_portal_usecase2.png
39879: 970_skype_firewall.png
39880: 970_U_Invertedtext.uc
39881: 970_U_Decrypt3.pas
39882: 970_U_Decrypt4Forms.pas
39883: 971_U_DrawScrambledPie.pas
39884: 971_wordscramble_forms1.png
39885: 970_U_ScrambledPiemX4Forms2.pas
39886: 972_words_specinelist_hashdemocount.txt
39887: 972_json_tester.txt
39888: 972_char_counts.txt
39889: 972_json_tester2.txt
39890: 973_RCDATA_resource_export64.txt
39891: 470_U_ScrollingLEDs3mX4_21.pas
39892: 975_facofinteger.txt
39893: 975_U_PrimeFactors1.pas
39894: 975_U_PrimeFactors2.pas
39895: 975_euler_pot.txt
39896: 975_euler_pot2.psb
39897: 602_multilang_game2.txt
39898: 976_stripchars.txt
39899: 977_Rosetta_Dijkstra_Console.htm
39900: 978_wininet_universaldownload.txt
39901: 979_U_DensityPlot.pas
39902: 979_U_ReactionTimes4.pas
39903: 980_Chart_DensityPlot.pas
39904: 980_U_TelephoneWords.pas
39905: 980_telephonewords.png
39906: 980_U_TelephoneWords2.pas
39907: 980_U_TelephoneWords2.psb
39908: 980_telephonewords21.png
39909: 980_Chart_DensityPlot2.pas
39910: 982_win_graphviz.txt
39911: 984_Drawaclock_computergraphic.txt
39912: RXMisc.pas
39913: 985_nist.dat
39914: 985_open_subtitles.txt
39915: 985_shal_impl2.txt
39916: 985_shal_impl21.psb
39917: 985_missing_permutation2.txt
39918: 986_findallfiles.txt
39919: detector_jupyter.ipynb
39920: hoppenstedtstwinwiz_out2.jpg
39921: 987_shell32_operation.txt
39922: 988_anti_aliasing.txt
39923: 990_ADO_Connection.txt
39924: 990_GUI_ADO_Output.png
39925: 990_ADO_Connection_Create_DB.txt
39926: 990_ADO_Connection_Create_DB2.txt
39927: 990_metadefender_API.txt
39928: 990_metadefender_API_2.txt
39929: 990_httpsend.png
39930: 990_httpsend.pac
39931: 990_winhttprequest_API_51.txt
39932: 991_oma_chartregression2.py
39933: 991_python_oma_svr21.txt
39934: 991_oma_charts_market_btc_150y.png
39935: 840_RungeKuttaTest.exe
39936: 991_python_oma_svr21integrate_dotnetcore.txt
39937: 992_detector21_wget_integrate.txt
39938: 992_detector21_wget_integrate2.rtf
39939: 992_detector21_wget_integrate2.htm
39940: 993_perceptron_form.txt
39941: 993_perceptron_form2.txt
39942: 994_Find_URI_in_text.txt
39943: 995_RunLengthEncoding.psb
39944: 996_Precision_Functions.txt
39945: 998_string_has_all_unique_characters.txt
39946: 998_weatherapp474_forecast2.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_game2ball1.txt
602_multilang_gameball1.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_shal_impl.txt
985_regex_subtitles.txt
985_shal_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_maxbase4runtimepdb6.mdb
986_findallfiles_metadefender.txt
990_ADO_Connection_Create_DB21.txt
990_httpsend2.png
990_metadefender_API_21.txt
990_httprequest_API.txt
991_python_oma_svr.txt
991_python_oma_svr2.txt
991_python_oma_svr21integrate.txt
840_RungeKutta4test2.pas
991_pyoma_svr22integrate_dotnetcore2.txt
992_detector21_wget_integrate.htm
992_detector21_wget_integrate2.txt
993_perceptron_graph2.png
993_perceptron_graph.png
993_perceptron_form2.psb
995_RunLengthEncoding.txt
995_RunLengthEncoding_Pascal.txt
997_Base64_decode_data.txt
998_string_has_all_unique_characters.psb
weatherapp474.txt

```

```

39947: 999_word_frequency.txt
39948: 999_synedit_codecompletion.txt
39949: 1000_earth_rotation.txt
39950: 1000_coffeemax.bmp
39951: 1000_3DLabScreen.txt
39952: 961_URLDecoding_indyresponse_WMI_Info.txt
39953: 1000_GenomeLibrary.pas
39954: 1000_chr01_GEN_sequence.txt
39955: 1000_JVCL_Chart_Demo.PAS
39956: 1000_GraphicsMathLibrary_PantographF.PAS
39957: 1001_entropy_test.txt
39958: 1001_entropy_test21.txt
39959: 1002_protein_translation2.txt
39960: 1000_sphere_rotation3direct.txt
39961: 1000_sphere_sourceforge2.png
39962: 1000_under_overfit.png
39963: 1000_Sphere_4poles.png
39964: 1000_sphere_sourceforge2.txt
39965: 1003_molecules.txt
39966: 1003_molecules1_3.txt
39967: 1003bellsound.wav
39968: 1003templebell.wav
39969: 1003_molecules1_5.txt
39970: 1005_drawsphere_ascii.txt
39971: 1006_write_PPM_2.txt
39972: 1006_renderspherefull2.png
39973: 1006_writeSphere_PPM_2readBMPfix.txt
39974: 1006spheremaxsmallout2.png
39975: 1006_bitbang_spheremaxsmallout2w.png
39976: 1006_bitbang_spheremaxsmallout2y.png
39977: 1006_manmachine.ppm
39978: 1009_with_multiple_ranges.txt
39979: 1010_ScreenPixel2.txt
39980: 1010_ScreenPixel2.txt
39981: 1010_U_GraphicsEffects2.pas
39982: 1010_geometer_intersection2.png
39983: 1010_circle_point.png
39984: 1010_U_Geometry4Forms.psb
39985: 1010_U_Geometry4Forms.htm
39986: 1010_U_ClockAngles2.pas
39987: 1011_cInternetUtils.pas
39988: 1012_flcTimers.pas
39989: 1012_flcBits32.pas
39990: 1013_saint_source23tests.txt
39991: 210_public_private_cryptosystem6_ibz_tuned.txt
39992: 1014_jcl_stringlist2.txt
39993: 1015_Scripting_Dictionary.txt
39994: 1016_newsfeed_sentiment.py
39995: 1016_sentiment_api2_bbc_newsfeed.txt
39996: 1017_XmlDocRssParser.pas
39997: 1017_bbcnews2.xml
39998: 1017_RSSExportUnit2Webform.pas
39999: 1018_Twelve_statements.txt
40000: 1018_Twelve_statements2.txt
40001: 1019_math_source_cap5.pas
40002: 1019_math_learn_tool.png
40003: 1020_FASTA_format.txt
40004: 1020_Power2FirstDigits.psb
40005: 1020_TopRankPerGroup.txt
40006: 1020_CustomSort_Stringlist_Department.txt
40007: 1021_uCEFMiscFunctions_mX4.pas
40008: 1022_U_SciGraph13_mX47.pas
40009: 1023_FANN_NeuralNetworkLogTrainer.txt
40010: 1024_floydalgo.txt
40011: 1025_Deranged_anagrams2.txt
40012: 1026_json_automation_coviddata.txt
40013: 1026_json_automation_coviddata2.txt
40014: 1026_worldwide5zoom.png
40015: 1026_worldwide4.png
40016: 1026_worldwide3.png
40017: 1026_worldwide.png
40018: 1026_swiss_weeklypattern.png
40019: 1026_swiss_firstwave.png
40020: 1026_jsondos_forecast.png
40021: 1026_json_automation_refactor.txt
40022: 1026_worldwide9_8confirmed.png
40023: 1017_RSSDataCampSSLStreamSentiment.pas
40024: 972_json_tester31.txt
40025: 1026_uJson_testeroauth_print.pas
40026: 1027_fpvectorial.jpg
40027: 1028_agg_math.pas
40028: 1028_JcfUnicodeFiles.pas
40029: 1028_filestream4test.pas
40030: 1028_filestream4test2.pas
40031: 1030_filepilot2.psb
40032: 1030_StrUtil.pas
40033: 1026_SVC_BitVision_MLPrimer.py
40034: 1031_Bitcoinaddress_validation.txt
40035: 1032_population_count.txt

999_word_frequency2.txt
999_RotImg.pas
1000_GraphicsMathLibrary.PAS
1000_earth_rotation3d.txt
1000_GraphicsMathLibrary_Pantograph.PAS
1000_graph2.pas
1000_Coordinates222.txt
1000_Coordinates_Gen.ods
1000_JVCL_Chart_Demo2.PAS
1000_JVCL_Chart_Demo2Form.PAS
1001_entropy_test2.txt
1002_protein_translation.txt
1000_sphere_rotation3d.txt
1000_Sphere_big.png
1000_sphere_sourceforge.png
1000_Sphere_small.png
1000_sphere_sourceforge.txt
1000_Nap.bmp
1003_molecules1_2.txt
1003_molecules1_4.txt
1003doorbell.wav
1003_sound_molecules.png
1004_wordwheel.txt
1006_write_PPM.txt
1006_write_PPM_2read.txt
1006_writeSphere_PPM_2readBMP.txt
1006_writeSphere_PPM_2readBMPfix2.txt
1006_writeSphere_PPM_2readBMPfix2frm.txt
1006_bitbang_spheremaxsmallout2r.png
1007_disabled_bitmap.txt
1008_Knuth_shuffle.txt
1010_ScreenPixel.txt
1010_JPGClock.txt
1010_ProcsVecteurs.pas
1010_geometer_tangent.png
1010_circle_tangent.png
1010_geometer_polygon.png
1010_U_Geometry4Forms.pas
1010_U_ClockAngles2.htm
1010_U_DecToFract2.pas
1011_flcStatistics_mX4.pas
1012_flcCharSet.pas
1010_U_DecToFract21.pas
unittests4.txt
1014_jcl_stringlist.txt
1014_jcl_stringlist21.txt
1015_Scripting_Dictionary_RegEx.txt
1016_newsfeed_sentiment.htm
1016_sentiment_api2_bbc_newsfeed2.txt
1017_bbcnews.xml
1017_XmlDocRssParser_BBCNewsDemo.pas
1017_RSSExportUnit2formSSLStream.pas
1018_ludic_numbers.txt
1019_math_source_cap5strain.pas
1018_Twelve_statements21.txt
1020_perlinNoise.txt
1020_FASTA_format2.txt
1020_Power2FirstDigits.txt
1020_ReST_Countries_API.txt
1020_CustomSort_Stringlist.txt
1021_uCEFMiscFunctions_mX47.pas
1022_scigraphplot.png
1023_FANN_Alllogic2.png
1025_Deranged_anagrams.txt
1026_SVC_BitVision.py
1026_coviddtimeserie.txt
1026_json_automation_coviddata21.txt
1026_worldwide5.png
1026_json_automation_coviddata22.txt
1026_worldwide2.png
1026_US_firstyear.png
1026_germany_weeklypattern.png
1026_jsondos_forecast2.png
1026_json_automation_coviddata22teechart.txt
972_json_tester3.txt
1026_json_automation_refactor2.txt
1017_RSSDataCampSSLStreamSentiment2.pas
972_json_tester32.txt
1027_fpvutils.pas
1028_STRandomSystem_tester32.txt
1028_JcfMiscFunctions.pas
1028_JcfMiscFunctions2.pas
1028_filestream5test.pas
1030_filepilot2.txt
1030_filepilot21.txt
1030_StrUtil2.pas
1026_SVC_BitVision_MLPrimer2.py
1031_Bitcoinaddress_validation.txt
1032_population_count.psb

```

```

40036: 1033_Corpus_from_Dir.py
40037: 1034_tamath.pas
40038: 1034_tamath_safe.pas
40039: 1035_covidapp.txt
40040: 1035_covidapp3.htm
40041: 652_graph3Dspiral2.pas
40042: 1037_bufferoverflow_tester.txt
40043: 1035_covidapp31.txt
40044: 1039_pcatest.pas
40045: 1039_pcatest3.pas
40046: 1039_roman.pas
40047: 1016_newsfeed_sentiment_integrate2.py
40048: 1016_newsfeed_sentiment_integrate2.htm
40049: 1041_signproof_tester.txt
40050: 1042_superpermutation_minim.txt
40051: 1034_weather_report_py.png
40052: weatherbox.txt
40053: 1045_XMLTreeview2.png
40054: xmldemo.txt
40055: 1045_semaphoregrids.pas
40056: weatherapp475.txt
40057: 1046_openweather_integrate.txt
40058: 1046_openweather_P4D_2.png
40059: 1046_JclTimeZones.pas
40060: 1047_microtonal_tomath_mX4.txt
40061: 1047_microtonal_tomath_mX4_2.txt
40062: 1049_InetUtils_Callback.TXT
40063: 1050_saint_Pythonengine.txt
40064: 1050_pydemo2.txt
40065: 974_64bit_Routines2.txt
40066: 1051_semaphoregrid.txt
40067: 778_advapi32_dll_SHA512_plusRefactor.txt
40068: 1052_delphi_arduino2.txt
40069: 1052_delphi_arduino2.png
40070: 1052_delphi_arduino3.txt
40071: 1052_delphi_arduino31.txt
40072: 1052_delphi_arduino32.ini
40073: 1052_bodmas.txt
40074: 1050_pydemo_5_powerliners.txt
40075: 1052_OoMiscpas2PyHTTPServer2LAB.txt
40076: 1052_pyhttpserver_integrate22.py
40077: 1053_U_TimeZoneDemo2.pas
40078: 1054_U_SoundGen31.pas
40079: 1054_U_SoundGen_mcisendstring32.pas
40080: 1055_U_IntListtest.pas
40081: 1055_tintlistdemo.png
40082: 1052_pydemo_pythonversion.txt
40083: 1056_neuralvolume_routines.pas
40084: 1057_pydemo_7_pyIO.txt
40085: 1058_Arduino.pas
40086: 1060_pypas_demo10.psb
40087: 1059_pypas_demo11.psb
40088: 1059_pypas_demo11.txt
40089: 1060_arduinoUnit1.pas
40090: 1062_utils.py
40091: 1061_gameoflifeUNIT1.PAS
40092: 1062_intervallunit2.txt
40093: 87_1062_neuraltrainer.png
40094: 1062_neuralnet_vectors2.txt
40095: 1064_httpd24.pas
40096: 1065_neuralnetwork_CAI.txt
40097: 1066_KMeans_tester.txt
40098: 1067_py_planet_moon.txt
40099: 1068_vector_matrix_tester.txt
40100: 1068_floattostr_tester.txt
40101: 1063_array_lists3.txt
40102: 1070_U_PerfectDeck.pas
40103: 1070_tprime_tshirt.png
40104: 1070_U_Keno2mX4.pas
40105: 1065_CAI_2_SimpleImageClassifier.txt
40106: 1065_CAI_2_SimpleImageClassifier2.txt
40107: 1071_Newadds_V47590_Modules.pas
40108: 1070_tshirt_prime2_tutor88.txt
40109: 1071_Newadds_V47590_Modules.pas
40110: 1065_CAI_2_SiImageClassifier21_Tutor_89_test3.txt
40111: 1072_captcha2.txt
40112: 1073_CAI_3_LearnerClassifier22_Tutor_89_2.txt
40113: 1074_CAI_3_visualautoencoder2.txt
40114: 1074_U_Systools_mX4Similarity.pas
40115: 1075_primes30_performance5.txt
40116: 1076_CAI_Ugradientascent.pas
40117: 1076_CAI_UVisualGAN2tester_MNIST.pas
40118: 1077_U_Catapult4.pas
40119: 1078_create_DB.pas
40120: 1077_U_Catapult42forms.pas
40121: 1077_U_metronome2.pas
40122: 1079_Fregan.pas
40123: 1077_U_Catapult43forms2.pas
40124: 1080_trend.pas

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_Opq_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
1050_pytest21_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
1061_neuraldatasets_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
1066_KMeans_tester2.txt
1070_U_Keno2mX4Form.pas
1070_U_Keno2mX4_Forms2.pas
1070_U_PrimesFromDigits.pas
1071_Newadds_V47590_Modules2.pas
1072_captcha.txt
weatherapp476.txt
1074_CAI_3_visualautoencoder.txt
1071_Newadds_V47610_Modules3.pas
1074_U_Systools_mX4Similarity2.pas
1065_CAI_2_SimpleImageClassifier21_EKON_blogstarter.txt
1076_CAI_UVisualGAN2.pas
1076_CAI_UVisualGAN2tester_MNIST2.pas
1077_U_Catapult41.pas
1077_U_Catapult41forms.pas
1077_U_Catapult43forms.pas
1078_bell_numbers.pas
1079_Test.txt
1077_U_metronome21.pas
1080_hist.txt

```

```

40125: 1080_hist.exe
40126: 1080_tprob.pas
40127: 1080_VCLScanner2021.txt
40128: 1080_syndat_concept.png
40129: 1081_syndat_pydemo32_2.psb
40130: 1082_Idiomatically.txt
40131: 1082_ToolsUtils2.pas
40132: 1083_Deltics_TCommandline.pas
40133: 1084_DelticsStrUtils.pas
40134: 1085_kmemo_dictionary.pas
40135: 1086_bigdecimals2.pas
40136: 1086_bigdecimals2python2.pas
40137: 1088_CAI_3_Hypotenuse.txt
40138: 1088_CAI_3_DenseNetBCL40_2.txt
40139: 1089_RSA_Example_crypto.htm
40140: 1090_simple_neuralwebserver.txt
40141: 1090_simple_neuralwebserver2.txt
40142: 1090_uformlamwinosettingspaths.pas
40143: 1090_Fractran_test_Python.txt
40144: 1090_U_Numbrix2.pas
40145: 1090_Numbrix2.txt
40146: 1090_numbrix_GUI_24.png
40147: 1091_U_SpringMass21.pas
40148: 1091_U_SpringMass2.pas
40149: 1092_Rollercoaster_Simulator21.pas
40150: 1093_XMLUtils_Tutor92tester.txt
40151: 1095_faker_tutor_PAS2JS.txt
40152: 1096_VCL4Python.pas
40153: 1096_MainFormSVG_2test4Python.pas
40154: 1096_pychats_demo4VCL4Python.png
40155: 1096_MainFormSVG_2test4Python_VCL4Python.pas
40156: 1100_901_turtle_purple_circle_EKON23.txt
40157: 1100_U_Compass.pas
40158: 1097_Pierpont_primes.pas
40159: 1098_image_noise.pas
40160: 1098_image_noise2xmlnodes.pas
40161: 1001_ESP32_Loop.TXT
40162: 1102_matrixmultiplication4.txt
40163: 1102_matrixmultiplication41.txt
40164: 1101_PJConsole1.txt
40165: 1104_TNeuronClass.txt
40166: 281_picturepuzzle31res.txt
40167: 1101_PJConsoleApp_Echoer.png
40168: 1101_PJConsole12Json.txt
40169: 1105_HTML_Builder2DB.txt
40170: 1101_PJConsole1ex2.txt
40171: 1105_HTML_Builder2Modules.txt
40172: 1107_Iris_Console.pas
40173: 1107_Iris_Console2.pas
40174: 752_U_AstroDemo_mX4_41.pas
40175: 1108_CAI_Geocoding.pas
40176: 1109_winapidownload.pas
40177: 1111_lifeappearance_Unit1.pas
40178: 1111_UVPlanitest_QR_Code.pas
40179: 1071_Newadds_V47610_Modules47.pas
40180: 1012_UTicTacToeCount.pas
40181: 1108_CAI_GeocodingTwininetAPI.pas
40182: 1112_lastUtils.pas
40183: 100U_CoinSearch2.pas
40184: 1111_animalmachine_pictures12.pas
40185: 1011_graphnodes_solution.png
40186: 1011_graphnodes_solution1.png
40187: 1112_U_Wordgrid.htm
40188: 1112_wordgrid1.png
40189: 1113_composition_tree.pas
40190: 1071_Newadds_V47610_Modules47tester.pas
40191: 1114_polynomial_synthetic_division.txt
40192: 1115_unitKWKalender.pas
40193: 1088_CAI_3_supersimplecorrelation.txt
40194: 1115_gecodingapi_tee_helvetia_json.txt
40195: 1117_sunpos.png
40196: 1117_U_SolarPos2Bern.pas
40197: 1117_sunpos_bern2.png
40198: 1118_Cumulative_standard_deviations.pas
40199: 1108_CAI_GeocodingTwininetAPI_Maponly.pas
40200: 1119_FloatToBinTests.pas
40201: 93_1108_CAI_GeocodingTwininetAPI_Maponly.pas.png
40202: 1120_DammAlgorithm12.pas
40203: 1121_sentiment_api3_bbc_newsfeed4rec.txt
40204: 1121_sentiment_api3_bbc_newsfeed4rec2.txt
40205: 1121_sentiment_api3_bbc_newsfeed4rec21.rtf
40206: 1122_langtranslator1.txt
40207: 1123_IntegerArraySort.pas
40208: 1123_CODEWIZU.PAS
40209: 1124_CODEReflection.PAS
40210: 1124_CODEReflection1.PAS
40211: 1125_unitGlobal1.pas
40212: 1125_unitGlobal13.pas
40213: 1126_Projectile_Physics.pas
1080_histdatgui.png
1080_fmTraceRouteMainU.pas
1080_VCLScanner2021.uc
1081_syndat_pydemo32_2.txt
781u_LatLonDistanceTestmX21.pas
1082_ToolsUtils.pas
1071_Newadds_V47610_Modules42.pas
1084_SeAES256_explain.pas
602_moonbug25.txt
1085_kmemo_dictionary2taunumber.pas
1086_bigdecimals2python.pas
1087_HQ9.pas
1088_CAI_3_DenseNetBCL40.txt
1050_pydemo_5_powerliners_def_2.txt
1089_RSA_Example_crypto.txt
1089_RSA_Example_crypto_Herd.txt
1090_uarduinoformworkspace.pas
1090_Fractran_test.txt
1090_U_Splines.pas
1090_numbrix_GUI.png
1090_U_Numbrix21.pas
1090_Numbrix2data.txt
1091_springmassdemo.png
1092_Rollercoaster_Simulator.pas
1092_Rollercoaster_Simulator21.htm
1094_pydemo13tutor_PAS2JS.txt
1096_MainFormSVG_2.pas
1096_P4D_VCL4Python.pas
1093_CAI_IdentityShortcutConnection2.pas
1096_standalonepyvcl.exe.png
1096_MainFormSVG_2test4Python_VCL4Python2.pas
1100_VectorProduct.pas
1100_U_Compass2.pas
1097_Pierpont_primes2.pas
1098_jaro_similarity.txt
1099_rest_localdb.pas
1069_pydemo30_matrixperformance4.txt
781u_LatLonDistanceTestmX22.pas
1101_PJConsole.txt
1103_DudsCommonUtils.pas
1104_TNeuronClass2.txt
1101_PJConsole12.txt
Echoer.exe
1105_HTML_Builder.txt
1101_PJConsoleApp_Echoer2.png
1101_PJConsole1_demo9.txt
1106_OnlineRandomizer.pas
1107_bezdekIris.fann.data.txt
1106_OnlineRandomizer_Random_Main.pas
752_astronomy_moon23.png
1108_geocoding_api.png
1110_JSONObject.txt
1111_uvisualcifar10animalmachine.pas
1111_URSOM_Network.pas
100U_CoinSearch.pas
1012_delphi-utils.pas
1108_CAI_GeocodingTwininetAPI_only.pas
1112_lastUtils2.pas
1111_machineanimal.png
1011_U_CoinSearc_form.pas
1011_U_CoinSearch_form.pas
1112_U_Wordgrid.pas
1112_wordgrid.png
1112_full.dic
1113_composition_tree1.pas
ubuntu16_session_maxbox4_cloud2022_full12.png
1115_Wireworld.txt
1115_scriptDatenbankdiagnose.pas
1088_CAI_3_supersimplehyperbolic tangent.txt
1116_U_ChallengingmathTeasers.pas
1117_sunpos_bern.png
1117_U_SolarPos2.pas
1117_U_SolarPos2Bern2.pas
1118_Cumulative_standard_deviations.htm
1117_StAstro.pas
1120_Department_numbers.pas
1120_DammAlgorithm.pas
sentiment3.txt, sentiment4.txt
1121_sentiment_api3_bbc_newsfeed4.txt
1122_umain.pas
1121_sentiment_api3_bbc_newsfeed4rec21.txt
1122_langtranslator1.txt
1123_MATHBOX.PAS
1123_CODEWIZU_2.PAS
1125_Forward_Difference.pas
1125_unitGlobal.pas
1125_unitGlobal12.pas
1100_U_Compass1_deltics_sysutils.pas
1126_Projectile_Physics2.pas

```

```

40214: 1126 Projectile_Physics21.pas
40215: 1126 fetchapi1.png
40216: 727_mondrian_next.png
40217: 727_Canvas_Artist_mondrian_gen.TXT
40218: 1127_RegistryWatchDog.pas
40219: 1128_fpvutils.pas
40220: 1129_urlscan_api_json.txt
40221: 1130_Scanner_gen21.TXT
40222: 1130_mondrian_QRCode_P2.png
40223: 1130_urlscan_api_json1_geoip.txt
40224: 1131_rosetta_numerical_integration.txt
40225: 1133_klib_utilities.txt
40226: 1134_syn_dat_pop.txt
40227: 1134_Cifar10Resize_win_lessthread.pas
40228: 1135_Cifar10SeparableConvolution_50.pas
40229: 1136_Chowla_numbers1.txt
40230: 1135_visualCNN5.png
40231: SimpleSeparableImageClassifier124_50_21.nn
40232: 1138_ocr_tesseract.txt
40233: 1138_bmp.bmp
40234: 1140_price_fraction.txt
40235: 1140_Statistics1.pas
40236: 1141_adafruit_tempsensor21.txt
40237: 1140_Om_AstronomicalAlgorithms.pas
40238: 1140_quaternion_3D.pas
40239: 1142_anti_primes.pas
40240: 1142_list_virtualconstructors_test.pas
40241: 1142_list_collections_pydemo42.txt
40242: 1143_KLibVC_Redist.pas
40243: 1071_Newadds_V47610_Modules48.pas
40244: 1145_271_closures_study_op_sys_tutor97.txt
40245: 1144_AzuluaUtils_mX4.pas
40246: 1146_filesize_distribution.txt
40247: 1148_julia_set.txt
40248: 1148_julia_set.bmp
40249: 1150_DigitalRoot.txt
40250: 1150_pydemo42httpheader1.txt
40251: 1150_Partition_function_P1.txt
40252: 1151_sentiment_api4simple.txt
40253: 1151_rapid_api_protol.txt
40254: 1152_dos_output_callback.txt
40255: 1151_rapid_api_proto_googleapi12.txt
40256: 1153_space_invaders3.txt
40257: 170_4gewinnt_main21.htm
40258: 1142_list_collections_pydemo42_1.txt
40259: 1154_energy_api_agsi_plot.txt
40260: 1154_energy_api_agsi_plot12.txt
40261: 1155_census_data.txt
40262: 1155_Utills47620.pas
40263: 1156_SearchUtils.pas
40264: 1058_teetest_screen21map.TXT
40265: 1159_uvisualgan_mX4.pas
40266: 1159_uvisualgan_mX4_1.pas
40267: 1161_wmiserv_commandlinelist.txt
40268: 1162_TIXML.TXT
40269: 1063_cryptosystem2testcrypto3.txt
40270: 1163_tcptable_dll.txt
40271: 1164_filelessStream.txt
40272: 1165_UAC_enabled.txt
40273: 1167_vcluster_py.txt
40274: 1168_teechart_demo3.txt
40275: 1169_test_postrequest2.txt
40276: 1169_test_postrequest2.txt
40277: 1169_resource_loader2postmethod.pas
40278: 1170_UUtils_mX4.pas
40279: 1170_UUtils_mX4_tester2.pas
40280: 1170_wdc_GetRemoteFileSize.txt
40281: 965_RSA_IBZ_PrivatePublicKey_Form4p_eng.pas
40282: 1154_energy_api_agsi_plot14wininet.txt
40283: 1171_powershell_dos_output.txt
40284: 534_arduino_cockpit3radar.TXT
40285: 1172_wininet_intftester.png
40286: 1172_ict_wininet_interface.psb
40287: 1174_mx4_cd_player2.jpg
40288: 1174_cdmain.txt
40289: 1174_mx4_cd_player2_1.jpg
40290: 1175_DRAG1.PAS
40291: 1050_pydemo35_signal.txt
40292: 1176_APILayer_Demo.txt
40293: 1175_DRAG1_Game.PAS
40294: 1177_increment_numerical_string.txt
40295: 1179_interactive_jupyter.txt
40296: 1179_interact_jupyter_notebook.png
40297: 1180_restcountries_API_2.txt
40298: 1180_jsonstruct_tutor104.png
40299: 1180_restcountries_API_21.txt
40300: 1181_MyFunc_.pas
40301: 1182_biglog_numeric_string.txt
40302: 1182_UCBUtils.pas
1126_projectile2.png
1126_fetch_api1.txt
727_mondrian_next21.png
727_Canvas_Artist_mondrian_gen2.TXT
727_Canvas_Artist_mondrian_gen21.TXT
1128_mean_angle1.txt
1129_urlscan_api_json1.txt
1130_mondrian_QRCode_P.png
1130_Scanner_QR_21.TXT
1130_urlscan_api_json1_geoip1.txt
1132_rosetta_Farey_sequence.txt
1133_klib_utilities2.txt
1134_Cifar10Resize.pas
1135_Cifar10SeparableConvolution.pas
1136_hamming_numbers1.txt
1135_uvisualcifar10test_mX4_1.psb
1135_uvisualcifar10test_mX4_1.pas
1137_gauss_legendre.pas
1138_testmap2.png
1139_pentagram.txt
1140_Statistics.pas
1140_USolarSystem.pas
1140_PlanetData.pas
1140_doubleVector3D.pas
1141_fast_sin.pas
1139_pentagram1.txt
1142_setbit_comparison.pas
1140_doubleVector3D_collection.pas
1143_KLibVC_Redist2.pas
1071_Newadds_V47610_Modules49.pas
1142_list_collections_pydemo42.txt
1145_271_closures_study_op_sys_tutor97_1.txt
1147_practicalnumbers_2.txt
1148_julia_set.png
1149_sum_to_100.txt
1150_pydemo42httpheader.txt
1150_Partition_function_P.txt
1151_rapid_api_proto.txt
1151_sentiment_api4simple1.txt
1150_pydemo42httpheader2openssl1.txt
629_Word_FORMULA2super.PAS
1152_shell_composer.txt
170_4gewinnt_main21.txt
170_4gewinnt_main31.txt
1154_energy_api_agsi.txt
1154_energy_api_agsi_plot1.txt
1154_energy_api_agsi_plot13.txt
1155_census_data_graph.txt
1155_dynmatrix47620.pas
1157_wininet_internetreadfile_dll.pas
1158_resource_loader2.pas
1159_SearchUtils_Searchcode.pas
1160_google_safebrowsingapi12.txt
1161_workshop_WMI_Proc32.png
1161_templatec_hack1_meterpreter.txt
1161_windefend_check.txt
845_TestPermutations3.txt
1164_filelessStream2.txt
1166_farey_sequence.txt
1167_penguins_data_graphtester_py.txt
1168_teechart_demo31.txt
1169_test_postrequest3.txt
1169_test_postrequest3.txt
1168_teechart_demo31.txt
1071_Newadds_V47610_Modules50.pas
1170_uRegistry.pas
1170_wdc_GetRemoteFileSize2.txt
965_RSA2_4.png
1154_energy_api_agsi_plot14wininet2.txt
1171_ict_cryptohack.txt
965_RSA_IBZ_PrivatePublicKey_Form4p_eng.pas
1172_ict_wininet_interface.txt
1173_quora_mathquest_utilities.txt
1174_cdmain_form.txt
1174_multimedia_cdplayer_docu.pdf
1174_cdmain_form2.txt
1174_cdmain_form21.txt
1050_pydemo35_4signals.png
1176_APILayerDemo.png
1177_rate_service1.txt
1178_student_t-test.txt
1179_interact_jupyter_notebook1.png
1180_restcountries_API.txt
1180_restcountries_API_2.htm
1180_osm_map_tutor104.jpg
1180_jsonstruct_validator104.png
1181_MyFunc_1.pas
1182_bigfunctions_v47620.txt
1071_Newadds_V47620_Modules90.pas

```



```

40303: 1182_UCBUtills1.pas
40304: 1154_agssi_plot14stacked.png
40305: geocoding4.txt
40306: 1183_word_fractal.txt
40307: 1183_word_fractal1.txt
40308: 1183_WordFractalblue.png
40309: 1183_word_fractal2turtle.txt
40310: 1183_word_fiboturtle2prime.txt
40311: 1183_word_fiboturtle_AB_Animation.txt
40312: 1135_classify_cifar10images2laz.pas
40313: 1135_classify_cifar10images1_5.psb
40314: 1184_tau_function.txt
40315: 1186_Casting_out_nines.txt
40316: 1188_fpsmeter.pas
40317: 1188_KOL_Routines.pas
40318: 667_URobo2EKON_FUN_Tracking2mapbox5.pas
40319: 1189_Distance_and_Bearing.pas
40320: 1189_airports.dat
40321: 1189_Distance_and_Bearing_Sydney.pas
40322: 1189_Distance_and_Bearing_Bern.pas
40323: 1189_heading_bern_belp.jpg
40324: 1189_Distance_and_Bearing_Bordcomputer.pas
40325: 1190_Continued_fraction.txt
40326: teeposter_20230227_115631.jpg
40327: 1032_population_count1.txt
40328: 1191_popcnt4.txt
40329: 1193_13th_root.txt
40330: 1194_ascendingprimes.txt
40331: 1194_ascendingprimes_py2_lin3.txt
40332: 1090_usuperresolutionapp2.pas
40333: 1197_Biorhythmus_Go.txt
40334: 1197_Biorhythmus_Go_Graph.txt
40335: 1197_Biorhythmus_Go_Graph2.txt
40336: 1198_pacmanMAIN.PAS
40337: 1198_pacmanMAIN12.PAS
40338: pacmanmaxbox12.png
40339: 1200_GCD.txt
40340: 1198_pacmanMAIN13.PAS
40341: 1200_GCD1.txt
40342: 1201_Prisoners100.txt
40343: 1202_Google_Translate_API2.txt
40344: 1202_chatgpt_API31.txt
40345: 1202_chatgpt_maxboxscript.png
40346: 1202_chatgpt_API33.txt
40347: 1202_chatgpt_API35.txt
40348: 1202_TRestClient_pac.png
40349: 1203_multiplicationtable.txt
40350: Parent_folder
40351: 1202_chatgpt_API37tester.txt
40352: 1203_multiplicationtable1.txt
40353: 1205_GpTimezone.pas
40354: 1206_geocoding4tripadvisor.txt
40355: 1207_cryptosystem2test_bigpi.txt
40356: 1207_cryptosystem2test_bigpi2euler12.txt
40357: 1207_bigeuler_bigpi.png
40358: weatherapp477.htm
40359: 1208_factorial_primes.txt
40360: 1202_chatgpt32.txt
40361: 1210_halt_and_catch_fire.txt
40362: 1210_cuberoot.txt
40363: 1210_jvprocess_test.txt
40364: 1202_chatgpt32_prompting.txt
40365: 1212_utf8getnextchar.txt
40366: 1212_FPCTypes.pas
40367: 1212_breakoutUnit1.pas
40368: 1213_breakoutUnit_main_0_3.pas
40369: 1214_java_process_run2.txt
40370: 1215_uFrm_PersonList.pas
40371: 1216_JNIUtils.pas.txt
40372: 1217_count_coin.txt
40373: 1198_pacmanMAIN14_2.PAS
40374: 1219_pydemo47_GUIAutomation.txt
40375: 1220_GUIUtils.pas
40376:
40377: https://softwareschule.code.blog/2023/04/01/how-to-chat-with-gpt/
40378:
40379: 685_uPSI_ALQuickSortList; //alcinoe for list of double fast
40380: with TALDoubleList.create do begin //singlelist
40381:   with CL.AddClassN(CL.FindClass('TALBaseQuickSortList'),'TALDoubleList') do begin
40382:     Function IndexOf( Item : Double) : Integer';
40383:     Function IndexOfObject( AObject : TObject) : Integer';
40384:     Function Add( const Item : Double) : Integer';
40385:     Function AddObject( const Item : Double; AObject : TObject) : Integer';
40386:     Function Find( item : Double; var Index : Integer) : Boolean';
40387:     Procedure Insert( Index : Integer; const item : Double);
40388:     Procedure InsertObject( Index : Integer; const item : Double; AObject : TObject);
40389:     RegisterProperty('Items','Double Integer', iptrw);
40390:     SetDefaultProperty('Items');
40391:     RegisterProperty('Objects','TObject Integer', iptrw);
1154_energy_api_agssi_plot14_wininet2stacked.txt
geocoding3.txt
geocoding4.htm
1183_word_fractal.htm
1183_WordFractalyellow.png
1183_word_fractal2pi.txt
1183_word_fiboturtle.txt
1183_word_fiboturtle_AB.txt
1135_classify_cifar10images2tutor4.pas
1135_classifiergui_tutor105.png
1135_classify_cifar10images1_5.pas
1185_SumOfASeries.txt
1187_format_basics.txt
1188_KOLNKDir.pas
1188_KOL_Routines2.pas
667_roboline_outdoor_layer5.bmp
1189_Distance_and_Bearing.psb
1189_Distance_and_Bearing_Bern.htm
1189_Distance_and_Bearing1.pas
1189_heading_bern_neuchatel_airport.jpg
1189_Distance_and_Bearing_Bordcomputer.psb
1189_Distance_and_Bearing_Bordcomputer2.pas
teeposter_model_20230227_150518.jpg
1190_Arithmetic_Numbers.txt
1032_population_count1_unittest.txt
1192_semiprimes.pas
1194_ascendingprimes_py.txt
1194_ascendingprimes_py2.txt
1195_sourceforge_download_python3_lin3.txt
1196_APILayer_Demo1UnicodeEscape.txt
1197_Biorhythm.png
1197_Biorhythm2.png
1197_Biorhythm21.png
1198_pacmanMAIN1.PAS
1198_pacmanMAIN12.htm
Pac_Man_Package.zip
1199_funtest.txt
1200_pyheart2.png
1200_pyheart21.png
1201_Prisoners100.psb
1202_chatgpt_API3.txt
1202_chatgpt_playground.png
1202_chatgpt_API32.txt
1202_chatgpt_API34.txt
maxbox4tutor110demos.zip
1202_chatgpt_API37.txt
1203_multiplicationtable1.txt
unit_uPSI_RestClient.pas
1202_chatgpt32.txt
1204_isbn_checksum.txt
1206_GpString_mX4.pas
1206_geocoding4tripadvisor.psb
1207_cryptosystem2test_bigpi1.txt
1207_bigeuler_bigpi2.png
1207_bigeuler_only.txt
weatherapp477.txt
1202_chatgpt_API37tester_goldbach.txt
1209_MAC_vendor_lookup.txt
1044_queens_performer3_81.txt
1044_queens_performer3_81_py.txt
1211_workdaysAPI.txt
1208_factorial_primes1queens_heatmap.txt
1212_pas2jsutils.pas
1071_Newadds_V47650_Modules100.pas
1212_breakoutUnit12.pas
naples3mX4mapbox_weatherzoom4.png
1214_java_process_run.txt
1215_uFrm_PersonList2.pas
1216_JNIUtils2.pas.txt
1218_TProcess2.txt
1214_java_process_run22.txt
1219_pydemo47_GUIAutomation2.txt
1220_GUIUtils2.pas

```

```

40392:     end;
40393:
40394: https://sourceforge.net/projects/maxbox/files/Examples/13_General/
40395: http://www.softwareschule.ch/examples/signtoolbatch.txt
40396: http://www.softwareschule.ch/examples/signtoolbatch.htm
40397:
40398: https://cdn.top4download.com/img/award_120x60_pick.gif
40399: http://www.kleiner.com/boxart.htm
40400: http://max.kleiner.com/boxart.htm
40401: http://www.kleiner.ch/kleiner/boxart7.htm
40402:
40403: All 35 Articles of maXbox Pascal in Blaise Pascal Magazine!
40404: -----
40405: Using Examples and Sources from Magazine CatalogX
40406: Art of Coding V.31/32 5/6 2013
40407: DLL for All V.33
40408: QR Code Coding V.34
40409: GEO Maps Nav V.35
40410: 3D Printing Lab V.36
40411: RegEx Report REX V.37/38
40412: Func Testing V.40
40413: Arduino Coding V.45/46
40414: Time is on my side V.48
40415: Compute Big Numbers V.52
40416: Code Metrics V.57
40417: Work with WineHQ V.60
40418: Work with Win WMI V.63
40419: Work with Win WMI II V.64
40420: Work MS CryptoAPI V.65
40421: The Perceptron! V.72
40422: No GUI Shell V.80
40423: Machine L. with CAI V.87/88
40424: Portable Pixmap PPM V.90
40425: Fundamentals Lib V.91
40426: RSS Feed of BBC News V.92
40427: RSS Feed Sentiment V.93
40428: JSON4Pascal V.94/95
40429: Python4Delphi 96
40430: Python4Delphi II 97
40431: Python4maXbox III 98
40432: Image Classification for Lazarus 99/100 !
40433: Faker SynDat 101
40434: Catapult, Coaster Simulator, VCL4Python 102
40435: Geocoding II 103/104
40436: CIFAR 10 Image Classifier 105
40437: Energy_API agsi_Plot 106
40438: Cluster Classify 107
40439: Penguin Data Science SVG Plot 108
40440: Image2Text API V109
40441: Cryptobox API RESTCountries Laz Classifier V110
40442:
40443: Proposal for Boolean Logic Abbreviation Symbol:
40444: -----
40445:
40446: 01 FALSE //Contradiction
40447:
40448: 02 AND //Conjunction x*y
40449:
40450: 03 INHIB //Inhibition x^y
40451:
40452: 04 PRAEP //Praependence x
40453:
40454: 05 PRAE //Praesection ^x*y
40455:
40456: 06 POST //Postpendence y
40457:
40458: 07 XOR //Exclusive OR x^y+^x*y
40459:
40460: 08 OR //Disjunction OR = x+y
40461:
40462: 09 NOR //Rejection
40463:
40464: 10 AEQ //Aequivalence x<-->y, ^x^y+x*y
40465:
40466: 11 NEGY //YNegation ^y
40467:
40468: 12 IMPY //YImplication y-->x; x+^y
40469:
40470: 13 NEGX //Xnegation ^x
40471:
40472: 14 IMPX //XImplication x-->y; ^x+y
40473:
40474: 15 NAND //Exclusion
40475:
40476: 16 TRUE //TRUE Tautologic
40477: -----
40478:
40479: http://sourceforge.net/projects/maxbox/files/Examples/
40480: http://sourceforge.net/projects/maxbox/files/Examples/

```

```

40481:
40482: Help Online: http://www.softwareschule.ch/maxbox\_functions.txt
40483:
40484: WebScript Examples: Get Web Script
40485: http://www.softwareschule.ch/examples/performer.txt;
40486: http://www.softwareschule.ch/examples/turtle.txt;
40487: http://www.softwareschule.ch/examples/SQLExport.txt;
40488: http://www.softwareschule.ch/examples/Richter.txt;
40489: http://www.softwareschule.ch/examples/checker.txt;
40490: http://www.softwareschule.ch/examples/demoscript.txt;
40491: http://www.softwareschule.ch/examples/ibzresult.txt;
40492: http://www.softwareschule.ch/examples/performindex.txt
40493: http://www.softwareschule.ch/examples/processlist.txt
40494: http://www.softwareschule.ch/examples/game.txt
40495: http://www.softwareschule.ch/examples/GEOGPS.txt
40496: http://www.softwareschule.ch/examples/turtle2.txt
40497: http://www.softwareschule.ch/examples/turtle3.txt
40498: http://www.softwareschule.ch/examples/asyncterminal.txt
40499: http://www.softwareschule.ch/examples/snowflake.txt
40500: http://www.softwareschule.ch/examples/arduinooled.txt
40501: http://www.softwareschule.ch/examples/moon2.txt
40502: http://www.softwareschule.ch/examples/cockpit.txt
40503: http://www.softwareschule.ch/examples/tartaruga.txt
40504: http://www.softwareschule.ch/examples/surprise.txt
40505: http://www.softwareschule.ch/examples/weatherapp2.txt
40506: http://www.softwareschule.ch/examples/weatherapp3.txt
40507: http://www.softwareschule.ch/examples/bigint.txt
40508: http://www.softwareschule.ch/examples/weatherapp.txt
40509: http://www.softwareschule.ch/examples/weatherapp.txt
40510: http://www.softwareschule.ch/examples/crypto.txt
40511: http://www.softwareschule.ch/examples/stack.htm
40512: http://www.softwareschule.ch/examples/fisher.htm
40513: http://www.softwareschule.ch/examples/arduinotemp.htm
40514: http://www.softwareschule.ch/examples/web.txt
40515: http://www.softwareschule.ch/examples/web3.txt
40516: http://www.softwareschule.ch/examples/web4.txt
40517: http://www.softwareschule.ch/examples/cryptocomplete.txt
40518: http://www.softwareschule.ch/examples/web4arduino.txt
40519: http://www.softwareschule.ch/examples/adafruit.txt
40520: http://www.softwareschule.ch/examples/stack.txt
40521: http://www.softwareschule.ch/examples/stack.htm
40522: http://www.softwareschule.ch/examples/billiard.txt
40523: http://www.softwareschule.ch/examples/systeminfo\_tester.htm
40524: http://www.softwareschule.ch/examples/systeminfo.htm
40525: http://www.softwareschule.ch/examples/elevator.txt
40526: http://www.softwareschule.ch/examples/sha256.txt
40527: http://www.softwareschule.ch/examples/robo.txt
40528: http://www.softwareschule.ch/examples/robo2.txt
40529: http://www.softwareschule.ch/examples/font.txt
40530: http://www.softwareschule.ch/examples/spider.txt
40531: http://www.softwareschule.ch/examples/totient.txt
40532: http://www.softwareschule.ch/examples/811\_mXpcatest\_dmath\_datascience.txt
40533: http://www.softwareschule.ch/examples/datascience.txt
40534: http://www.softwareschule.ch/examples/pi.txt
40535: http://www.softwareschule.ch/examples/excelexport.txt
40536: http://www.softwareschule.ch/examples/tokenizer.htm
40537: http://www.softwareschule.ch/examples/snake.txt
40538: http://www.softwareschule.ch/examples/asteroid.txt
40539: http://www.softwareschule.ch/examples/weatherapp46.txt
40540: http://www.softwareschule.ch/examples/cryptocomplete.txt
40541: http://www.softwareschule.ch/examples/web4arduino.txt
40542: http://www.softwareschule.ch/examples/wmi.txt
40543: http://www.softwareschule.ch/examples/wmi2.txt
40544: http://www.softwareschule.ch/examples/reverse.htm
40545: http://www.softwareschule.ch/examples/weatherapp4.txt
40546: http://www.softwareschule.ch/examples/tictac.txt
40547: http://www.softwareschule.ch/examples/tictac3.txt
40548: http://www.softwareschule.ch/examples/mapbox2.txt
40549: http://www.softwareschule.ch/examples/neuralnetwork.txt
40550: http://www.softwareschule.ch/examples/datascience.txt
40551: http://www.softwareschule.ch/examples/billiard.txt
40552: http://www.softwareschule.ch/examples/sentiment2.txt
40553: http://www.softwareschule.ch/examples/python.htm
40554: http://www.softwareschule.ch/examples/asteroid.txt
40555: http://www.softwareschule.ch/examples/penny.txt
40556: http://www.softwareschule.ch/examples/snake3.txt
40557: http://www.softwareschule.ch/examples/unittest.txt
40558: http://www.softwareschule.ch/examples/410\_titanic\_keras\_predictor.txt
40559: http://www.softwareschule.ch/examples/410\_titanic\_keras\_predictor.htm
40560: http://www.softwareschule.ch/examples/histogram.txt
40561: http://www.softwareschule.ch/examples/odometer.txt
40562: http://www.softwareschule.ch/examples/chartapi.txt
40563: http://www.softwareschule.ch/examples/geoloc.txt
40564: http://www.softwareschule.ch/examples/xml.txt
40565: http://www.softwareschule.ch/examples/4gewinnt.txt
40566: http://www.softwareschule.ch/examples/cgi.txt
40567: http://www.softwareschule.ch/examples/singlesamplepredict.py
40568: http://www.softwareschule.ch/examples/perform.txt
40569: http://www.softwareschule.ch/examples/statistic.txt

```

```

40570: http://www.softwareschule.ch/examples/picbox.txt
40571: http://www.softwareschule.ch/examples/memory2.txt
40572: http://www.softwareschule.ch/examples/getgeomap.txt
40573: http://www.softwareschule.ch/examples/getzip.txt
40574: http://www.softwareschule.ch/examples/getrsa.txt
40575: http://www.softwareschule.ch/examples/getrsa5.txt
40576: http://www.softwareschule.ch/examples/getrsa5.htm
40577: http://www.softwareschule.ch/examples/scramble.txt
40578: http://www.softwareschule.ch/examples/factor2.txt
40579: http://www.softwareschule.ch/examples/bug.txt
40580: http://www.softwareschule.ch/examples/smalldic.txt
40581: http://www.softwareschule.ch/examples/metadefender.txt
40582: http://www.softwareschule.ch/examples/pythonoma.txt
40583: http://www.softwareschule.ch/examples/992_detector21_wget_integrate2.htm
40584: http://www.softwareschule.ch/examples/perceptron.txt
40585: http://www.softwareschule.ch/examples/perceptron2.txt
40586: http://www.softwareschule.ch/examples/perceptron2.htm
40587: http://www.softwareschule.ch/examples/sphere.txt
40588: http://www.softwareschule.ch/examples/entropy.txt
40589: http://www.softwareschule.ch/examples/protein.txt
40590: http://www.softwareschule.ch/examples/wordwheel.txt
40591: http://www.softwareschule.ch/examples/soundcloud.txt
40592: http://www.softwareschule.ch/examples/geometry.txt
40593: http://www.softwareschule.ch/examples/time.txt
40594: http://www.softwareschule.ch/examples/sphere2.txt
40595: http://www.softwareschule.ch/examples/sphere2.htm
40596: http://www.softwareschule.ch/examples/unittests.txt
40597: http://www.softwareschule.ch/examples/newssentiment.txt
40598: http://www.softwareschule.ch/examples/newssentiment.htm
40599: http://www.softwareschule.ch/examples/bbcnews.txt
40600: http://www.softwareschule.ch/examples/mathtool.txt
40601: http://www.softwareschule.ch/examples/newssentiment2.txt
40602: http://www.softwareschule.ch/examples/covid2.txt
40603: http://www.softwareschule.ch/examples/covid3.txt
40604: http://www.softwareschule.ch/examples/covidapp3.txt
40605: http://www.softwareschule.ch/examples/signtoolbatch.txt
40606: http://www.softwareschule.ch/examples/openweather.txt
40607: http://www.softwareschule.ch/examples/weatherbox.txt
40608: http://www.softwareschule.ch/examples/winet.txt
40609: http://www.softwareschule.ch/examples/digiclock.txt
40610: http://www.softwareschule.ch/examples/pydemo.txt
40611: http://www.softwareschule.ch/examples/pydemo2.txt
40612: http://www.softwareschule.ch/examples/pydemo3.txt
40613: http://www.softwareschule.ch/examples/pydemo13.txt
40614: http://www.softwareschule.ch/examples/pydemo31.txt
40615: http://www.softwareschule.ch/examples/catapult.txt
40616: http://www.softwareschule.ch/examples/pydemo33.txt
40617: http://www.softwareschule.ch/examples/syndat.png
40618: http://www.softwareschule.ch/examples/cryptosystem2.txt
40619: http://www.softwareschule.ch/examples/cryptosystem3.txt
40620: http://www.softwareschule.ch/examples/springdemo.txt
40621: http://www.softwareschule.ch/examples/coaster.txt
40622: http://www.softwareschule.ch/examples/pydemo38.txt
40623: http://www.softwareschule.ch/examples/pydemo40.txt
40624: http://www.softwareschule.ch/examples/sunpos.txt
40625: http://www.softwareschule.ch/examples/geocoding.txt
40626: http://www.softwareschule.ch/examples/sentiment3.txt
40627: http://www.softwareschule.ch/examples/sentiment4.txt
40628: http://www.softwareschule.ch/examples/qrcode.txt
40629: http://www.softwareschule.ch/examples/json3.txt
40630: http://www.softwareschule.ch/examples/json4.txt
40631: http://www.softwareschule.ch/examples/shellcomposer.txt
40632: http://www.softwareschule.ch/examples/4gewinnt2.txt
40633: http://www.softwareschule.ch/examples/space2.txt
40634: http://www.softwareschule.ch/examples/pydemo48.txt
40635: http://www.softwareschule.ch/examples/json7.txt
40636: http://www.softwareschule.ch/examples/cdplayer.txt
40637: http://www.softwareschule.ch/examples/json9.txt
40638: http://www.softwareschule.ch/examples/ocr.txt
40639: http://www.softwareschule.ch/examples/chatgpt32.txt
40640: http://www.softwareschule.ch/weatherapp477.txt
40641:
40642: https://maxbox4.wordpress.com">maxbox4.wordpress.com
40643: https://softwareschule.code.blog">softwareschule.code.blog
40644: https://my6.code.blog">my6.code.blog
40645: https://github.com/maxkleiner/maxbox4/blob/master/newstoday_sentiment.ipynb
40646: http://www.softwareschule.ch/download/exampleedition2016.zip
40647: https://github.com/maxkleiner/fundamentals5
40648: https://thenewstack.io/which-programming-languages-use-the-least-electricity/
40649: https://www.clevercomponents.com/articles/article052/
40650:
40651:
40652: Delphi Basics Run Time Library listing
40653: *****
40654: A
40655: Compiler Directive $A Determines whether data is aligned or packed
40656: Compiler Directive $Align Determines whether data is aligned or packed
40657: Compiler Directive $AppType Determines the application type : GUI or Console
40658: Proc SysUtils Abort Aborts the current processing with a silent exception

```

40659: Func System Abs Gives the **absolute** value of a number (-ve sign is removed)
40660: Directive **Abstract** Defines a **class** method only implemented in subclasses
40661: Variable System AbstractErrorProc Defines a proc called when an **abstract** method is called
40662: Func System Addr Gives the address of a variable, **Func** or **procedure**
40663: Keyword **And** Boolean **and** or bitwise **and** of two arguments
40664: **Type** System AnsiChar A character **type** guaranteed to be 8 bits in size
40665: Func SysUtils AnsiCompareStr Compare two strings for equality
40666: Func SysUtils AnsiCompareText Compare two strings for equality, ignoring case
40667: Func StrUtils AnsiContainsStr Returns true if a **string** contains a substring
40668: Func StrUtils AnsiEndsStr Returns true if a **string** ends with a substring
40669: Func StrUtils AnsiIndexStr Compares a **string** with a list of strings - returns match index
40670: Func StrUtils AnsiLeftStr Extracts characters from the left of a **string**
40671: Func SysUtils AnsiLowerCase Change upper case characters in a **string** to lower case
40672: Func StrUtils AnsiMatchStr Returns true if a **string** exactly matches one of a list of strings
40673: Func StrUtils AnsiMidStr Returns a substring from the middle characters of a **string**
40674: Func StrUtils AnsiPos Find the position of one **string** in another
40675: Func StrUtils AnsiReplaceStr Replaces a part of one **string** with another
40676: Func StrUtils AnsiReverseString Reverses the sequence of letters in a **string**
40677: Func StrUtils AnsiRightStr Extracts characters from the right of a **string**
40678: Func StrUtils AnsiStartsStr Returns true if a **string** starts with a substring
40679: **Type** System Ansistr A data **type** that holds a **string** of AnsiChars
40680: Func SysUtils AnsiUpperCase Change lower case characters in a **string** to upper case
40681: Proc System Append Open a text **file** to allow appending of text to the end
40682: Proc SysUtils AppendStr Concatenate one **string** onto the end of another
40683: Func Math ArcCos The Arc Cosine of a number, returned in radians
40684: Func Math ArcSin The Arc Sine of a number, returned in radians
40685: Func System ArcTan The Arc Tangent of a number, returned in radians
40686: Keyword **Array** A data **type** holding indexable collections of data
40687: Keyword **As** Used for casting **object** references
40688: Proc System Assign Assigns a **file** handle to a binary or text **file**
40689: Func System Assigned Returns true if a reference is not nil
40690: Proc System AssignFile Assigns a **file** handle to a binary or text **file**
40691: Proc Printers AssignPrn Treats the printer as a text **file** - an easy way of printing text
40692:
40693: B
40694: Compiler Directive **\$B** Whether to short cut and and or operations
40695: Compiler Directive **\$BoolEval** Whether to short cut and and or operations
40696: Proc SysUtils Beep Make a beep sound
40697: Keyword **Begin** Keyword that starts a statement block
40698: Func System BeginThread Begins a separate thread of code execution
40699: Proc System BlockRead Reads a block of data records from an untyped binary **file**
40700: Proc System BlockWrite Writes a block of data records to an untyped binary **file**
40701: **Type** System Boolean Allows just True and False values
40702: Func Classes Bounds Create a TRect value from top left and size values
40703: Proc System Break Forces a jump out of a single loop
40704: **Type** System Byte An Int **type** supporting values 0 to 255
40705:
40706: C
40707: **Type** System Cardinal The basic unsigned Int **type**
40708: Keyword **Case** A mechanism for acting upon different values of an Ordinal
40709: Func StdConvrs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
40710: Func SysUtils ChangeFileExt Change the extension part of a **file** name
40711: **Type** System Char Variable **type** holding a single character
40712: Proc System ChDir Change the working drive plus path for a specified drive
40713: Func System Chr Convert an Int into a character
40714: Keyword **Class** Starts the declaration of a **type** of object class
40715: Proc System Close Closes an open **file**
40716: Proc System CloseFile Closes an open **file**
40717: Variable System CmdLine Holds the execution text used to start the current program
40718: **Type** System Comp A 64 bit signed Int
40719: Func SysUtils CompareStr Compare two strings to see which is greater than the other
40720: Func SysUtils CompareText Compare two strings for equality, ignoring case
40721: Func Math CompareValue Compare numeric values with a tolerance
40722: Func System Concat Concatenates one or more strings into one **string**
40723: Keyword **Const** Starts the definition of fixed data values
40724: Keyword **Constructor** Defines the method used to create an **object** from a **class**
40725: Proc System Continue Forces a jump to the next iteration of a loop
40726: Func ConvUtils Convert Convert one measurement value to another
40727: Func System Copy Create a copy of part of a **string** or an **array**
40728: Func System Cos The Cosine of a number
40729: Func SysUtils CreateDir Create a directory
40730: **Type** System Currency A floating point **type** with 4 decimals used for financial values
40731: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
40732: Variable SysUtils CurrencyFormat Defines currency **string** placement in curr display functions
40733: Variable SysUtils CurrencyString The currency **string** used in currency display functions
40734: Func SysUtils CurrToStr Convert a currency value to a **string**
40735: Func SysUtils CurrToStrF Convert a currency value to a **string** with formatting
40736:
40737: D
40738: Compiler Directive **\$D** Determines whether application debug information is built
40739: Compiler Directive **\$DebugInfo** Determines whether application debug information is built
40740: Compiler Directive **\$Define** Defines a compiler directive symbol - as used by IfDef
40741: Compiler Directive **\$DefinitionInfo** Determines whether application symbol information is built
40742: Func SysUtils Date Gives the current date
40743: Variable SysUtils DateSeparator The character used to separate display date fields
40744: Func SysUtils DateTimeToFileDate Convert a TDateTime value to a **File** date/time format
40745: Func SysUtils DateTimeToStr Converts TDateTime date and time values to a **string**
40746: Proc SysUtils DateTimeToString Rich formatting of a TDateTime variable into a **string**
40747: Func SysUtils DateToStr Converts a TDateTime date value to a **string**


```

40748: Func DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
40749: Func DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
40750: Func DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
40751: Func SysUtils DayOfWeek Gives day of week index for a TDateTime value
40752: Func DateUtils DaysBetween Gives the whole number of days between 2 dates
40753: Func DateUtils DaysInAMonth Gives the number of days in a month
40754: Func DateUtils DaysInAYear Gives the number of days in a year
40755: Func DateUtils DaySpan Gives the fractional number of days between 2 dates
40756: Proc System Dec Decrement an ordinal variable
40757: Variable SysUtils DecimalSeparator The character used to display the decimal point
40758: Proc SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
40759: Proc DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
40760: Proc SysUtils DecodeTime Break a TDateTime value into individual time values
40761: Directive Default Defines default processing for a property
40762: Func Math DegToRad Convert a degrees value to radians
40763: Proc System Delete Delete a section of characters from a string
40764: Func SysUtils DeleteFile Delete a file specified by its file name
40765: Keyword Destructor Defines the method used to destroy an object
40766: Func SysUtils DirectoryExists Returns true if the given directory exists
40767: Func SysUtils DiskFree Gives the number of free bytes on a specified drive
40768: Func SysUtils DiskSize Gives the size in bytes of a specified drive
40769: Proc System Dispose Dispose of storage used by a pointer type variable
40770: Keyword Div Performs Int division, discarding the remainder
40771: Keyword Do Defines the start of some controlled action
40772: Type System Double A floating point type supporting about 15 digits of precision
40773: Keyword DownTo Prefixes an decremental for loop target value
40774: Func StrUtils DupeString Creates a string containing copies of a substring
40775: Directive Dynamic Allows a class method to be overridden in derived classes
40776:
40777: E
40778: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
40779: Compiler Directive $EndIf Terminates conditional code compilation
40780: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
40781: Keyword Else Starts false section of if, case and try statements
40782: Func SysUtils EncodeDate Build a TDateTime value from year, month and day values
40783: Func DateUtils EncodeDateTime Build a TDateTime value from day and time values
40784: Func SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
40785: Keyword End Keyword that terminates statement blocks
40786: Func DateUtils EndOfDay Generate a TDateTime value set to the very end of a day
40787: Func DateUtils EndOfMonth Generate a TDateTime value set to the very end of a month
40788: Proc System EndThread Terminates a thread with an exit code
40789: Proc System Eof Returns true if a file opened with Reset is at the end
40790: Func System Eoln Returns true if the current text file is pointing at a line end
40791: Proc System Erase Erase a file
40792: Variable System ErrorAddr Sets the error address when an application terminates
40793: Keyword Except Starts the error trapping clause of a Try statement
40794: Proc System Exclude Exclude a value in a set variable
40795: Proc System Exit Exit abruptly from a Func or procedure
40796: Variable System ExitCode Sets the return code when an application terminates
40797: Func System Exp Gives the exponent of a number
40798: Directive System Export Makes a Func or Proc in a DLL externally available
40799: Type System Extended The floating point type with the highest capacity and precision
40800: Func SysUtils ExtractFileDir Extracts the dir part of a full file name
40801: Func SysUtils ExtractFileDrive Extracts the drive part of a full file name
40802: Func SysUtils ExtractFileExt Extracts the extension part of a full file name
40803: Func SysUtils ExtractFileName Extracts the name part of a full file name
40804: Func SysUtils ExtractFilePath Extracts the path part of a full file name
40805:
40806: F
40807: Func StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
40808: Keyword File Defines a typed or untyped file
40809: Func SysUtils FileAge Get the last modified date/time of a file without opening it
40810: Func SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
40811: Func SysUtils FileExists Returns true if the given file exists
40812: Func SysUtils FileGetAttr Gets the attributes of a file
40813: Variable System FileMode Defines how Reset opens a binary file
40814: Func System FilePos Gives the file position in a binary or text file
40815: Func SysUtils FileSearch Search for a file in one or more directories
40816: Func SysUtils FileSetAttr Sets the attributes of a file
40817: Func SysUtils FileSetDate Set the last modified date and time of a file
40818: Func System FileSize Gives the size in records of an open file
40819: Proc System FillChar Fills out a section of storage with a fill character or byte value
40820: Keyword Finally Starts the unconditional code section of a Try statement
40821: Func SysUtils FindClose Closes a successful FindFirst file search
40822: Func SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
40823: Func SysUtils FindFirst Finds all files matching a file mask and attributes
40824: Func SysUtils FindNext Find the next file after a successful FindFirst
40825: Func SysUtils FloatToStr Convert a floating point value to a string
40826: Func SysUtils FloatToStrF Convert a floating point value to a string with formatting
40827: Proc System Flush Flushes buffered text file data to the file
40828: Keyword For Starts a loop that executes a finite number of times
40829: Func SysUtils ForceDirectories Create a new path of directories
40830: Func SysUtils Format Rich formatting of numbers and text into a string
40831: Func SysUtils FormatCurr Rich formatting of a currency value into a string
40832: Func SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
40833: Func SysUtils FormatFloat Rich formatting of a floating point number into a string
40834: Func System Frac The fractional part of a floating point number
40835: Proc SysUtils FreeAndNil Free memory for an object and set it to nil
40836: Proc System FreeMem Free memory storage used by a variable

```

40837: Keyword System Func Defines a subroutine that returns a value
 40838:
 40839: G
 40840: Func SysUtils GetCurrentDir Get the current directory (drive plus directory)
 40841: Proc System GetDir Get the **default** directory (drive plus path) **for** a specified drive
 40842: Func System GetLastError Gives the error code **of** the last failing Windows API call
 40843: Proc SysUtils GetLocaleFormatSettings Gets locale values **for** thread-safe functions
 40844: Func System GetMem Get a specified number **of** storage bytes
 40845: Keyword Goto Forces a jump **to** a **label**, regardless **of** nesting
 40846:
 40847: H
 40848: Compiler Directive \$H Treat **string** types **as** Ansistr **or** ShortString
 40849: Compiler Directive \$Hints Determines whether Delphi shows compilation hints
 40850: Proc System Halt Terminates the **program with** an optional dialog
 40851: Func System Hi Returns the hi-order byte **of** a (2 byte) Int
 40852: Func System High Returns the highest value **of** a **type or** variable
 40853:
 40854: I
 40855: Compiler Directive \$I Allows code **in** an include **file to** be incorporated into a **Unit**
 40856: Compiler Directive \$IfDef Executes code **if** a conditional symbol has been defined
 40857: Compiler Directive \$IfNDef Executes code **if** a conditional symbol has **not** been defined
 40858: Compiler Directive \$IfOpt Tests **for** the state **of** a Compiler directive
 40859: Compiler Directive \$Include Allows code **in** an include **file to** be incorporated into a **Unit**
 40860: Compiler Directive \$IOChecks When **on**, an IO operation error throws an exception
 40861: Keyword **If** Starts a conditional expression **to** determine what **to do** next
 40862: Keyword **Implementation** Starts the **implementation** (code) section **of** a **Unit**
 40863: Keyword **In** Used **to test if** a value **is** a member **of** a **set**
 40864: Proc System Inc Increment an ordinal variable
 40865: Func DateUtils IncDay Increments a TDateTime variable by **+** **or** **-** number **of** days
 40866: Proc System Include Include a value **in** a **set** variable
 40867: Func DateUtils IncMillisecond Increments a TDateTime variable by **+** **or** **-** number **of** milliseconds
 40868: Func DateUtils IncMinute Increments a TDateTime variable by **+** **or** **-** number **of** minutes
 40869: Func SysUtils IncMonth Increments a TDateTime variable by a number **of** months
 40870: Func DateUtils IncSecond Increments a TDateTime variable by **+** **or** **-** number **of** seconds
 40871: Func DateUtils IncYear Increments a TDateTime variable by a number **of** years
 40872: Directive Index Principally defines indexed **class** data properties
 40873: Constant Math Infinity Floating point value **of** infinite size
 40874: Keyword **Inherited** Used **to call** the parent **class constructor or destructor** method
 40875: Variable System Input Defines the standard input text **file**
 40876: Func Dialogs InputBox Display a dialog that asks **for** user text input, **with default**
 40877: Func Dialogs InputQuery Display a dialog that asks **for** user text input
 40878: Proc System Insert Insert a **string** into another **string**
 40879: Func System Int The Int part **of** a floating point number **as** a float
 40880: **Type** System Int64 A 64 bit sized Int - the largest **in** Delphi
 40881: **Type** System Int The basic Int **type**
 40882: Keyword System **Interface** Used **for Unit** external definitions, **and as** a **Class** skeleton
 40883: Func SysUtils IntToHex Convert an Int into a hexadecimal **string**
 40884: Func SysUtils IntToStr Convert an Int into a **string**
 40885: Func System IOResult Holds the return code **of** the last I/O operation
 40886: Keyword **Is** Tests whether an **object is** a certain **class or** ascendant
 40887: Func Math IsInfinite Checks whether a floating point number **is** infinite
 40888: Func SysUtils IsLeapYear Returns true **if** a given calendar year **is** a leap year
 40889: Func System IsMultiThread Returns true **if** the code **is** running multiple threads
 40890: Func Math IsNaN Checks **to see if** a floating point number holds a real number
 40891:
 40892: L
 40893: Compiler Directive \$L Determines what application debug information **is** built
 40894: Compiler Directive \$LocalSymbols Determines what application debug information **is** built
 40895: Compiler Directive \$LongStrings Treat **string** types **as** Ansistr **or** ShortString
 40896: Func SysUtils LastDelimiter Find the last position **of** selected characters **in** a **string**
 40897: Func System Length Return the number **of** elements **in** an **array or** **string**
 40898: Func System Ln Gives the natural logarithm **of** a number
 40899: Func System Lo Returns the low-order byte **of** a (2 byte) Int
 40900: Func Math Log10 Gives the log **to** base 10 **of** a number
 40901: Variable SysUtils LongDateFormat Long version **of** the date **to** **string** format
 40902: Variable SysUtils LongDayNames An **array of** days **of** the week names, starting 1 = Sunday
 40903: **Type** System LongInt An Int whose size **is** guaranteed **to** be 32 bits
 40904: Variable SysUtils LongMonthNames An **array of** days **of** the month names, starting 1 = January
 40905: Variable SysUtils LongTimeFormat Long version **of** the time **to** **string** format
 40906: **Type** System LongWord A 32 bit unsigned Int
 40907: Func System Low Returns the lowest value **of** a **type or** variable
 40908: Func SysUtils LowerCase Change upper **case** characters **in** a **string to** lower **case**
 40909:
 40910: M
 40911: Compiler Directive \$MinEnumSize Sets the minimum storage used **to** hold enumerated types
 40912: Func Math Max Gives the maximum **of** two Int values
 40913: Constant System MaxInt The maximum value an Int can have
 40914: Constant System MaxLongInt The maximum value an LongInt can have
 40915: Func Math Mean Gives the average **for** a **set of** numbers
 40916: Func Dialogs MessageDlg Displays a **message**, symbol, **and** selectable buttons
 40917: Func Dialogs MessageDlgPos Displays a **message** plus buttons at a given screen position
 40918: Func Math Min Gives the minimum **of** two Int values
 40919: Constant SysUtils MinsPerDay Gives the number **of** minutes **in** a day
 40920: Proc System Mkdir Make a directory
 40921: Keyword **Mod** Performs Int division, returning the remainder
 40922: Constant SysUtils MonthDays Gives the number **of** days **in** a month
 40923: Func DateUtils MonthOfTheYear Gives the month **of** the year **for** a TDateTime value
 40924: Proc System Move Copy bytes **of** data from a source **to** a destination
 40925:

40926: N
 40927: Constant Math NaN **Not** a real number
 40928: Variable SysUtils NegCurrFormat Defines negative amount formatting **in** currency displays
 40929: Proc System New Create a new pointer **type** variable
 40930: Constant System **Nil** A pointer value that **is** defined **as** undetermined
 40931: Keyword **Not** Boolean **Not** or bitwise **not** of one arguments
 40932: Func SysUtils Now Gives the current date **and** time
 40933: Variable Variants Null A variable that has no value
 40934:
 40935: O
 40936: Compiler Directive **\$O** Determines whether Delphi optimises code when compiling
 40937: Compiler Directive **\$Optimization** Determines whether Delphi optimises code when compiling
 40938: Compiler Directive **\$OverflowChecks** Determines whether Delphi checks **Int** **and** enum bounds
 40939: Keyword System **Object** Allows a subroutine data **type** to refer to an **object** method
 40940: Func System Odd Tests whether an **Int** has an odd value
 40941: Keyword **Of** Linking keyword used **in** many places
 40942: Keyword **On** Defines exception handling **in** a **Try Except** clause
 40943: Keyword **Or** Boolean **or** or bitwise **or** of two arguments
 40944: Func System Ord Provides the Ordinal value **of** an **Int**, character **or** enum
 40945: Directive **Out** Identifies a routine parameter **for** output only
 40946: Variable System Output Defines the standard output text **file**
 40947: Directive **Overload** Allows **2** **or** more routines to have the same name
 40948: Directive **Override** Defines a method that replaces a **virtual** parent **class** method
 40949:
 40950: P
 40951: Keyword **Packed** Compacts complex data types into minimal storage
 40952: **Type** System PAnsiChar A pointer to an AnsiChar value
 40953: **Type** System PAnsiStr Pointer to an AnsiStr value
 40954: Func System ParamCount Gives the number **of** parameters passed to the current **program**
 40955: Func System ParamStr Returns one **of** the parameters used to run the current **program**
 40956: **Type** System PChar A pointer to a Char value
 40957: **Type** System PCurrency Pointer to a Currency value
 40958: **Type** System PDateTime Pointer to a TDateTime value
 40959: **Type** System PExtended Pointer to a Extended floating point value
 40960: Func System Pi The mathematical constant
 40961: **Type** System PInt64 Pointer to an Int64 value
 40962: Func Classes Point Generates a TPoint value from X **and** Y values
 40963: **Type** System Pointer Defines a general use Pointer to any memory based data
 40964: Func Classes PointsEqual Compares two TPoint values **for** equality
 40965: Func System Pos Find the position **of** one **string** **in** another
 40966: Func System Pred Decrement an ordinal variable
 40967: Func Printers Printer Returns a reference to the global Printer **object**
 40968: Directive **Private** Starts the section **of** **private** data **and** methods **in** a **class**
 40969: Keyword System Proc Defines a subroutine that does **not** return a value
 40970: Proc FileCtrl ProcessPath Split a drive/path/filename **string** into its constituent parts
 40971: Keyword System **Program** Defines the start **of** an application
 40972: Func Dialogs PromptForFileName Shows a dialog allowing the user to select a **file**
 40973: Keyword System **Property** Defines controlled access to **class** fields
 40974: Directive **Protected** Starts a section **of** **class** **private** data accesible to sub-classes
 40975: **Type** System PShortString A pointer to an ShortString value
 40976: **Type** System PString Pointer to a String value
 40977: Func Types PtInRect Tests to see **if** a point lies within a rectangle
 40978: Directive **Public** Starts an externally accessible section **of** a **class**
 40979: Directive **Published** Starts a **published** externally accessible section **of** a **class**
 40980: **Type** System PVariant Pointer to a Variant value
 40981: **Type** System PWideChar Pointer to a WideChar
 40982: **Type** System PWideString Pointer to a WideString value
 40983:
 40984: Q
 40985: Compiler Directive **\$Q** Determines whether Delphi checks **Int** **and** enum bounds
 40986:
 40987: R
 40988: Compiler Directive **\$R** Determines whether Delphi checks **array** bounds
 40989: Compiler Directive **\$RangeChecks** Determines whether Delphi checks **array** bounds
 40990: Compiler Directive **\$ReferenceInfo** Determines whether symbol reference information **is** built
 40991: Compiler Directive **\$Resource** Defines a resource **file** to be included **in** the application linking
 40992: Func Math RadToDeg Converts a radian value to degrees
 40993: Keyword **Raise** **Raise** an exception
 40994: Func System Random Generate a random floating point **or** **Int** number
 40995: Proc System Randomize Reposition the Random number generator next value
 40996: Func Math RandomRange Generate a random **Int** number within a supplied range
 40997: Variable System RandSeed Reposition the Random number generator next value
 40998: Proc System **Read** **Read** data from a binary **or** text **file**
 40999: Proc System ReadLn **Read** a complete line **of** data from a text **file**
 41000: **Type** System Real A floating point **type** supporting about **15** digits **of** precision
 41001: **Type** System Real48 The floating point **type** with the highest capacity **and** precision
 41002: Proc System ReallocMem Reallocate an existing block **of** storage
 41003: Func DateUtils RecodeDate Change only the date part **of** a TDateTime variable
 41004: Func DateUtils RecodeTime Change only the time part **of** a TDateTime variable
 41005: Keyword **Record** A structured data **type** - holding fields **of** data
 41006: Func Classes Rect Create a TRect value from **2** points **or** **4** coordinates
 41007: Func SysUtils Removedir Remove a directory
 41008: Proc System Rename Rename a **file**
 41009: Func SysUtils RenameFile Rename a **file** **or** directory
 41010: Keyword **Repeat** **Repeat** statements **until** a termination condition **is** met
 41011: Proc SysUtils ReplaceDate Change only the date part **of** a TDateTime variable
 41012: Proc SysUtils ReplaceTime Change only the time part **of** a TDateTime variable
 41013: Proc System Reset Open a text **file** **for** reading, **or** binary **file** **for** read/write
 41014: Variable System Result A variable used to hold the return value from a **function**

```

41015: Proc System ReWrite Open a text or binary file for write access
41016: Proc System Rmdir Remove a directory
41017: Func System Round Rounds a floating point number to an Int
41018: Proc System RunError Terminates the program with an error dialog
41019:
41020: S
41021: Constant SysUtils SecsPerDay Gives the number of seconds in a day
41022: Proc System Seek Move the pointer in a binary file to a new record position
41023: Func System SeekEof Skip to the end of the current line or file
41024: Func System SeekEoln Skip to the end of the current line or file
41025: Func FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
41026: Variable System Self Hidden parameter to a method - refers to the containing object
41027: Keyword Set Defines a set of up to 255 distinct values
41028: Func SysUtils SetCurrentDir Change the current directory
41029: Proc System SetLength Changes the size of a string, or the size(s) of an array
41030: Proc System SetString Copies characters from a buffer into a string
41031: Keyword Shl Shift an Int value left by a number of bits
41032: Variable SysUtils ShortDateFormat Compact version of the date to string format
41033: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
41034: Type System ShortInt An Int type supporting values -128 to 127
41035: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
41036: Type System ShortString Defines a string of up to 255 characters
41037: Variable SysUtils ShortTimeFormat Short version of the time to string format
41038: Proc Dialogs ShowMessage Display a string in a simple dialog with an OK button
41039: Proc Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
41040: Proc Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
41041: Keyword Shr Shift an Int value right by a number of bits
41042: Func System Sin The Sine of a number
41043: Type System Single The smallest capacity and precision floating point type
41044: Func System SizeOf Gives the storage byte size of a type or variable
41045: Func System Slice Creates a slice of an array as an Open Array parameter
41046: Type System SmallInt An Int type supporting values from -32768 to 32767
41047: Func System Sqr Gives the square of a number
41048: Func System Sqrt Gives the square root of a number
41049: Proc System Str Converts an Int or floating point number to a string
41050: Type System String A data type that holds a string of characters
41051: Func System StringOfChar Creates a string with one character repeated many times
41052: Func SysUtils StringReplace Replace one or more substrings found within a string
41053: Func System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
41054: Func SysUtils StrScan Searches for a specific character in a constant string
41055: Func SysUtils StrToCurr Convert a number string into a currency value
41056: Func SysUtils StrToDate Converts a date string into a TDateTime value
41057: Func SysUtils StrToDateTime Converts a date+time string into a TDateTime value
41058: Func SysUtils StrToFloat Convert a number string into a floating point value
41059: Func SysUtils StrToInt Convert an Int string into an Int value
41060: Func SysUtils StrToInt64 Convert an Int string into an Int64 value
41061: Func SysUtils StrToInt64Def Convert a string into an Int64 value with default
41062: Func SysUtils StrToIntDef Convert a string into an Int value with default
41063: Func SysUtils StrToTime Converts a time string into a TDateTime value
41064: Func StrUtils StuffString Replaces a part of one string with another
41065: Func System Succ Increment an ordinal variable
41066: Func Math Sum Return the sum of an array of floating point values
41067:
41068: T
41069: Func Math Tan The Tangent of a number
41070: Type Classes TBits An object that can hold an infinite number of Boolean values
41071: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
41072: Type ConvUtils TConvType Defines a measurement type as used by Convert
41073: Type System TDateTime Data type holding a date and time value
41074: Type System Text Defines a file as a text file
41075: Type System TextFile Declares a file type for storing lines of text
41076: Type SysUtils TFloatFormat Formats for use in floating point number display functions
41077: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
41078: Keyword Then Part of an if statement - starts the true clause
41079: Variable SysUtils ThousandSeparator The character used to display the thousands separator
41080: Keyword ThreadVar Defines variables that are given separate instances per thread
41081: Func SysUtils Time Gives the current time
41082: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
41083: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
41084: Variable SysUtils TimeSeparator The character used to separate display time fields
41085: Func SysUtils TimeToStr Converts a TDateTime time value to a string
41086: Type Classes TList General purpose container of a list of objects
41087: Keyword To Prefixes an incremental for loop target value
41088: Type System TObject The base class type that is ancestor to all other classes
41089: Func DateUtils Tomorrow Gives the date tomorrow
41090: Type Dialogs TOpenDialog Displays a file selection dialog
41091: Type Types TPoint Holds X and Y Int values
41092: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
41093: Type Types TRect Holds rectangle coordinate values
41094: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
41095: Func SysUtils Trim Removes leading and trailing blanks from a string
41096: Func SysUtils TrimLeft Removes leading blanks from a string
41097: Func SysUtils TrimRight Removes trailing blanks from a string
41098: Func System Trunc The Int part of a floating point number
41099: Proc System Truncate Truncates a file size - removes all data after the current position
41100: Keyword Try Starts code that has error trapping
41101: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
41102: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
41103: Type Classes TStringList Holds a variable length list of strings

```

```

41104: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
41105: Type System TThreadFunc Defines the Func to be called by BeginThread
41106: Var SysUtils TwoDigitYearCenturyWindow Sets century threshold for 2 digit year string conversions
41107: Keyword Type Defines a new category of variable or process
41108:
41109: U
41110: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
41111: Keyword Unit Defines the start of a unit file - a Delphi module
41112: Keyword Until Ends a Repeat control loop
41113: Func System UpCase Convert a Char value to upper case
41114: Func SysUtils UpperCase Change lower case characters in a string to upper case
41115: Keyword Uses Declares a list of Units to be imported
41116:
41117: V
41118: Proc System Val Converts number strings to Int and floating point values
41119: Keyword Var Starts the definition of a section of data variables
41120: Type System Variant A variable type that can hold changing data types
41121: Func Variants VarType Gives the current type of a Variant variable
41122: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
41123: Directive Virtual Allows a class method to be overridden in derived classes
41124:
41125: W
41126: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
41127: Keyword While Repeat statements whilst a continuation condition is met
41128: Type System WideChar Variable type holding a single International character
41129: Func System WideCharToString Copies a null terminated WideChar string to a normal string
41130: Type System WideString A data type that holds a string of WideChars
41131: Keyword With A means of simplifying references to structured variables
41132: Type System Word An Int type supporting values 0 to 65535
41133: Func SysUtils WrapText Add line feeds into a string to simulate word wrap
41134: Proc System Write Write data to a binary or text file
41135: Proc System WriteLn Write a complete line of data to a text file
41136:
41137: X
41138: Compiler Directive $X Controls some Pascal extension handling
41139: Keyword Xor Boolean Xor or bitwise Xor of two arguments
41140: Y
41141: Compiler Directive $Y Determines whether application symbol information is built
41142: Func DateUtils Yesterday Gives the date yesterday
41143: Z
41144: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
41145:
41146:
41147: Proc SIRegister_uPSUtils(CL: TPSPascalCompiler);
41148: begin      //'TPSBaseType','').SetString( Byte);
41149:   PSMainProcName','String').SetString( '!MAIN;
41150:   PSMainProcNameOrg','String').SetString( 'Main Proc;
41151:   'PSLowBuildSupport','LongInt').SetInt(12);
41152:   'PSCurrentBuildNo','LongInt').SetInt(23);
41153:   'PSCurrentVersion','String').SetString('1.31;
41154:   'PSValidHeader','LongInt').SetInt(1397769801);
41155:   'PSAddrStackStart','LongInt').SetInt(1610612736);
41156:   'PSAddrNegativeStackStart','LongInt').SetInt(1073741824);
41157:   'btReturnAddress','LongInt').SetInt(0);
41158:   'btU8','LongInt').SetInt(1);
41159:   'btS8','LongInt').SetInt(2);
41160:   'btU16','LongInt').SetInt(3);
41161:   'btS16','LongInt').SetInt(4);
41162:   'btU32','LongInt').SetInt(5);
41163:   'btS32','LongInt').SetInt(6);
41164:   'btSingle','LongInt').SetInt(7);
41165:   'btDouble','LongInt').SetInt(8);
41166:   'btExtended','LongInt').SetInt(9);
41167:   'btString','LongInt').SetInt(10);
41168:   'btRecord','LongInt').SetInt(11);
41169:   'btArray','LongInt').SetInt(12);
41170:   'btPointer','LongInt').SetInt(13);
41171:   'btPChar','LongInt').SetInt(14);
41172:   'btResourcePointer','LongInt').SetInt(15);
41173:   'btVariant','LongInt').SetInt(16);
41174:   'btS64','LongInt').SetInt(17);
41175:   'btU64','LongInt').SetInt(30);
41176:   'btChar','LongInt').SetInt(18);
41177:   'btWideString','LongInt').SetInt( 19);
41178:   'btWideChar','LongInt').SetInt( 20);
41179:   'btProcPtr','LongInt').SetInt( 21);
41180:   'btStaticArray','LongInt').SetInt( 22);
41181:   'btSet','LongInt').SetInt( 23);
41182:   'btCurrency','LongInt').SetInt( 24);
41183:   'btClass','LongInt').SetInt( 25);
41184:   'btInterface','LongInt').SetInt( 26);
41185:   'btNotificationVariant','LongInt').SetInt( 27);
41186:   'btUnicodeString','LongInt').SetInt( 28);
41187:   'btType','LongInt').SetInt( 130);
41188:   'btEnum','LongInt').SetInt( 129);
41189:   'btExtClass','LongInt').SetInt( 131);
41190:   'CM_A','LongInt').SetInt( 0);
41191:   'CM_CA','LongInt').SetInt( 1);
41192:   'CM_P','LongInt').SetInt( 2);

```



```

41193: 'CM_PV','LongInt').SetInt( 3);
41194: 'CM_PO','LongInt').SetInt( 4);
41195: 'Cm_C','LongInt').SetInt( 5);
41196: 'Cm_G','LongInt').SetInt( 6);
41197: 'Cm_CG','LongInt').SetInt( 7);
41198: 'Cm_CNG','LongInt').SetInt( 8);
41199: 'Cm_R','LongInt').SetInt( 9);
41200: 'Cm_ST','LongInt').SetInt( 10);
41201: 'Cm_Pt','LongInt').SetInt( 11);
41202: 'CM_CO','LongInt').SetInt( 12);
41203: 'Cm_cv','LongInt').SetInt( 13);
41204: 'cm_sp','LongInt').SetInt( 14);
41205: 'cm_bn','LongInt').SetInt( 15);
41206: 'cm_vm','LongInt').SetInt( 16);
41207: 'cm_sf','LongInt').SetInt( 17);
41208: 'cm_fg','LongInt').SetInt( 18);
41209: 'cm_puehx','LongInt').SetInt( 19);
41210: 'cm_poexh','LongInt').SetInt( 20);
41211: 'cm_in','LongInt').SetInt( 21);
41212: 'cm_spc','LongInt').SetInt( 22);
41213: 'cm_inc','LongInt').SetInt( 23);
41214: 'cm_dec','LongInt').SetInt( 24);
41215: 'cm_nop','LongInt').SetInt( 255);
41216: 'Cm_PG','LongInt').SetInt( 25);
41217: 'Cm_P2G','LongInt').SetInt( 26);
41218: TypeS('TbtU8','Byte;
41219: TypeS('TbtS8','ShortInt;
41220: TypeS('TbtU16','Word;
41221: TypeS('TbtS16','SmallInt;
41222: TypeS('TbtU32','Cardinal;
41223: TypeS('TbtS32','Longint;
41224: TypeS('TbtSingle','Single;
41225: TypeS('TbtDouble','double;
41226: TypeS('TbtExtended','Extended;
41227: TypeS('tbtCurrency','Currency;
41228: TypeS('tbts64','int64;
41229: TypeS('Tbtu64','uint64;
41230: TypeS('TbtString','string;
41231: Func MakeHash( const s : TbtString) : Longint;
41232: // TbtString = {$IFDEF DELPHI2009UP}Ansistr{$ELSE}String{$ENDIF};
41233: // 'PointerSize','LongInt').SetInt( IPointer ( 8 4 ));
41234: end;
41235:
41236: -----
41237: mapX:
41238: if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
41239:   writeln('cologne map found;
41240:   GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
41241:   writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
41242:   OpenMapX('church trier;
41243:   GetGeoCode(C_form,apath:Str; const data:Str; sfile:Bool):Str;
41244:   writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));}
41245:   >>> //latitude: '50.94133705' longitude: '6.95812076100766'
41246:   // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
41247:   writeln(GetGeoCoord('xml','church cefalu sicily',true))
41248:   CoordinateStr(Idx: Int; PosInSec: Double; PosIn: TPos):Str;
41249:   Func SendInput( cInputs : UINT; var pInputs : TInput; cbSize : Int) : UINT;
41250:   Func GetLastInputInfo( var plii : TLastInputInfo) : BOOL;
41251:   Proc JvErrorIntercept;
41252:   writeln(GetGeoInfo4('178.196.192.131', UrlGeoLookupInfo3));
41253:
41254:   {$IFDEF MSWINDOWS}
41255:   Proc Process;
41256:   var
41257:     Msg: TMsg;
41258:   begin
41259:     if ApplicationHasPriority then begin
41260:       Application.ProcessMessages;
41261:     end else begin
41262:       // This guarantees it will not ever call Application.Idle
41263:       if PeekMessage(Msg, 0, 0, 0, PM_NOREMOVE) then begin
41264:         Application.HandleMessage;
41265:       end;
41266:     end;
41267:   end;
41268:
41269:
41270: Ref:
41271: 1. writeln(SHA1(Exepath+'\\maxbox4.exe'))
41272: 2. shdig: TSHA1Digest;
41273:   shdig:= GetSHA1OfFile(false,exepath+'\\maxbox4.exe;
41274:   for i:= 0 to 19 do write(ByteToHex(shdig[i]));
41275:
41276: 3. writeln(IntToHex(CRC32OfFile(exepath+'\\maxbox4.exe'),4));
41277: 4. writeln(shaltohex(SHA1ofStr(loadStringOfFile(exepath+'maxbox4.exe'))))
41278:
41279: I have to get only the numbers, without the hyphen and the dots, to my calcs worth.
41280: sr := '123.456.789-00'
41281: writeln(ReplaceRegExpr('\\D',sr,'',true))

```

```

41282:
41283: Func GetLocalStream(C_form:Str; const lat,long:Str):boolean;
41284: var encodedURL, UrlMapQuestAPI, bufstr:Str;
41285: mStream: TMemoryStream;
41286: idhttp: TIdHttp; // THTTPEnd;
41287: len: Int;
41288: begin
41289: UrlMapQuestAPI:= 'http://192.168.1.53:9000';
41290: encodedURL:= UrlMapQuestAPI;
41291: idHTTP:= TIdHTTP.Create(NIL)
41292: mStream:= TMemoryStream.create;
41293: try
41294: //HttpGet(EncodedURL, mStream); {WinInet}
41295: idHTTP.Get1(EncodedURL, mStream)
41296: mStream.Position:= 0;
41297: writeln('stream size: '+inttostr(mStream.size));
41298: //mStream.memory;
41299: len:= mStream.Size - mStream.Position;
41300: SetLength(bufstr, len);
41301: mStream.readbuffer(bufstr, len)
41302: writeln('debug stream local back: '+bufstr)
41303: finally
41304: encodedURL:= '';
41305: mStream.Free;
41306: idHTTP.Disconnect
41307: idHTTP.Free;
41308: result:= true;
41309: end;
41310: end;
41311:
41312: 39/40 PE imports of V maxbox 4.2.5.10
41313:
41314: 1[+] AVICAP32.DLL 2[+] AVICAP32.dll 3[+] GLU32.dll 40[++] dmath.dll
41315: 4[+] IMAGEHLP.DLL 6[+] MSVCRT.DLL 7[+] MSVFW32.DLL
41316: 8[+] OpenGL32.dll 9[+] SHFolder.dll 11[+] URLMON.DLL 10[+] SHfolder.dll
41317: 12[+] advapi32.dll 13[+] comctl32.dll 14[+] comdlg32.dll
41318: 15[+] gdi32.dll 16[+] imagehlp.dll 17[+] imm32.dll
41319: 18[+] iphlapi.dll 19[+] kernel32.dll 20[+] mpr.dll
41320: 21[+] msacm32.dll 24[+] ole32.dll 25[+] oleacc.dll
41321: 26[+] oleaut32.dll 27[+] oledlg.dll 28[+] opengl32.dll
41322: 29[+] shell32.dll 30[+] shlwapi.dll 31[+] user32.dll
41323: 32[+] usp10.dll 33[+] version.dll 34[+] winhttp.dll
41324: 35[+] wininet.dll 36[+] winmm.dll 37[+] winspool.drv 5[+] KERNEL32.DLL
41325: 38[+] ws2_32.dll 39[+] wsock32.dll 22[+] msimg32.dll 23[+] netapi32.dll
41326:
41327: Magic literal PE32 executable for MS Windows (GUI) Intel 80386 32-bit
41328: *****
41329: Microsoft Windows SDK for Windows 7 and .NET Framework 4 Interface (UI Automation)
41330: Overall test results Number of tests 68 Verify
41331: Passed 67 Failed 1
41332: Unexpected Error 0
41333:
41334: UI Tests caused Unexpected Error: 0
41335: Tests Failed: 1
41336: Test Name : ControlType.Window:SetFocus.1
41337: Result Status: Failed
41338: Test Priority: Pri1
41339: Description: Precondition: Some controls are not persisted after they loose focus (ie, Floating edit in
Excel), don't tests these
41340: !! Exception !!
41341: Message: Step 12 : Focus event was not fired and was expected to be fired
41342: Unexpected: false Known Bug: false Incorrect Configuration: false
41343:
41344: 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)
41345:
41346: Tests Passed:67/68
41347: Test Name:ControlType.Window:FocusedElement.1
41348: Test Name:ControlType.Window:GetRuntimeIdType.1
41349: Test Name:ControlType.Window:GetSupportedPatterns.1
41350: Test Name:ControlType.Window:GetSupportedProperties.1
41351: Test Name:ControlType.Window:GetType.1
41352: Test Name:ControlType.Window:HwndWindowRect.1.MSAA
41353: Test Name:ControlType.Window:HwndWindowRect.2.UIA
41354: Test Name:ControlType.Window:LocalizedControlType.1.8.1
41355: Test Name:ControlType.Window:ToString.1
41356: Test Name:ControlType.Window:AutomationIdProperty.1.7.1
41357: Test Name:ControlType.Window:AutomationIdProperty.1.7.8
41358: Test Name:ControlType.Window:BoundingRect.1
41359: Test Name:ControlType.Window:ClickablePointProperty.1
41360: Test Name:ControlType.Window:ControlTypeProperty.1
41361: Test Name:ControlType.Window:GetClickablePoint.2

```

```

41362: Test Name:ControlType.Window:GetHashCode.1
41363: Test Name:ControlType.Window:IsKeyboardFocusable.1.1.1
41364: Test Name:ControlType.Window:IsPasswordProperty.1.3.2
41365: Test Name:ControlType.Window:KeyboardFocusable.1.1.2
41366: Test Name:ControlType.Window:Navigation.1
41367: Test Name:ControlType.Window:Navigation.10
41368: Test Name:ControlType.Window:Navigation.11
41369: Test Name:ControlType.Window:Navigation.2
41370: Test Name:ControlType.Window:Navigation.3
41371: Test Name:ControlType.Window:Navigation.4
41372: Test Name:ControlType.Window:Navigation.5
41373: Test Name:ControlType.Window:Navigation.6
41374: Test Name:ControlType.Window:Navigation.7
41375: Test Name:ControlType.Window:Navigation.8
41376: Test Name:ControlType.Window:Navigation.9
41377: Test Name:ControlType.Window:Navigation.ControlFirstChild.1
41378: Test Name:ControlType.Window:Navigation.ControlLastChild.1
41379: Test Name:ControlType.Window:Navigation.ControlNextSibling.1
41380: Test Name:ControlType.Window:Navigation.ControlParent.1
41381: Test Name:ControlType.Window:Navigation.ControlPreviousSibling.1
41382: Test Name:ControlType.Window:SetFocus.10
41383: Test Name:ControlType.Window:SetFocus.3
41384: Test Name:ControlType.Window:AutomationElement.PropertyChange.Enabled.1
41385: Test Name:ControlType.Window:AutomationElement.PropertyChange.Name.1
41386: Test Name:ControlType.Window:GetCurrentPattern.1
41387: Test Name:ControlType.Window:GetCurrentPropertyValue.1
41388: Test Name:ControlType.Window:SetFocus.4
41389: Test Name:ControlType.Window:BulkAdd.1
41390: Test Name:ControlType.Window:BulkRemove.1
41391: Test Name:ControlType.Window:ChildAdd.1
41392: Test Name:ControlType.Window:ChildRemove.1
41393: Test Name:ControlType.Window:Invalidate.1
41394: Test Name:ControlType.Window:Reorder.1
41395: Test Name:ControlType.Window:TestContentView
41396: Test Name:ControlType.Window:TestControlPatterns
41397: Test Name:ControlType.Window:TestControlProperties
41398: Test Name:ControlType.Window:TestControlView
41399: Test Name:ControlType.Window:CanRotate.1.6
41400: Test Name:ControlType.Window:Move.2.1
41401: Test Name:ControlType.Window:Move.2.2
41402: Test Name:ControlType.Window:Move.2.8
41403: Test Name:ControlType.Window:Move.2.13
41404: Test Name:ControlType.Window:Move.2.17
41405: Test Name:ControlType.Window:Move.2.18
41406: Test Name:ControlType.Window:Move.2.19
41407: Test Name:ControlType.Window:Move.2.3
41408: Test Name:ControlType.Window:Move.2.4
41409: Test Name:ControlType.Window:Move.2.5
41410: Test Name:ControlType.Window:Window.MaximizableProperty.S.6.1
41411: Test Name:ControlType.Window:Window.MaximizableProperty.S.6.2
41412: Test Name:ControlType.Window:Window.MinimizableProperty.S.7.2
41413: Test Name:ControlType.Window:Window.ModalProperty.S.10.1
41414:
41415: Func GetCachedFileFromURL(strUL:Str; var strLocalFile:Str):Bool;
41416: Func IAddrToHostName(const IP:Str):Str;
41417: Func GetIEHandle(WebBrowser: TWebbrowser; ClassName:Str): HWND;
41418: Func GetTextFromHandle(WinHandle: THandle):Str;
41419: Proc Duplicate_Webbrowser(WB1, WB2: TWebbrowser);
41420: Func FillWebForm(WebBrowser:TWebBrowser;FieldName:str;Value:str):Bool;
41421: Proc WB_LoadHTML(WebBrowser: TWebBrowser; HTMLCode:Str);
41422: Func NetSend(dest, Source, Msg:Str): Longint; overload;
41423: Func RecordsetFromXML2(const XML:Str): variant;;
41424: Func RecordsetToXML2(const Recordset: variant):Str;;
41425: Func GetCharEncoding( alias :Str; var _name :Str) : Int;
41426: Func MicrosoftCodePageToMIMECharset( cp : word) :Str;
41427: Func MicrosoftLangageCodeToISOCODE( langcode : Int) :Str;
41428: Proc CopyHTMLToClipboard(const str:Str; const htmlStr:Str = ');
41429: Func RFC1123ToDateTime(Date:Str): TDateTime;
41430: Func DateTimeToRFC1123(aDate: TDateTime):Str;
41431: Proc CopyHTMLToClipboard(const str:Str; const htmlStr:Str);
41432: Proc DumpDOSHeader(const h: IMAGE_DOS_HEADER; Lines: TStrings);
41433: Proc DumpPEHeader(const h: IMAGE_FILE_HEADER; Lines: TStrings);
41434: Proc DumpOptionalHeader(const h: IMAGE_OPTIONAL_HEADER; Lines: TStrings);
41435: Func checkSystem:Str;
41436: Func getSystemReport:Str;
41437:
41438: //Commonly used Delphi WinAPI routines
41439: http://www.rosseid.be/DRO/Delphi/Delphi%20WinAPI.htm
41440:
41441: source of the new units: http://sourceforge.net/projects/maxbox/files/Docu/SourceV4/
41442:
41443: http://www.slideshare.net/maxkleiner1/codesign-2015
41444: http://basta_2015_speaker_max_kleiner
41445: http://www.arduino.cc/en/Tutorial/ASCIITable
41446: http://www.vwlowen.co.uk/arduino/usb-digital/pc-control.htm
41447: http://www.yunqa.de/delphi/doku.php/products/regex/syntax#quantifiers
41448: http://elib.uni-stuttgart.de/opus/volltexte/2008/3440/pdf/diss_kroell_hs.pdf
41449: http://www.softwareschule.ch/download/exampleedition2016.zip
41450:

```

```

41451: Published Doc maXbox Example Edition 2015/2016/2017/2018
41452:
41453: \example_edition\01_Algorithm;
41454: \example_edition\02_Graphics;
41455: \example_edition\03_Games;
41456: \example_edition\04_Multimedia;
41457: \example_edition\05_Internet;
41458: \example_edition\06_Communication;
41459: \example_edition\07_Geographical;
41460: \example_edition\08_Operating;
41461: \example_edition\09_Database;
41462: \example_edition\10_Science;
41463: \example_edition\11_Embedded;
41464: \example_edition\12_Security;
41465: \example_edition\13_General;
41466: \example_edition\14_Energy;
41467: \example_edition\15_Transport;
41468: \example_edition\16_Robotics;
41469:
41470: Ref:
41471: if IsValidPeFile(exepath+'maxbox3.exe') then begin
41472:   x1:= ComputePEChecksum(exepath+'maxbox3.exe; // original filename
41473:   x2:= ComputePEChecksum(exepath+'maxbox3.exe;
41474: end;
41475: WriteLn('Checksum 1: ' + itoa(x1)+ #13#10+'Checksum 2: ' + itoa(x2));
41476:
41477: if ConnectDrive('Z:', '\\MAXBOX8\Users\Public', True, True) = NO_ERROR then
41478:   writeln('Net Share Z:\ Connected;
41479:   DisconnectNetDrive('Z:', True, True, True);
41480:
41481: ComTerminal:= TCustomComTerminal.create(self);
41482: EditComTerminal(comterminal); //TComTrmSetForm comterminal.Free;
41483: LastSysErrorMessage :Str; LastSysErrorMessageA : Ansistr;
41484:
41485: writeln(shaltohex(SHA1ofStr(loadstringj(exepath+'maxbox4.exe'))))
41486: writeln(shaltohex(SHA1ofStr(loadstringoffile(exepath+'maxbox4.exe'))))
41487: writeln(strtoHex1(SHA1ofStr(lsof(exepath+'maxbox4.exe'))))
41488: writeln(shaltohex(SHA1ofStr(filetostring(exepath+'maxbox4.exe'))))
41489:
41490: SHA512 sr:= filetostring(Exepath+'maXbox4.exe')
41491: writeln(uppercase(SHA512DigesttoHEX(CalcSHA512(sr))))
41492: writeln(Base64EncodeStr((SHA512DigestAsString(CalcSHA512(sr))));
41493: base64:w4ueRtnUZ641dsy4yIISYvhMB9r0c9kc8YO9o5FgkWQRbayHLaZay7tjnt1JmhV3vck2a7Mec7wfK8PbHacrbQ==
41494: PE Checksum 1: 27767976 PE Checksum 2: 27767976
41495:
41496: // REST API Example
41497: procedure TOnlineRandomizerGetResultsFromWeb(qUrl: String);
41498: var FResultStream: TMemoryStream;
41499:   FList: TStringList;
41500:   FResultList: TStringList;
41501:   FSettings:TFormatSettings;
41502: begin
41503:   FResultList:= TStringList.Create;
41504:   //FSettings:= TFormatSettings.Create;
41505:   //FSettings.DecimalSeparator := '.';
41506:   FResultStream:= TMemoryStream.Create;
41507:   FList:= TStringList.Create;
41508:   //httpObj.Get1(qUrl,FResultStream);
41509:   HTTPGet(qURL, FResultStream)
41510:   FResultStream.Position:= 0;
41511:   FList.LoadFromStream(FResultStream);
41512:   FResultStream.Free;
41513:   FResultList.AddStrings(FList);
41514:   writeln('online random: '+flots(StrToFloat(FResultList[3])));
41515:   writeln('online random: '+flots(StrToFloat(FResultList[4])));
41516:   writeln('online random: '+flots(StrToFloat(FResultList[5])));
41517:   FResultList.Free;
41518:   FList.Free;
41519: end;
41520:
41521: //var FDecUrlPattern:String;
41522: procedure TOnlineRandomizerGetRandomDecimals(dPrecision, dCount: Integer);
41523: var qUrl, FDecUrlPattern: String;
41524: begin
41525:   FDecUrlPattern:= 'https://www.random.org/decimal-fractions/'+
41526:   '?dec=%d&num=%d&col=1&format=plain&rnd=new';
41527:   qUrl:= Format(FDecUrlPattern,[dPrecision, dCount]);
41528:   TOnlineRandomizerGetResultsFromWeb(qUrl);
41529: end;
41530:
41531: C:\maXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
41532:
41533: //////////////////////////////////////
41534:
41535:
41536: https://www.metadefender.com/#!/results/file/933baa9823514320806c9533482a3b89/regular/analysis
41537: https://www.virustotal.com/en/file/b80b0bfef22c6b4be3dbc4af984ca897144895f3c1e162f3ad7895d14fb4e667/analysis/1476479319/
41538: Name Virtual address Virtual size Raw size Entropy MD5

```

```

41539: .text 4096 21923324 21923328 6.61 0449d5754ddb069630808bfc6bb356ef
41540: .itext 21929984 52108 52224 6.64 0226e6c46fcf9cc23805e75784f48b4b
41541: .data 21983232 407520 407552 5.25 91d5dac4714aa4119ea18fb5c4ed250d
41542: .bss 22392832 390424 0 0.00 d41d8cd98f00b204e9800998ecf8427e
41543: .idata 22786048 59722 59904 5.53 40cc92b8f087c512630bf807d4d20fe1
41544: .edata 22847488 77 512 0.89 0fcb1fd6f1dde231aa401cffffaa8cd63
41545: .tls 22851584 484 0 0.00 d41d8cd98f00b204e9800998ecf8427e
41546: .rdata 22855680 24 512 0.28 916275c8032d420cb22a05c2086b7e93
41547: .reloc 22859776 1378116 1378304 6.76 51ab0f1c39aa5fc52a7bc82557df173e
41548: .rsrc 24240128 4132864 4132864 5.37 c49d04317b7a6aad424538ff5ede5c0
41549: CodeSize 21975552 InitializedDataSize 5979648
41550: TimeStamp 2019-08-06 08:06:39
41551: EntryPoint 22518720
41552: OriginalFileName maxXbox4_6.exe
41553: MIMETYPE application/octet-stream
41554:
41555: Article concerning maxXbox and mapbox at PHP Magazine 1.2017
41556: https://entwickler.de/php-magazin/php-magazin-1-17-301136.html
41557: WineHQ Update:
41558: https://appdb.winehq.org/objectManager.php?sClass=application&iId=17937
41559: http://www.slideshare.net/maxkleiner1/maxbox-starter46-work-with-wine
41560: https://www.academia.edu/30401550/maxXbox_Starter_46_Work_with_WineHQ
41561: https://appdb.winehq.org/appimage.php?iId=50376
41562: https://www.scribd.com/document/333983581/maxXbox-starter46-work-with-WineHQ
41563: http://www.softwareschule.ch/download/maxbox_starter46.pdf
41564:
41565: Update for Windows 10 Version 1511 for x64-based Systems (KB3152599)
41566: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3200970)
41567: Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3201845)
41568: Update for Windows 10 Version 1607 for x64-based Systems (KB3211320)
41569: Intel Corporation - Display - 11/10/2016 12:00:00 AM - 20.19.15.4549
41570: 2017-10 Cumulative Update for Win 10 Version 1703 for x64-based Systems (KB4041676)
41571: 2017-12 Cumulative Update for Win 10 Version 1703 for x64-based Systems (KB4053580)
41572: Feature update to Windows 10, version 1709
41573: Feature update to Windows 10, version 1803 amd64 2019-04
41574: Feature update to Windows 10, version 1903
41575: Feature update to Windows 10, version 1909
41576: November 12, 2019-KB4524570 (OS Builds 18362.476 and 18363.476)
41577: Version: 1903-OS Build 18362.535 and 1909-OS Build 18363.535
41578: Done Adding Additional Store Successfully signed: maxXbox4.exe
41579: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maxXbox4.exe
41580: Verifying: maxXbox4.exe
41581: Successfully verified: maxXbox4.exe
41582: Number of files successfully Verified: 1
41583: Number of warnings: 0
41584: Number of errors: 0C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
41585: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
41586: https://maxbox4.wordpress.com/
41587: http://www.softwareschule.ch/teelicht.jpg
41588:
41589: Verifying: maxXbox4.exe 4.7.1.80
41590: Signature Index: 0 (Primary Signature)
41591: Hash of file (shal): 9D181955E579DCFF6920CEEC32749F0E43F2C157
41592: Signing Certificate Chain:
41593: Issued to: maxXboxCertAuth
41594: Issued by: maxXboxCertAuth
41595: Expires: Sun Jan 01 00:59:59 2040
41596: SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
41597: Issued to: maxXbox4signer
41598: Issued by: maxXboxCertAuth
41599: Expires: Sun Jan 01 00:59:59 2040
41600: SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
41601: Issued to: maxXbox4exe
41602: Issued by: maxXbox4signer
41603: Expires: Sun Jan 01 00:59:59 2040
41604: SHA1 hash: FOEB0CA218C5707FAC78921F81092CECA12AD0E9
41605: The signature is timestamped: Tue Dec 03 10:51:06 2019
41606: Timestamp Verified by:
41607: Issued to: Starfield Root Certificate Authority - G2
41608: Issued by: Starfield Root Certificate Authority - G2
41609: Expires: Fri Jan 01 00:59:59 2038
41610: SHA1 hash: B51C067CEE2B0C3DF855AB2D92F4FE39D4E70F0E
41611: Issued to: Starfield Secure Certificate Authority - G2
41612: Issued by: Starfield Root Certificate Authority - G2
41613: Expires: Sat May 03 08:00:00 2031
41614: SHA1 hash: 7EDC376DCFD45E6DDF082C160DF6AC21835B95D4
41615: Issued to: Starfield Timestamp Authority - G2
41616: Issued by: Starfield Secure Certificate Authority - G2
41617: Expires: Tue Sep 17 08:00:00 2024
41618: SHA1 hash: E8551398FF530A9278FD9818E448CB333F67924D
41619:
41620: Successfully verified: maxXbox4.exe
41621: Number of files successfully Verified: 1
41622: Number of warnings: 0
41623: Number of errors: 0
41624: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
41625:
41626: *****
41627: Release Notes maxXbox 4.7.2.82 April 2020 mX47

```



```

41628: *****
41629: 1254 unit uPSI_MaskEdit.pas      FCL
41630: 1255 unit uPSI_SimpleRSSTypes;   BlueHippo
41631: 1256 unit uPSI_SimpleRSS;        BlueHippo
41632: 1257 unit uPSI_psULib.pas        Prometheus
41633: 1258 unit uPSI_psUFinancial;     Prometheus
41634: 1259 uPSI_PsAPI_2.pas            mX4
41635: 1260 uPSI_PersistSettings_2     mX4
41636: Totals of Func Calls: 32339
41637: for 4.7.2.82
41638: https://sourceforge.net/projects/prometheuslibra/
41639:
41640: CDS Func CreateBlobStream(Field: TField; Mode: TBlobStreamMode): TStream;
41641:      Func BookmarkValid(Bookmark: TBookmark): Boolean;
41642:      Func CompareBookmarks(Bookmark1, Bookmark2: TBookmark): Integer;
41643:
41644: Proc ModifyFontsFor(ctrl: TWinControl);
41645: Proc ModifyFontsFor(ctrl: TWinControl; afontname: Str);
41646: Proc GetAccessTimeOut(var bTimeOut: Boolean; var iTimeOutTime: Integer);
41647: Func GetParentProcessName(): Str;
41648: writeln('ParentProcessName: '+GetProcessName(GetParentProcessID(getprocessID)));
41649: Func wFillRect(hDC: HDC; const lprc: TRect; hbr: HBRUSH): Integer; stdcall;
41650: Func wFrameRect(hDC: HDC; const lprc: TRect; hbr: HBRUSH): Integer; stdcall;
41651: Proc GetIpAddresses(Results: TStrings); overload;
41652: Proc GetIpAddresses(Results: TStrings; const HostName: Ansistr);
41653: S.RegisterDelphiFunction(@GetIPAddresses, 'GetIPs', cdRegister);
41654: RegisterMethod(@TStringGrid.CellRect, 'CellRect');
41655: RegisterMethod(@TStringGrid.MouseToCell, 'MouseToCell');
41656: Proc SendKeyVar(AppName: Str; key: Variant);
41657: http://www.scalabium.com/faq/dct0069.htm
41658: Proc GetComponentNames(lst: TStrings);
41659: Proc AngleTextOut(ACanvas: TCanvas; Angle, X, Y: Integer; Str: Str);
41660: Proc PrintText2Printer(s: Str);
41661: Proc UASwapEndian(var UC: TUArrDW);
41662: Func SliceByteArray(const B: array of Byte; Start, Len: Integer): TBytes;
41663: Proc TailOfFile(aFilename: Str; aTailContainer: TStrings; MaxNumLines: Integer=5);
41664: Func CloneByteArray(const B: array of Byte): TBytes;
41665: Func PopByteArray(var A: TBytes): Byte;
41666: Proc PushByteArray(const B: Byte; var A: TBytes);
41667: Proc AppendByteArray(var B1: TBytes; const B2: array of Byte);
41668: Func ShiftByteArray(var A: TBytes): Byte;
41669:
41670: *****
41671: Release Notes maXbox 4.7.4.64 June 2020 mX47
41672: *****
41673: add Tutor 57 - 72
41674: add NoGUI Shell Tutorial 71 and 70 Units
41675: 1307 unit uPSI_statmach, {StateMachine}
41676: 1308 uPSI_uTPLb_RSA_Primitives,
41677: 1309 unit uPSI_UMatrix, //for Tensorflow dll
41678: 1310 uPSI_DXUtil,
41679: 1311 uPSI_crlfParser,
41680: 1312 unit uPSI_DCPhase64;
41681: 1313 unit uPSI_FlyFilesUtils;
41682: 1314 uPSI_PJConsoleApp.pas
41683: 1315 uPSI_PJStreamWrapper.pas
41684: 1316 uPSI_LatLonDist, //DFF
41685: 1317 uPSI_cHash2.pas //Fundamentals SHA512
41686: 1318 uPSI_ZLib2.pas //compressor
41687: 1319 unit uPSI_commDriver
41688: 1320 unit uPSI_PXLNetComs.pas //PXL
41689: 1321 unit uPSI_PXLTiming.pas //PXL
41690: 1322 uPSI_Odometer.pas
41691: 1323 unit uPSI_UIntList2;
41692: 1324 uPSI_UIntegerpartition.pas
41693: 1325 unit uPSI_idPHPRunner.pas //prepare for PHP4D
41694: 1326 unit uPSI_idCGIRunner.pas
41695: 1327 uPSI_DrBobCGI, //4.7.1.20
41696: 1328 uPSI_OverbyteIcsLogger,
41697: 1329 uPSI_OverbyteIcsNntpCli, testset
41698: 1330 uPSI_OverbyteIcsCharsetUtils,
41699: 1331 uPSI_OverbyteIcsMimeUtils,
41700: 1332 uPSI_OverbyteIcsUrl(CL: TPSPascalCompiler);
41701: 1333 uPSI_uWebSocket.pas
41702: 1334 uPSI_KhFunction.pas
41703: 1335 uPSI_ALOpenOffice.pas
41704: 1336 unit uPSI_ALLibPhoneNumber
41705: 1337 unit uPSI_ALPhpRunner2;
41706: 1338 unit uPSI_ALWebSpider2;
41707: 1339 unit uPSI_ALFcnHTML2; // RunJavaScript2
41708: 1340 unit uPSI_ALExecute2.pas
41709: 1341 uPSI_ALIsapiHTTP.pas
41710: 1342 uPSI_ALOpenOffice_Routines
41711: 1343 unit uPSI_uUsb;
41712: 1344 uPSI_uWebcam.pas
41713: 1345 uPSI_PersistSettings.pas //fixing
41714: 1346 uPSI_uTPLb_MemoryStreamPool.pas
41715: 1347 uPSI_uTPLb_Signatory.pas
41716: 1348 unit uPSI_uTPLb_Constants.pas //TurboPower

```

```

41717: 1349 uPSI_uTPLb_Random.pas
41718: 1350 unit uPSI_uTPLb_PointerArithmetic;
41719: 1351 unit uPSI_EwbCoreTools.pas
41720: 1352 unit uPSI_EwbUrl.pas
41721: 1353 unit uPSI_SendMail_For_Ewb.pas
41722: 1354 unit uPSI_MaskEdit.pas FCL
41723: 1355 unit uPSI_SimpleRSSTypes; BlueHippo
41724: 1356 unit uPSI_SimpleRSS; BlueHippo
41725: 1357 unit uPSI_psULib.pas Prometheus
41726: 1358 unit uPSI_psUFinancial; Prometheus
41727: 1359 uPSI_PsAPI_2.pas mX4
41728: 1360 uPSI_PersistSettings_2 mX4
41729: 1361 uPSI_rfc1213util2.pas IP
41730: 1362 uPSI_JTools.pas JCL
41731: 1363 unit uPSI_neuralbit.pas CAI
41732: 1364 unit uPSI_neuralab.pas CAI
41733: 1365 unit uPSI_winsvc2.pas TEK
41734: 1366 unit uPSI_wmiserv2.pas TEK
41735: 1367 uPSI_neuralcache.pas CAI
41736: 1368 uPSI_neuralbyteprediction CAI
41737: 1369 unit uPSI_USolarSystem; glscene.org
41738: 1370 uPSI_USearchAnagrams.pas DFF
41739: 1371 uPSI_JsonsUtilsEx.pas Randolph
41740: 1372 unit uPSI_Jsons.pas Randolph
41741: 1373 unit uPSI_HashUnit; DFF
41742: 1374 uPSI_U_Invertedtext.pas DFF
41743: 1275 unit uPSI_Bricks; Dendron
41744: 1376 unit uPSI_lifeblocks.pas Dendron
41745:
41746: Bugfix: Pagecontrol, TTabcontrol with change event TTabChangingEvent, TGetSiteInfoEvent,
41747:   Json extension with objects and arrays
41748:   TCheckBox.Style Property TListBoxStyle Enumeration
41749:   glsincos11 glsincos0 fix uPSI_UMatrix; Gaussian_Elimination fix array[1..30] of Extended
41750:   more ico and bmp resources, fix Textmetric type and strtocars() alias StrtoCharSet()
41751:   shortcut Ctrl+Alt+M to jump to output (memo2) console
41752:
41753: Func MercScaling(Long0,lat0,long1,lat1:extended;x0,y0,x1,y1:integer) :TMercScalingRec');
41754: Func LonglatToPlotPt(Long,lat:Extended; Scalerec:TMercScalingrec):TPoint');
41755: Func PlotPtToLonglat(PlotPt:TPoint; Scalerec:TMercScalingrec):TRealPoint');
41756: Func SphericalEarthDistance(lat1,lon1,lat2,lon2:extended;Units:integer):extended');
41757: Proc PrintUsingShell(psFileName :str);
41758: Func LocalExecute32(FileName:Str;Wait:bool;Visibility:int;lWaitFor:Card=INFINITE):int;
41759:
41760: TidCGIRunner component allows to execute CGI scripts using
41761: Indy TidHTTPServer component.
41762: // Project: 1307 State Machine
41763: // Module: statmach.pas
41764: // Description: Visual Finite State Machine.
41765: // Version: 2.2a, Release: 6 preprocessor, compiler, interpreter
41766: Func VarArrayToStr(const vArray: variant):Str;
41767: Func GetWMIObject(const objectName:Str): IDispatch; //create the Wmi instance
41768: Func GetAntiVirusProductInfo: TStringlist;
41769: Proc GetWaveOutDevices(DeviceNames: TStrings);
41770: Proc GetMIDIOutDevices(Devices: TStrings);
41771: Proc SetDecimalSeparator2(Ch: Char);
41772: Totals of Func Calls: 32356
41773: Proc ResizeBitmap(Bitmap:TBitmap;Width,Height:Integer;Background: TColor);
41774: Func ProcessCount(const ExeName:str): Integer;
41775: Func MaxWidthOfStrings(const Strings:Classes.TStrings;const Font:Graphics.TFont):Integer;
41776: Proc ScreenShotMonitor(var Bitmap:TBitmap;const MonitorNum:Integer;const DrawCursor:Bool;const
Quality:TPixelFormat);
41777:
41778: https://github.com/maxkleiner/maxbox/blob/master/logisticregression2.ipynb
41779:
41780: ---- mX4 bigbitbox code_cleared_checked IMPORTS DLL LIB ----
41781: AVICAP32.DLL 01 [+] AVICAP32.DLL
41782: AVICAP32.dll 02 [+] AVICAP32.dll
41783: GLU32.dll 03 [+] GLU32.dll
41784: IMAGEHLP.DLL 04 [+] IMAGEHLP.DLL
41785: KERNEL32.DLL 05 [+] KERNEL32.DLL
41786: MSVCRT.DLL 06 [+] MSVCRT.DLL
41787: MSVFW32.DLL 07 [+] MSVFW32.DLL
41788: OpenGL32.dll 08 [+] OpenGL32.dll
41789: SHFolder.dll 09 [+] SHFolder.dll
41790: SHfolder.dll 10 [+] SHfolder.dll
41791: URLMON.DLL 11 [+] URLMON.DLL
41792: advapi32.dll 12 [+] advapi32.dll
41793: comctl32.dll 13 [+] comctl32.dll
41794: comdlg32.dll 14 [+] comdlg32.dll
41795: gdi32.dll 15 [+] gdi32.dll
41796: imagehlp.dll 16 [+] imagehlp.dll
41797: imm32.dll 17 [+] imm32.dll
41798: iphlapi.dll 18 [+] iphlapi.dll
41799: kernel32.dll 19 [+] kernel32.dll
41800: mpr.dll 20 [+] mpr.dll
41801: msacm32.dll 21 [+] msacm32.dll
41802: msimg32.dll 22 [+] msimg32.dll
41803: netapi32.dll 23 [+] netapi32.dll
41804: ole32.dll 24 [+] ole32.dll

```

```

41805: oleacc.dll      25 [+] oleacc.dll
41806: oleaut32.dll     26 [+] oleaut32.dll
41807: oledlg.dll      27 [+] oledlg.dll
41808: opengl32.dll    28 [+] opengl32.dll
41809: shell32.dll     29 [+] shell32.dll
41810: shlwapi.dll     30 [+] shlwapi.dll
41811: user32.dll      31 [+] user32.dll
41812: usp10.dll       32 [+] usp10.dll
41813: version.dll     33 [+] version.dll
41814: winhttp.dll     34 [+] winhttp.dll
41815: wininet.dll     35 [+] wininet.dll
41816: winmm.dll       36 [+] winmm.dll
41817: winspool.drv   37 [+] winspool.drv
41818: ws2_32.dll      38 [+] ws2_32.dll
41819: wsock32.dll    39 [+] wsock32.dll
41820: fannfloat.dll  40 [++] fannfloat.dll
41821: dmath.dll      41 [++] dmath.dll
41822: midas.dll      42 [++] midas.dll
41823:
41824: [+] AVICAP32.DLL      [+] AVICAP32.dll      [+] GLU32.dll      [+] IMAGEHLP.DLL
41825: [+] KERNEL32.DLL     [+] MSVCRT.DLL      [+] MSVFW32.DLL   [+] OpenGL32.dll
41826: [+] SHFolder.dll     [+] SHfolder.dll    [+] URLMON.DLL    [+] advapi32.dll
41827: [+] comctl32.dll     [+] comdlg32.dll    [+] gdi32.dll     [+] imagehlp.dll
41828: [+] imm32.dll        [+] iphlapi.dll     [+] kernel32.dll  [+] mpr.dll
41829: [+] msacm32.dll      [+] msimg32.dll     [+] netapi32.dll  [+] ole32.dll
41830: [+] oleacc.dll       [+] oleaut32.dll    [+] oledlg.dll    [+] opengl32.dll
41831: [+] shell32.dll      [+] shlwapi.dll     [+] user32.dll    [+] usp10.dll
41832: [+] version.dll      [+] winhttp.dll     [+] wininet.dll   [+] winmm.dll [+] midas.dll
41833: [+] winspool.drv     [+] ws2_32.dll      [+] wsock32.dll   [+] dmath.dll [+] fannfloat.dll
41834:
41835: Exports      CreateIncome      ExePath      GetASCII      GetBiosVendor
41836:      GetMemoryInfo  GetOSName      GetProcessorName  GetScriptName2
41837:      GetScriptPath2 IsCOMConnected IsInternet      RemainingBatteryPercent
41838:      UpTime        getBigPI       getHostIP      isSound
41839:
41840: mem01.CodeFolding.FoldRegions.Add begin','end''try','end repeat, until'case','end [$IFDEF'','$ENDIF'];
41841: mem01.InitCodeFolding; true, '{','}';
41842: writeln(GETDOSOutput('cmd.exe /c wmic cpu get name','C:\'));
41843: >>> Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
41844: V 4.7.4.64
41845: Totals of Func Calls: 32633
41846: SHA1: of 4.7.4.64 DA4C716E31E2A4298013DFFBDA7A98D48650B0C7
41847: CRC32: 3EB27A87: 28.2 MB (29,608,248) bytes
41848: V 4.7.5.20
41849: Totals of Func Calls: 33282
41850: SHA1: of 4.7.5.20 D82EAD01C58738887661428F94B207DB1D8FAEB5
41851: CRC32: 203C82F0 29.5 MB (31,012,768) bytes
41852: V 4.7.5.80
41853: Totals of Calls: 33848
41854: SHA1: of 4.7.5.80 3E38A48072D4F828A4BE4A52320F092FE50AE9C3
41855: CRC32: 198E756B: 31344456 bytes
41856: [0] mX4 executed: 27/07/2021 19:05:10 Runtime:0:12:3.247 Memload: 39% use
41857: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool sign /f "maxbox4.exe
41858: .pfx" /p "password" /tr http://timestamp.comodoca.com /td SHA256 maxbox4.exe
41859: C:\maxXbox\EKON_BASTA\Windows Kits\10\bin\x64>signtool verify /v /pa maxbox4.exe
41860: For SHA256 the Parameter name has to be /tr. The /t is only valid for SHA1.
41861: ExecuteShell(OpenSSL_Path+'openssl.exe',
41862: 'pkcs12 -in'+OpenSSL_Path+'./certs/maxXboxCertAuth3.pfx -out'+OpenSSL_Path+'./certs/maxXboxCertAuth3.pfx.pem
-nodes')
41863: Verifying: maxbox4.exe 4.7.6.10 IV Compilation Timestamp 2022-01-14 13:20:37
41864: Entry Point 24467336 Contained Sections 10
41865:
41866: *****
41867: Release Notes maxbox 4.7.6.10 VIII August 2022 mX476
41868: *****
41869: Add 48 Units + 16 Tutorials
41870:
41871: 1441 unit uPSI_neuralgeneric.pas; CAI
41872: 1442 unit uPSI_neuralthread.pas; CAI
41873: 1443 unit uPSI_uSysTools; TuO
41874: 1444 unit uPSI_neuralsets; mX4
41875: 1445 unit uPSI_uWinNT.pas mX4
41876: 1446 unit uPSI_URungeKutta4.pas ICS
41877: 1447 unit uPSI_UrlConIcs.pas ICS
41878: 1448 unit uPSI_OverbyteIcsUtils.pas ICS
41879: 1449 unit uPSI_Numedit2 mX4
41880: 1450 unit uPSI_PsAPI_3.pas mX4
41881: 1451 unit uPSI_SeSHA256.pas
41882: 1452 unit IdHashMessageDigest_max2;
41883: 1453 unit uPSI_BlocksUnit.pas
41884: 1454 unit uPSI_DelticsCommandLine.pas
41885: 1455 unit uPSI_DelticsStrUtils;
41886: 1456 unit uPSI_DelticsBitFields;
41887: 1457 unit uPSI_DelticsSysUtils;
41888: 1458 unit uPSI_ALIniFiles2.pas
41889: 1459 unit uPSI_StarCalc2.pas
41890: 1460 unit uPSI_IdHashMessageDigest2.pas
41891: 1461 unit uPSI_U_Splines;
41892: 1462 unit uPSI_U_CoasterB.pas;

```

```

41893: 1463 U_SpringMass2.pas
41894: 1464 uPSI_MARSCoreUtils;
41895: 1465 unit uPSI_clJsonParser.pas
41896: 1466 unit uPSI_SynHighlighterPython.pas
41897: 1467 unit uPSI_DudsCommonDelphi;
41898: 1468 unit uPSI_AINNNeuron;
41899: 1469 unit uPSI_PJConsoleApp2;
41900: 1470 unit uPSI_PJPipeFilters2;
41901: 1471 unit uPSI_uHTMLBuilder;
41902: 1472 unit uPSI_PJPipe2;
41903: 1473 uPSI_WinApiDownload;
41904: 1474 uPSI_pxQRcode, //beta
41905: 1475 unit uPSI_neuralplanbuilder2
41906: 1476 unit uPSI_DelphiZXingQRcode;
41907: 1477 unit uPSI_RestJsonUtils;
41908: 1478 unit UtilsTimeCode;
41909: 1479 unit uPSC_classes2.pas; //TList
41910: 1480 unit uPSC_std2.pas
41911: 1481 unit uPSI_maxIniFiles.pas
41912: 1482 unit uROPSImports.pas
41913: 1483 unit uROPSServerLink.pas
41914: 1484 unit uPSI_KLibUtils;
41915: 1485 unit uPSI_PathFunc2; //inno setup
41916: 1486 unit KLibVC_Redist.pas;
41917: 1487 unit HTTPApp2.pas;
41918: 1488 unit uPSI_XCollection2;
41919:
41920: Total of Function Calls: 35597
41921: SHA1: 4.7.6.10 AA16F6BCD100F0AD16A39CD9741E78A96DF5F7F1
41922: CRC32: 412DE996: 31.7 MB (33,331,992 bytes)
41923: Amount of Functions: 21002
41924: Amount of Procedures: 12504
41925: Amount of Constructors: 1980
41926: Amount of Destructors: 14
41927: Totals of Calls: 35500
41928: SHA1: of 4.7.6.10 AEF0EE239713A3E142EE3480F86C30B9F0B872FD
41929: CRC32: 2D7A3208: 33300808 bytes
41930: [ ] mX4 executed: 19/03/2022 09:21:28 Runtime: 0:13:36.211 Memload: 42% use
41931: Amount of Functions: 21220
41932: Amount of Procedures: 12616
41933: Amount of Constructors: 1992
41934: Amount of Destructors: 14
41935: Totals of Calls: 35842
41936: SHA1: of 4.7.6.10 B8590FA3326B735E14D9687076B302461D8D49D9
41937: CRC32: CCB8C56: 33472792 bytes
41938: [ ] mX4 executed: 26/08/2022 09:53:21 Runtime: 0:14:24.171 Memload: 42% use
41939: [ ]
41940: Total of Function Calls: 35842
41941: SHA1: 4.7.6.10 B8590FA3326B735E14D9687076B302461D8D49D9
41942: CRC32: CCB8C56: 31.9 MB (33,472,792 bytes)
41943: recompiled
41944: 25/08/2022 22:41 483,539 fMain.dcu
41945: 25/08/2022 22:42 71,769 ALHttpClient2.dcu
41946: 25/08/2022 15:48 48,080 ALMultiPartParser.dcu
41947: 25/08/2022 22:11 20,366 ALWininetHttpClient2.dcu
41948: 25/08/2022 22:42 13,520 BlocksUnit.dcu
41949: 25/08/2022 23:24 30,875 MathsLib.dcu
41950: 25/08/2022 22:42 6,566 PXLTiming.dcu
41951: 25/08/2022 22:42 7,431 PythonAction.dcu
41952: 25/08/2022 22:42 279,159 PythonEngine.dcu
41953: 25/08/2022 22:42 6,446 SeSHA256.dcu
41954: 25/08/2022 20:14 45,539 UBigIntsV4.dcu
41955: 25/08/2022 22:42 66,215 uPSI_ALHttpClient2.dcu
41956: 25/08/2022 14:03 39,724 uPSI_ALHttpCommon.dcu
41957: 25/08/2022 23:22 7,652 uPSI_ALWininetHttpClient2.dcu
41958: 25/08/2022 22:42 28,590 uPSI_AzuliaUtils.dcu
41959: 25/08/2022 22:42 20,494 uPSI_BlocksUnit.dcu
41960: 25/08/2022 22:42 8,645 uPSI_PXLTiming.dcu
41961: 25/08/2022 22:42 175,123 uPSI_PythonEngine.dcu
41962: 25/08/2022 20:14 19,746 uPSI_UBigIntsV4.dcu
41963: 25/08/2022 22:42 46,239 VarPyth.dcu
41964: 25/08/2022 14:03 77,150 OverbyteIcsUtils.dcu
41965: [ ]
41966: Recompiled: dir /s /o N *.dcu | find "03/01/2022" > C:\maXbox\maxboxunitalldisk_sort.txt
41967: 03/01/2022 14:28 479,283 fMain.dcu
41968: 03/01/2022 14:28 13,506 BlocksUnit.dcu
41969: 03/01/2022 13:05 30,110 clJsonParser.dcu
41970: 03/01/2022 15:22 30,878 MathsLib.dcu
41971: 03/01/2022 14:28 6,566 PXLTiming.dcu
41972: 03/01/2022 14:28 7,417 PythonAction.dcu
41973: 03/01/2022 14:28 279,035 PythonEngine.dcu
41974: 03/01/2022 14:28 6,446 SeSHA256.dcu
41975: 03/01/2022 14:28 20,494 uPSI_BlocksUnit.dcu
41976: 03/01/2022 13:50 21,242 uPSI_clJsonParser.dcu
41977: 03/01/2022 14:11 59,984 uPSI_OverbyteIcsUtils.dcu
41978: 03/01/2022 14:28 8,645 uPSI_PXLTiming.dcu
41979: 03/01/2022 14:28 175,131 uPSI_PythonEngine.dcu
41980: 03/01/2022 10:50 37,866 uPSI_U_Splines.dcu
41981: 03/01/2022 14:50 30,632 uPSI_uLkJSON.dcu

```

```

41982: 03/01/2022 11:45 44,566 uPSI_VectorLists.dcu
41983: 03/01/2022 14:28 46,239 VarPyth.dcu
41984: 03/01/2022 14:11 40,901 OverbyteIcsMimeUtils.dcu
41985: 03/01/2022 14:28 77,151 OverbyteIcsUtils.dcu
41986: 03/01/2022 15:21 108,526 uPSI_UtilsMax4.dcu
41987: 02/01/2022 11:52 11,355 U_Splines.dcu
41988: 02/01/2022 11:27 24,281 uPSI_neuraldatasets.dcu
41989: 02/01/2022 12:15 77,381 uPSI_neuralvolume.dcu
41990: 01/01/2022 22:44 48,480 U_CoasterB.dcu
41991: 01/01/2022 22:44 34,098 uPSI_U_CoasterB.dcu
41992:
41993: mX4.7.6.10 VI
41994: 16/02/2022 21:29 50,625 uPSI_StrUtils.dcu
41995: 16/02/2022 21:36 59,984 uPSI_OverbyteIcsUtils.dcu
41996: 16/02/2022 15:49 16,601 uPSI_PJConsoleApp.dcu
41997: 16/02/2022 15:27 6,527 uPSI_PJPipe.dcu
41998: 16/02/2022 18:25 105,750 uPSI_PsAPI.dcu
41999: 16/02/2022 21:36 40,891 OverbyteIcsMimeUtils.dcu
42000:
42001: 17/02/2022 11:02 479,639 fMain.dcu
42002: 17/02/2022 09:55 30,889 uPSUtils.dcu
42003: 17/02/2022 10:56 3,723 AINNNNeuron.dcu
42004: 17/02/2022 11:02 13,524 BlocksUnit.dcu
42005: 17/02/2022 11:27 9,594 DudsCommonDelphi.dcu
42006: 17/02/2022 11:34 30,868 MathsLib.dcu
42007: 17/02/2022 11:02 6,568 PXLTiming.dcu
42008: 17/02/2022 11:02 7,434 PythonAction.dcu
42009: 17/02/2022 11:02 278,950 PythonEngine.dcu
42010: 17/02/2022 11:02 6,446 SeSHA256.dcu
42011: 17/02/2022 11:02 5,744 uPSI_AINNNNeuron.dcu
42012: 17/02/2022 11:02 20,494 uPSI_BlocksUnit.dcu
42013: 17/02/2022 11:23 5,442 uPSI_DudsCommonDelphi.dcu
42014: 17/02/2022 10:07 11,793 uPSI_PJEnvVars.dcu
42015: 17/02/2022 11:02 8,642 uPSI_PXLTiming.dcu
42016: 17/02/2022 11:02 175,123 uPSI_PythonEngine.dcu
42017: 17/02/2022 11:02 46,244 VarPyth.dcu
42018: 17/02/2022 11:02 77,152 OverbyteIcsUtils.dcu
42019:
42020: mX4.7.6.10 VII
42021: 17/03/2022 14:08 194,307 IFSI_WinFormlpuzzle.dcu
42022: 17/03/2022 20:30 46,854 uPSC_classes.dcu
42023: 17/03/2022 19:48 5,226 uPSC_std.dcu
42024: 17/03/2022 18:49 53,416 uPSR_classes.dcu
42025: 17/03/2022 20:30 19,528 uPSI_SynEditHighlighter.dcu
42026: 17/03/2022 16:03 11,784 uPSI_SynEditMiscProcs.dcu
42027: 17/03/2022 21:15 173,341 uPSI_TeEngine.dcu
42028: 17/03/2022 21:16 30,854 MathsLib.dcu
42029: 17/03/2022 20:13 77,627 uPSI_neuralvolume.dcu
42030: 17/03/2022 13:10 13,188 uPSI_UcomboV2.dcu
42031: 17/03/2022 13:26 14,813 uPSI_UTGraphSearch.dcu
42032: 17/03/2022 13:22 21,041 UTGraphSearch.dcu
42033: 17/03/2022 08:35 <DIR> n
42034:
42035: 18/03/2022 18:38 5,234 uPSC_std.dcu
42036: 18/03/2022 18:38 30,854 MathsLib.dcu
42037: 18/03/2022 18:33 77,624 uPSI_neuralvolume.dcu
42038:
42039: mX4.7.6.10 VIII
42040: Recompiled Units:
42041: c:\maxbox>dir /s /o N *.dcu | find "01/08/2022" > C:\maxbox\maxboxunitalldisk_sort.txt
42042: 28/07/2022 18:44 211,349 IFSI_SysUtils_max.dcu
42043: 28/07/2022 15:56 46,996 uPSC_classes.dcu
42044: 28/07/2022 15:23 49,036 uPSC_DB.dcu
42045: 28/07/2022 16:02 53,529 uPSR_classes.dcu
42046: 28/07/2022 15:23 89,402 uPSR_DB.dcu
42047: 28/07/2022 22:37 11,941 uPSI_SynEditMiscProcs.dcu
42048: 28/07/2022 22:37 26,896 uPSI_StrUtil.dcu
42049: 28/07/2022 14:51 9,910 uPSI_IdThreadSafe.dcu
42050: 31/07/2022 22:43 7,253 KLibTypes.dcu
42051: 31/07/2022 21:46 59,910 uPSI_OverbyteIcsUtils.dcu
42052: 31/07/2022 21:38 10,411 uPSI_XCollection.dcu
42053: 31/07/2022 21:38 40,887 OverbyteIcsMimeUtils.dcu
42054: 01/08/2022 12:28 482,562 fMain.dcu
42055: 01/08/2022 13:37 66,516 uPSI_HTTPApp.dcu
42056: 01/08/2022 12:28 13,517 BlocksUnit.dcu
42057: 01/08/2022 14:19 38,401 KLibUtils.dcu
42058: 01/08/2022 14:22 30,856 MathsLib.dcu
42059: 01/08/2022 12:28 6,566 PXLTiming.dcu
42060: 01/08/2022 12:28 7,430 PythonAction.dcu
42061: 01/08/2022 12:28 278,932 PythonEngine.dcu
42062: 01/08/2022 12:28 6,446 SeSHA256.dcu
42063: 01/08/2022 12:28 20,494 uPSI_BlocksUnit.dcu
42064: 01/08/2022 14:21 21,506 uPSI_KLibUtils.dcu
42065: 01/08/2022 12:22 7,927 uPSI_PathFunc.dcu
42066: 01/08/2022 12:28 8,645 uPSI_PXLTiming.dcu
42067: 01/08/2022 12:28 175,131 uPSI_PythonEngine.dcu
42068: 01/08/2022 12:28 46,238 VarPyth.dcu
42069: 01/08/2022 12:28 77,151 OverbyteIcsUtils.dcu
42070: 03/08/2022 18:25 482,909 fMain.dcu

```



```

42071: 03/08/2022 18:24 64,415 AzulialUtils.dcu
42072: 03/08/2022 18:25 13,523 BlocksUnit.dcu
42073: 03/08/2022 14:27 3,112 KlibConstants.dcu
42074: 03/08/2022 15:15 41,680 KLibWindows.dcu
42075: 03/08/2022 18:36 30,869 MathsLib.dcu
42076: 03/08/2022 18:25 6,568 PXLTiming.dcu
42077: 03/08/2022 18:25 7,434 PythonAction.dcu
42078: 03/08/2022 18:25 278,919 PythonEngine.dcu
42079: 03/08/2022 18:25 6,446 SeSHA256.dcu
42080: 03/08/2022 14:21 22,204 uPSI_ALHttpClient.dcu
42081: 03/08/2022 18:24 28,585 uPSI_AzulialUtils.dcu
42082: 03/08/2022 18:25 20,494 uPSI_BlocksUnit.dcu
42083: 03/08/2022 16:33 17,512 uPSI_KLibWindows.dcu
42084: 03/08/2022 18:15 59,933 uPSI_OverbyteIcsUtils.dcu
42085: 03/08/2022 18:25 8,644 uPSI_PXLTiming.dcu
42086: 03/08/2022 18:25 175,123 uPSI_PythonEngine.dcu
42087: 03/08/2022 16:35 108,026 uPSI_VectorGeometry.dcu
42088: 03/08/2022 18:25 46,235 VarPyth.dcu
42089: 03/08/2022 18:15 40,883 OverbyteIcsMimeUtils.dcu
42090: 03/08/2022 18:25 77,145 OverbyteIcsUtils.dcu
42091:
42092: Target Machine Intel 386 or later processors and compatible processors
42093: Compilation Timestamp 2022-08-03 16:37:05 UTC
42094: Entry Point 24656140
42095: Contained Sections 10
42096: Target Machine Intel 386 or later processors and compatible processors
42097: Compilation Timestamp 2022-08-29 09:10:00 UTC
42098: Entry Point 24725780
42099: Contained Sections 10
42100:
42101: mX4.7.6.10 X
42102: Recompiled Units:
42103: C:\maxXbox>dir /s /o N *.dcu | find "23/09/2022" > C:\maxXbox\maxboxunitallldisk_sort.txt
42104: 23/09/2022 16:15 483,806 fMain.dcu
42105: 23/09/2022 10:00 194,750 IFSI_WinFormIpuzzle.dcu
42106: 23/09/2022 12:43 81,932 uPSI_Chart.dcu
42107: 23/09/2022 11:39 97,061 uPSI_TeCanvas.dcu
42108: 23/09/2022 11:39 177,404 uPSI_TeEngine.dcu
42109: 23/09/2022 11:20 73,888 uPSI_TeeProcs.dcu
42110: 23/09/2022 16:16 71,782 ALHttpClient2.dcu
42111: 23/09/2022 16:16 13,522 BlocksUnit.dcu
42112: 23/09/2022 17:45 30,862 MathsLib.dcu
42113: 23/09/2022 16:16 6,568 PXLTiming.dcu
42114: 23/09/2022 16:16 7,434 PythonAction.dcu
42115: 23/09/2022 16:16 278,867 PythonEngine.dcu
42116: 23/09/2022 16:06 10,517 RestUtils.dcu
42117: 23/09/2022 16:16 6,446 SeSHA256.dcu
42118: 23/09/2022 16:16 66,632 uPSI_ALHttpClient2.dcu
42119: 23/09/2022 17:45 31,846 uPSI_AzulialUtils.dcu
42120: 23/09/2022 16:16 20,494 uPSI_BlocksUnit.dcu
42121: 23/09/2022 16:16 8,642 uPSI_PXLTiming.dcu
42122: 23/09/2022 16:16 175,123 uPSI_PythonEngine.dcu
42123: 23/09/2022 16:15 11,240 uPSI_RestUtils.dcu
42124: 23/09/2022 16:16 46,235 VarPyth.dcu
42125: 23/09/2022 15:11 63,422 uPSI_Series.dcu
42126:
42127: Total of Function Calls: 35958
42128: SHA1: 4.7.6.20 D4619B18E231839334AB0FE0B90D4018DC6276A7
42129: CRC32: 9E995FA2 32.2 MB (33,805,592 bytes)
42130: Compilation Timestamp 2022-10-17 09:27:44 UTC Signing time 17 Oct 2022 11:48:48
42131: Entry Point 24783132 - Contained Sections 10
42132:
42133: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maxXbox4.exe
42134: Verifying: maxXbox4.exe
42135: Signature Index: 0 (Primary Signature)
42136: Hash of file (sha1): 5E556D92930A5ED88CBDAE8E5A174EE8D3161F56
42137:
42138: Signing Certificate Chain:
42139: Issued to: maxXboxCertAuth
42140: Issued by: maxXboxCertAuth
42141: Expires: Sun Jan 01 01:59:59 2040
42142: SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
42143:
42144: Issued to: maxXbox4signer
42145: Issued by: maxXboxCertAuth
42146: Expires: Sun Jan 01 01:59:59 2040
42147: SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
42148:
42149: Issued to: maxXbox4exe
42150: Issued by: maxXbox4signer
42151: Expires: Sun Jan 01 01:59:59 2040
42152: SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
42153:
42154: The signature is timestamped: Mon Oct 17 11:48:48 2022
42155: Timestamp Verified by:
42156: Issued to: USERTrust RSA Certification Authority
42157: Issued by: USERTrust RSA Certification Authority
42158: Expires: Tue Jan 19 01:59:59 2038
42159: SHA1 hash: 2B8F1B57330DBBA2D07A6C51F70EE90DDAB9AD8E

```

```

42160:      Issued to: Sectigo RSA Time Stamping CA
42161:      Issued by: USERTrust RSA Certification Authority
42162:      Expires:  Tue Jan 19 01:59:59 2038
42163:      SHA1 hash: 02D65B95E28370C1570095FA88F923DD937FAD8F
42164:      Issued to: Sectigo RSA Time Stamping Signer #3
42165:      Issued by: Sectigo RSA Time Stamping CA
42166:      Expires:  Thu Aug 11 01:59:59 2033
42167:      SHA1 hash: AB34013AAC4097319F081AF0B318E183F80F7881
42168:
42169: Successfully verified: maxbox4.exe
42170:
42171: Number of files successfully Verified: 1
42172: Number of warnings: 0
42173: Number of errors: 0
42174:
42175: C:\maxbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
42176:
42177: MD5 863f4a600090421779a0d4855f31593c
42178: SHA-1 b2b44fc20da22111ef81553b94b0b903b9d7fa7a
42179: SHA-256 66e02bfc560c5082c2bfc5e4b8e2a7b19fa1f60386474097625d55c7cc6b472
42180: Vhash 0370a6666d5c0d555c051031e0900e5003116012z33034z17013109054843115z10
42181: Authntihash 4999cd522416fd5fc367b54d2d00b253177a463c174fea9ece121a50e639e16
42182: Imphash da442fa7c70f4ee112e5300fldac062
42183: SSDEEP 393216:1WhOYzrD7vZw7VJlT8pGxysz1Pg8DeDAmcAR9+XR7R/j4ENWDQXZd5eGlJFUVTKv:0OYzrD7cvNbzmcs9I9Rn5PdGfod
42184: TLSH T113778C62B1C0D632D05705788C0FD6E85A6A3D206DB4949B79F6BF8D2E311A17F362CB
42185: File type Win32 EXE
42186: Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
42187: 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%)
42188: DetectItEasy PE32 Compiler: Borland Delphi (2006) Linker: Turbo Linker (2.25*,Delphi) [GUI32,signed]
42189: File size 32.44 MB (34014488 bytes)
42190:
42191: Verifying: maxbox4.exe
42192: Signature Index: 0 (Primary Signature)
42193: Hash of file (sha1): 221468297F0D9C645950C99EFB8F92C12E1C9A39
42194: Signing Certificate Chain:
42195:      Issued to: maxboxCertAuth
42196:      Issued by: maxboxCertAuth
42197:      Expires:  Sun Jan 01 00:59:59 2040
42198:      SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
42199:      Issued to: maxbox4signer
42200:      Issued by: maxboxCertAuth
42201:      Expires:  Sun Jan 01 00:59:59 2040
42202:      SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
42203:      Issued to: maxbox4exe
42204:      Issued by: maxbox4signer
42205:      Expires:  Sun Jan 01 00:59:59 2040
42206:      SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
42207:
42208: The signature is timestamped: Thu Dec 08 11:12:55 2022
42209: Timestamp Verified by:
42210:      Issued to: USERTrust RSA Certification Authority
42211:      Issued by: USERTrust RSA Certification Authority
42212:      Expires:  Tue Jan 19 00:59:59 2038
42213:      SHA1 hash: 2B8F1B57330DBBA2D07A6C51F70EE90DDAB9AD8E
42214:      Issued to: Sectigo RSA Time Stamping CA
42215:      Issued by: USERTrust RSA Certification Authority
42216:      Expires:  Tue Jan 19 00:59:59 2038
42217:      SHA1 hash: 02D65B95E28370C1570095FA88F923DD937FAD8F
42218:      Issued to: Sectigo RSA Time Stamping Signer #3
42219:      Issued by: Sectigo RSA Time Stamping CA
42220:      Expires:  Thu Aug 11 00:59:59 2033
42221:      SHA1 hash: AB34013AAC4097319F081AF0B318E183F80F7881
42222: Successfully verified: maxbox4.exe
42223: Number of files successfully Verified: 1
42224: Number of warnings: 0
42225: Number of errors: 0
42226: C:\maxbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
42227:
42228: Borland Conf 1999 Video: https://www.youtube.com/watch?v=mlGn4echQIE
42229: http://www.softwareschule.ch/examples/classifier\_compare2confusion2.py.htm
42230: 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.
42231: https://www.udemy.com/course/learn-coding-from-the-scratch/learn/
42232: - Valuable information-Clear explanations Engaging delivery-Helpful practice activities
42233: - Accurate course description - Knowledgeable instructor
42234:
42235: This Version is dedicated to Katherine Goble Johnson, NASA mathematician
42236: --- The girl who asked questions ---
42237:
42238: https://github.com/joaopauloschuler/neural-api
42239: https://sourceforge.net/projects/cai/
42240: http://www.formatio-reticularis.de/code.html
42241: http://cyberunits.sourceforge.net/
42242: https://torry.net/files/tools/developers/scripts/maxbox4.zip
42243: https://softwareschule.code.blog/
42244: https://maxbox4.wordpress.com/

```

42245: <https://my6.code.blog/2020/01/17/maxbox4machine/>

42246: <https://www.filecroco.com/download-maxbox/>

42247: <https://towardsdatascience.com/how-to-implement-machine-learning-for-predictive-maintenance-4633cdbe4860>

42248: <https://www.udemy.com/course/learn-coding-from-the-scratch/learn/>

42249: <https://data.world/maxbox>

42250: <https://data.world/maxbox/maxbox>

42251: <https://maxbox4.files.wordpress.com/2020/08/twinwizard-copy.jpeg?w=688&h=&zoom=2>

42252: https://maxbox4.files.wordpress.com/2020/07/ice4_interlaken_max.jpg?w=768&h=&zoom=2

42253: <https://github.com/maxkleiner/maXbox4/blob/master/objectdetector3.ipynb>

42254: <https://colab.research.google.com/github/maxkleiner/maXbox4/blob/master/objectdetector3.ipynb>

42255: <https://my6.code.blog/>

42256: <https://entwickler-konferenz.de/blog/machine-learning-mit-cai/>

42257: <https://kiosk.entwickler.de/windows-developer-magazin/windows-developer-magazin-10-20/erklaerbare-modelle-schaffen-vertrauen/>

42258: <https://entwickler-konferenz.de/blog/machine-learning-mit-cai/>

42259: <https://repl.it/@MaxKleiner/machinelearning4#main.py>

42260: <https://maxbox4.wordpress.com/2020/05/15/ekon-24/>

42261: https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb

42262: <https://github.com/maxkleiner/maXbox4/blob/master/sentimenttree2.ipynb>

42263: <https://towardsdatascience.com/6-amateur-mistakes-ive-made-working-with-train-test-splits-916fabb421bb>

42264: <https://repl.it/@maxbox/machinelearning4#main.py>

42265: <https://maxbox.sourceforge.io/>

42266: <https://linuxschweizag.wordpress.com/>

42267: <https://maxkleiner1.medium.com/>

42268: <https://ideone.com/hQ3zAd>

42269: <https://appdb.winehq.org/objectManager.php?sClass=version&iId=39678>

42270: <https://bitbucket.org/maxbox4/maxbox4/downloads/>

42271: <https://entwickler-konferenz.de/blog/machine-learning-mit-cai/>

42272: <https://github.com/maxkleiner/python4delphi/blob/master/Source/PythonEngine.pas>

42273: <https://maxbox4.blogspot.com/>

42274: <https://wiki.freepascal.org/Python4Delphi>

42275: <https://www.clevercomponents.com/articles/article052/>

42276: <https://entwickler-konferenz.de/delphi-innovations-fundamentals/vcl4python/>

42277:

42278: 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>.

42279: Benefits: 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

42280:

42281: 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.

42282: Association Rules uncover the relationship between two or more attributes. It is mainly in the form of- If antecedent than consequent.

42283: Picture Exhibition 2021

42284: <https://my6.code.blog/2021/11/12/duesseldorf-ekon-25/>

42285: <https://github.com/maxkleiner/neural-api>

42286: <https://my6.code.blog/2021/09/08/improver-4/>

42287: <https://my6.code.blog/2020/11/09/grauzone/>

42288: <https://github.com/maxkleiner/DelphiVCL4Python>

42289: <https://www.scribd.com/document/553749096/Blaise-101-UK-6P-Max-Kleiner-V-Nr91-Tutor-Synthetic-Data-SynDat>

42290: <https://opensea.io/maxbox4>

42291: <https://blogs.embarcadero.com/fun-maxbox-is-an-all-in-one-script-engine-application-powered-by-delphi/>

42292: https://www.academia.edu/37015210/Machine_Learning_maXbox_starter60_1

42293: <https://github.com/maxkleiner/agsi-data>

42294: <https://www.hybrid-analysis.com/sample/c6e8c0828c61b0a05cb26ebdaf6985983eb8f6b4d92fb99ad2fa62d2ba50bed>

42295:

42296: Data Scientist also **requires** strong technical skills in:

42297: 1. Math (e.g. linear algebra, calculus and probability).

42298: 2. Statistics (e.g. hypothesis testing and summary statistics).

42299: 3. Machine learning tools and techniques (e.g. k-nearest neighbours, random forests, ensemble methods, svm).

42300: 4. Software engineering skills (e.g. distributed computing, algorithms and data structures).

42301: 5. Data mining.

42302: 6. Data cleaning and merging.

42303: 7. Data visualization (e.g. plottsystem for R ggplot2, py and JavaScript library for manipulating documents based on data d3.js) and reporting techniques.

42304: 8. Unstructured data techniques.

42305: 9. R and/or SAS languages.

42306: 10. SQL databases and database querying languages.

42307: 11. Python (most common), C/C++ Java, Perl.

42308: 12. Big data platforms like Hadoop, Hive & Pig.

42309: 13. Cloud tools like Amazon S3.

42310: 14. Pascal Python in maXbox4

42311: 15. TotInf in maXbox4

42312:

42313: Data Analyst

42314: 1. Work with IT teams, management and/or data scientists to determine organizational goals.

42315: 2. Mine data from primary and secondary sources.

42316: 3. Clean and prune data to discard irrelevant information.

```

42317: 4. Analyze and interpret results using standard statistical tools and techniques.
42318: 5. Pinpoint trends, correlations and patterns in complicated data sets.
42319: 6. Identify new opportunities for process improvement.
42320: 7. Provide concise data reports and clear data visualizations for management.
42321: 8. Design, create and maintain relational databases and data systems.
42322: 9. Triage code problems and data-related issues.
42323: 10. check always the semantic consistency (does data make valid sense?)
42324:
42325: 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.
42326: Suppose you have a python script file httpheader1.py:
42327: "G:\maXbox\works2022\maXbox4\examples\httpheader1.py"
42328:
42329: Doc :
42330: AssignFile versus TStringlist
42331: -----
42332: 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.
42333: 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
42334:
42335:
42336:
42337:
42338:
42339:
42340:
42341:
42342:
42343: Sort of a microservice you find at:
42344: https://www.academia.edu/31112544/Work_with_microservice_maXbox_starter48.pdf
42345:
42346:
42347:
42348:
42349:
42350:
42351:
42352:
42353:
42354:
42355:
42356:
42357:
42358:
42359:
42360:
42361:
42362:
42363:
42364:
42365:
42366:
42367:
42368:
42369:
42370:
42371:
42372:
42373:
42374:
42375:
42376:
42377: 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.
42378:
42379: Hybrid Analysis <noreply@hybrid-analysis.com>
42380:
42381: 7:27 PM (1 hour ago)
42382:
42383: to
42384: Congratulations! Your analysis is done and available at: https://www.hybrid-
analysis.com/sample/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3?environmentId=160
42385:
42386: --- Falcon Sandbox Analysis Overview ---
42387:
42388: https://www.hybrid-analysis.com/sample/102653b74b1efb16af424c243130e3c0395f9875d0e8900e688dd42a25d218a3
42389:
42390: --- Falcon Sandbox Analysis Summary ---
42391:
42392: File Name: maXbox4.exe
42393: Analysis State: SUCCESS
42394: Threat Verdict: no specific threat
42395: Threat Score: n/a/100
42396: AV Detection Ratio: n/a

```

```

42397: AV Family Name:
42398: Time of analysis: 2022-12-15 18:16:53
42399: File Size (bytes): 34057496
42400: File Type: PE32 executable (GUI) Intel 80386, for MS Windows
42401: Contacted Domains: none
42402: Contacted Hosts: none
42403: Environment: Windows 10 64 bit (ID: 160)
42404:
42405: Submission name: maXbox4.exe
42406: Size: 33MiB
42407: Type: peexe executable
42408: Mime:application/x-dosexec
42409: SHA256: 97943841d5908ea0846dc6763b14941350a0c1aa5d2b6248bfc79ffad7a0314e Copy SHA256 to clipboard
42410: Operating System: Windows
42411: Last Anti-Virus Scan:
42412: 01/25/2023 08:08:02 (UTC)
42413: Last Sandbox Report:
42414: 01/25/2023 08:08:00 (UTC)
42415:
42416:
42417:
42418:
42419:
42420:
42421:
42422:
42423:
42424:
42425:
42426:
42427:
42428:
42429:
42430:
42431:
42432:
42433:
42434:
42435:
42436:
42437:
42438:
42439: A microservice architecture shifts around complexity. Instead of a single complex system, you have a
42440: bunch of simple services with complex interactions.
42441:
42442: Simplify API development for users, teams, and enterprises with the Swagger open source and
42443: professional toolset. Find out how Swagger can help you design, document your APIs at scale.
42444:
42445:
42446:
42447:
42448:
42449:
42450:
42451:
42452:
42453:
42454:
42455:
42456:
42457:
42458:
42459:
42460:
42461:
42462:
42463:
42464:
42465:
42466:
42467:
42468:
42469: This is what we want to use in maXbox:
42470:
42471: const res = await fetch("https://libretranslate.com/translate", {
42472:   method: "POST",
42473:   body: JSON.stringify({
42474:     q: "Hello!",
42475:     source: "en",
42476:     target: "es"
42477:   }),
42478:   headers: { "Content-Type": "application/json" }
42479: });
42480: console.log(await res.json());
42481:
42482: Feature update to Windows 10, version 22H2
42483: Target Machine Intel 386 or later processors and compatible processors

```



```

42484: SHA1: 4.7.6.20 F60338A77B77F2032061BF72A545AFB727F6395F
42485: f60338a77b77f2032061bf72a545afb727f6395f
42486: CRC32: 48455EF8 32.8 MB (34,419,992 bytes)
42487: Compilation Timestamp 2023-01-26 15:36:15 UTC Signing time 26 Jan 2023 16:41:42
42488: Entry Point 25033256 - Contained Sections 10
42489: sha1: f60338a77b77f2032061bf72a545afb727f6395f
42490: sha256: 9347258d5985a2fe88aldf45c3cd99747babbba034de3e1725139684a7b6e1c7
42491:
42492: https://maxbox4.wordpress.com/2022/11/17/data-science-story/
42493:
42494: Teach and Train it with the Jupyter Notebook:
42495: Tested on
42496: Delphi 10.4 Community Edition Update 2
42497: Lazarus V 2.2.4 for Windows 64 bit
42498: maxXbox4 Version 4.7.6.20
42499: PyScripter 4.1.1.0 x64
42500: SVGImage32Package270.bpl V 3.9.4
42501: jupyter notebook v7.0.0a11
42502:
42503: MD5 3a228997326e3ab981e047e406ee0a77
42504: SHA-1 f60338a77b77f2032061bf72a545afb727f6395f
42505: SHA-256 9347258d5985a2fe88aldf45c3cd99747babbba034de3e1725139684a7b6e1c7
42506: Vhash 0370a6666d5c0d555c051031e0900e5003116012z33034z17883109054843115z10
42507: AuthentiHash 344a8eeb7669f994df01e8ba7af69e8b4a3827b782c4733506002905656ec483
42508: Imphash 1cdc6c1287a2d8f5bfb5ac0d1f3fc6f6
42509: SSDEEP 393216:6clUvGNCN3jg98pGx8RzxU0+jBLe9xcuMjJoy3edsNzViWVD52WB/CefyTeFcncH:AGNCN4F9AgjidOvNmcimfoc
42510: TlSH T134778C62B1C0D632D05705788C0BD6D85A6A3D207DB4949B79F6BF8E2E31161BF362CB
42511: File type Win32 EXE
42512: Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
42513: 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%)
42514: DetectItEasy PE32 Compiler: Borland Delphi (2006) Linker: Turbo Linker (2.25*,Delphi) [GUI32,signed]
42515: File size 32.83 MB (34419992 bytes)
42516:
42517: https://github.com/maxkleiner/Bayes_theorem/blob/master/clusterclass4penguins.ipynb
42518: https://github.com/maxkleiner/maxXbox4/blob/master/clusterclass4penguins.ipynb
42519: https://github.com/maxkleiner/maxXbox4/releases/download/V4.2.4.80/maxXbox4.exe
42520: https://my6.code.blog/2022/09/02/webpostdata/
42521: https://maxbox4.wordpress.com/2022/01/31/tutorials-overview-2014-2022/
42522: https://blogs.embarcadero.com/why-a-data-scientist-chooses-delphi-for-powerful-real-world-visualizations/
42523: https://youtu.be/k1-bv-74Rf8
42524: https://maxbox4.wordpress.com/2023/05/23/mapbox-in-maxbox/
42525: https://linuxschweizag.wordpress.com/2023/04/06/tutorials/
42526: https://en.wikipedia.org/wiki/Anathema_(band)
42527: https://github.com/project-jedi/jvcl/blob/master/jvcl/devtools/JvclVclClx/Utils.pas
42528: https://github.com/gabr42/GpDelphiUnits/tree/master/src
42529: https://github.com/TheUnknownOnes/theunknownones/blob/master/Libraries/CommonTools/uGraphicalTools.pas
42530: https://github.com/coderserdar/DelphiComponents/blob/main/Other/API%20Pack/API_source/API_strings.pas
42531: https://github.com/graemeg/dunit2/blob/master/src/TestUtils.pas
42532:
42533: Songs for maxXbox
42534: 1. Beat the Blues /2. Play Chess /3. Cary Copper (White magic) /4. Eat Garlic (NowAge)
42535: 5. Exerce de Meditation /6. Practise running (Japan) /7. Bett von Nord nach Süd - Zeitraum
42536: 8. Olemos Bruja /9. Listen white noise (Lord of Darkness)
42537:
42538: https://sourceforge.net/projects/maxbox/files/maxXbox_Archive/Improver/Improver_4_Edition_2021.zip/download
42539:
42540: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>signtool verify /v /pa maxXbox4.exe
42541:
42542: V4.7.6.50 V
42543: Verifying: maxXbox4.exe
42544: Signature Index: 0 (Primary Signature)
42545: Hash of file (sha1): FD55A476E52E8A13C41DD3AF5F0A1115DC390EB0
42546:
42547: Signing Certificate Chain:
42548: Issued to: maxXboxCertAuth
42549: Issued by: maxXboxCertAuth
42550: Expires: Sun Jan 01 01:59:59 2040
42551: SHA1 hash: 6F83207B500DCC0E32A719599CBC6BD7E6B2A04D
42552:
42553: Issued to: maxXbox4signer
42554: Issued by: maxXboxCertAuth
42555: Expires: Sun Jan 01 01:59:59 2040
42556: SHA1 hash: 6A89501B76D47C189A60BF1070BAA2FBFD38D7D7
42557:
42558: Issued to: maxXbox4exe
42559: Issued by: maxXbox4signer
42560: Expires: Sun Jan 01 01:59:59 2040
42561: SHA1 hash: F0EB0CA218C5707FAC78921F81092CECA12AD0E9
42562:
42563: The signature is timestamped: Thu Jun 15 08:42:33 2023
42564: Timestamp Verified by:
42565: Issued to: USERTrust RSA Certification Authority
42566: Issued by: USERTrust RSA Certification Authority
42567: Expires: Tue Jan 19 01:59:59 2038
42568: SHA1 hash: 2B8F1B57330DBBA2D07A6C51F70EE90DDAB9AD8E
42569:
42570: Issued to: Sectigo RSA Time Stamping CA
42571: Issued by: USERTrust RSA Certification Authority

```

```

42572: Expires: Tue Jan 19 01:59:59 2038
42573: SHA1 hash: 02D65B95E28370C1570095FA88F923DD937FAD8F
42574:
42575: Issued to: Sectigo RSA Time Stamping Signer #4
42576: Issued by: Sectigo RSA Time Stamping CA
42577: Expires: Thu Aug 03 01:59:59 2034
42578: SHA1 hash: AE62AF750A0CBD47D6461F7568E2BC8CE7CA4F94
42579:
42580: Successfully verified: maxXbox4.exe
42581:
42582: Number of files successfully Verified: 1
42583: Number of warnings: 0
42584: Number of errors: 0
42585:
42586: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
42587: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
42588: C:\maxXbox\EKON_BASTA\EKON19\Windows Kits\10\bin\x64>
42589:
42590:
42591: -----
42592: -----
42593: -----
42594: -----
42595: -----
42596: -----
42597: -----
42598: -----
42599: -----
42600: -----
42601: -----
42602: -----
42603: -----
42604: -----
42605: -----
42606: -----
42607: -----
42608: https://maxbox4.wordpress.com/blog/
42609:
42610: Total of Function Calls: 36852
42611: SHA1: 4.7.6.50 D4FD4CACFD766EB8F78F2BB7B5EFDDEBB386597A
42612: d4fd4cacfd766eb8f78f2bb7b5efddebb386597a
42613: CRC32: C1DCD693 33.50 MB (35,128,600 bytes)
42614: Compilation Timestamp 2023-05-31 12:44:17 UTC Signtime 31 May 2023 14:48:34
42615: Entry Point 25234104 - Contained Sections 10
42616: sha1: d4fd4cacfd766eb8f78f2bb7b5efddebb386597a
42617: sha256: 4dfbada6765e47c72b7c2496f831419b3461fa7c4ac6e05e2a941b501e10e022
42618:
42619: Total of Function Calls: 36449
42620: SHA1: 4.7.6.50 C56F0A26DFBE706E6A1A9D0E23B6114AFCC09CDC
42621: c56f0a26dfbe706e6a1a9d0e23b6114afcc09cdc
42622: CRC32: 29BCAE4D 33.12 MB (34,724,120 bytes)
42623: Compilation Timestamp 2023-05-21 13:42:16 UTC Signtime 21 May 2023 15:45:22
42624: Entry Point 25102976 - Contained Sections 10
42625: sha1: c56f0a26dfbe706e6a1a9d0e23b6114afcc09cdc
42626: sha256: ca5b41a709e61c1174d5105d1edc2759e01bd3c4a57a30b54dad568e104cac77
42627:
42628: Total of Function Calls: 36781
42629: SHA1: 4.7.6.50 FBC9E7794EEF888DCBE94350D8F53A8FD05BD51F
42630: fbc9e7794eef888dcbe94350d8f53a8fd05bd51f
42631: CRC32: 94780E06 33.37 MB (34,993,432 bytes)
42632: Compilation Timestamp 2023-05-26 07:25:55 UTC Signtime 26 May 2023 09:30:37
42633: Entry Point 25209520 - Contained Sections 10
42634: sha1: fbc9e7794eef888dcbe94350d8f53a8fd05bd51f
42635: sha256: c9f275808708fa8bbe56f816a6ae90f2438dd8a7085ecf8e4bbcff1e1747571
42636: https://sourceforge.net/projects/maxbox/files/Examples/13_General/
42637: Total of Function Calls: 36798
42638: SHA1: 4.7.6.50 CAA09249B338ED2814B760FA549DAD47F6C789D2
42639: caa09249b338ed2814b760fa549dad47f6c789d2
42640: CRC32: 17A61798 33.41 MB (35,033,880)
42641: Compilation Timestamp 2023-05-30 13:17:50 UTC Signtime 30 May 2023 15:20:39
42642: Entry Point 25217720 - Contained Sections 10
42643: http://www.softwareschule.ch/maxbox_functions.txt
42644: sha1: caa09249b338ed2814b760fa549dad47f6c789d2
42645: sha256: 222ee48c3409cbf176ffe9037d7958952e6754549c66d8397e927ad68ee3be17
42646:
42647: Total of Function Calls: 36834
42648: SHA1: 4.7.6.50 EDE730F514834A85ECA6D0190DAC8CA6046C26EE
42649: ede730f514834a85eca6d0190dac8ca6046c26ee
42650: CRC32: 4E7C4A2B 33.49 MB (35,117,336 bytes)
42651: Compilation Timestamp 2023-05-31 08:25:23 UTC Signtime 31 May 2023 10:33:00
42652: Entry Point 25225912 - Contained Sections 10
42653: sha1: ede730f514834a85eca6d0190dac8ca6046c26ee
42654: sha256: 168f85cff6ecdcc4e7614758bbdb333a38f4896f94b7056f3e53af8fd15a66b
42655:
42656: Total of Function Calls: 36852
42657: SHA1: 4.7.6.50 D4FD4CACFD766EB8F78F2BB7B5EFDDEBB386597A
42658: d4fd4cacfd766eb8f78f2bb7b5efddebb386597a
42659: CRC32: C1DCD693 33.50 MB (35,128,600 bytes)
42660: Compilation Timestamp 2023-05-31 12:44:17 UTC Signtime 31 May 2023 14:48:34

```

```
42661: Entry Point 25234104 - Contained Sections 10
42662: sha1: d4fd4cacfd766eb8f78f2bb7b5efddebb386597a
42663: sha256: 4dfbada6765e47c72b7c2496f831419b3461fa7c4ac6e05e2a941b501e10e022
42664: Total of Function Calls: 36947
42665: SHA1: 4.7.6.50 C430FDBA9880317E31D4DA2F66C884799298FCC3
42666: c430fdb9880317e31d4da2f66c884799298fcc3
42667: CRC32: 5BD9839C 33.62 MB (35,257,624 bytes)
42668: Compilation Timestamp 2023-06-06 14:16:08 UTC Signtime 06 June 2023 16:22:14
42669: Entry Point 25287360 - Contained Sections 10
42670: sha1: c430fdb9880317e31d4da2f66c884799298fcc3
42671: sha256: 1658eb368e4e8621c51d8e568de40a12be00ad02c63dfe7a3b655603661dc4f9
42672:
42673: V4.7.6.50 IV
42674: Amount of Functions: 22064
42675: Amount of Procedures: 12992
42676: Amount of Constructors: 2051
42677: Amount of Destructors: 14
42678: Totals of Calls: 37121
42679: SHA1: of 4.7.6.50 9E9D9D10762AE0D0BC8AFD747C2381EA5478AE49
42680: CRC32: 01B431EE: 35558168 bytes
42681: mX4 executed: 12/06/2023 15:52:43 Runtime: 0:16:6.980 Memload: 41% use
42682:
42683:
42684: Total of Function Calls: 37121
42685: SHA1: 4.7.6.50 9E9D9D10762AE0D0BC8AFD747C2381EA5478AE49
42686: 9e9d9d10762ae0d0bc8afd747c2381ea5478ae49
42687: CRC32: 01B431EE 33.91 MB (35,558,168 bytes)
42688: Compilation Timestamp 2023-06-12 13:17:28 UTC Signtime 12 June 2023 15:20:10
42689: Entry Point 25377544 - Contained Sections 10
42690: sha1: 9e9d9d10762ae0d0bc8afd747c2381ea5478ae49
42691: sha256: fa0f30abf34292e91070a5bd4682040eb6af79a8f6c7f55111c9692153120988
42692:
42693: V4.7.6.50 V
42694: Total of Function Calls: 37372
42695: SHA1: 4.7.6.50 D047DBD5412C3E4A436089018B9C7FACF17A2EB5
42696: d047dbd5412c3e4a436089018b9c7facf17a2eb5
42697: CRC32: 38562FA8 34.04 MB (35,697,944 bytes)
42698: Compilation Timestamp 2023-06-15 06:40:19 UTC Signtime 15 June 2023 08:42:33
42699: Entry Point 25484072 - Contained Sections 10
42700: sha1: d047dbd5412c3e4a436089018b9c7facf17a2eb5
42701: sha256: 193679043c46821c4b2460269111ff112e7c67b49e9cd9d951854e61bceedac7
42702:
42703: maXbox4.exe 4.7.6.50
42704:
42705: This report is generated from a file or URL submitted to this webservice on June 12th 2023 14:22:33 (UTC)
42706: Guest System: Windows 7 64 bit, Professional, 6.1 (build 7601), Service Pack 1
42707: Report generated by Falcon Sandbox v10.1.5 © Hybrid Analysis
42708:
42709: Classification (TrID)
42710:
42711: 60.1% (.OCX) Windows ActiveX control
42712: 22.2% (.EXE) InstallShield setup
42713: 5.4% (.EXE) Win64 Executable (generic)
42714: 5.1% (.EXE) DOS Borland compiled Executable (generic)
42715: 2.3% (.EXE) Win32 Executable (generic)
42716:
42717: https://www.hybrid-analysis.com/sample/fa0f30abf34292e91070a5bd4682040eb6af79a8f6c7f55111c9692153120988
42718: https://www.hybrid-analysis.com/sample/882ddadb9b330318bfff73d8db66d1a33be575be4cedec3960374bfeb7c49c968
42719: https://linuxschweizag.wordpress.com/2023/04/06/tutorials/
42720: https://maxbox4.wordpress.com/2023/05/23/mapbox-in-maxbox/
42721:
```