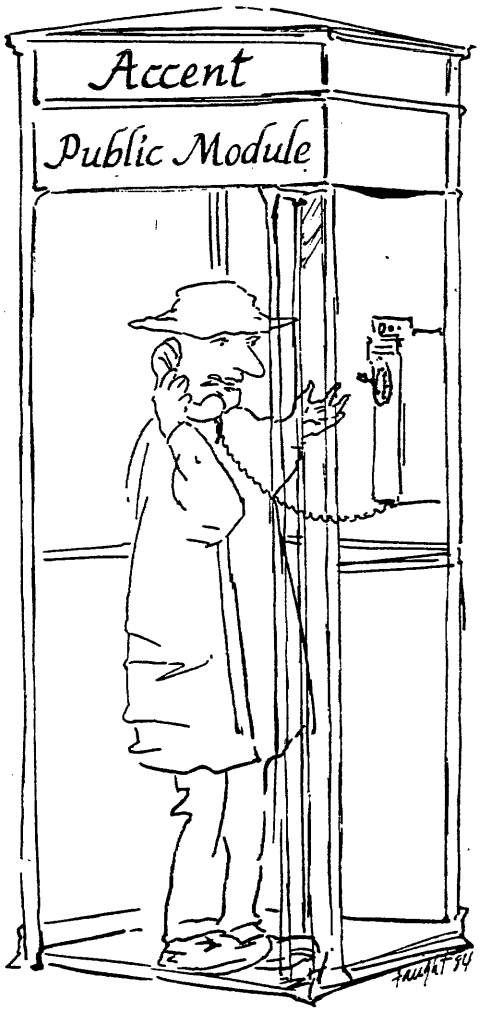
CARNEGIE-MELLON UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SPICE PROJECT

Accent Public Module Index

Spice Project



29 June 1984

Supported by the Defense Advanced Research Projects Agency, Department of Defense, ARPA Order 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Projects Agency or the U.S. Government.

Accent Public Module Index

29 June 1984

Modules:

Module:	*File:*
AccCall	acccall.pas
AccentType	accenttype.pas
AccInt	accentuser.pas
ALoad	aload.pas
Auth	authuser.pas
AuthDefs	authdefs.pas
BootInfo	bootinfo.pas
CFileDefs	cfiledefs.pas
Cload	cload.pas
Clock	clock.pas
Code	code.pas
CommandDefs	commanddefs.pa
CommandParse	commandparse.p
Configuration	configuration.
Dynamic	dynamic.pas
EnvMgr	envmgruser.pas
EnvMgrDefs	envmgrdefs.pas
Ether10Defs	ether10defs.pa
EtherTypes	ethertypes.pas
EtherUser	etheruser.pas
Except	except.pas
ExtraCmdParse	extracmdparse.
IO	iouser.pas
IODefs	iodefs.pas
IPCRecordIO	ipcrecordio.pa
ModGetEvent	modgetevent.pa
MsgN	msgnuser.pas
NameErrors	nameerrors.pas
Net10MB	net10mbuser.pa
Net10MBRecvServer	net10mbrecvser
OldTimeStamp	oldtimestamp.p
PascalInit	pascalinit.pas
PasLong	paslong.pas
PasReal	pasreal.pas
PathName	pathname.pas
PMatch	pmatch.pas
ProcMgr	procmgruser.pa
ProcMgrDefs	procmgrdefs.pa
QMapDefs	qmapdefs.pas
Reader	reader.pas
RealFunctions	realfunctions.
RunDefs	rundefs.pas
SaltError	salterror.pas

SaphEmrExceptions	saphemrexcepti
SaphEmrServer	saphemrserver.
Sapph	sapphuser.pas
SapphDefs	sapphdefs.pas
SapphFileDefs	sapphfiledefs.
SegDefs	segdefs.pas
Sesame	sesameuser.pas
SesameDefs	sesamedefs.pas
SesDisk	sesdiskuser.pa
SesDiskDefs	sesdiskdefs.pa
Spawn	spawn.pas
SpawnInitFlags	spawninitflags
Spice_String	spice_string.p
Stream	stream.pas
SymDefs	symdefs.pas
SysType	systype.pas
Time	timeuser.pas
TimeDefs	timedefs.pas
TS	tsuser.pas
TSDefs	tsdefs.pas
ViewKern	viewkern.pas
ViewPt	viewptuser.pas
WindowUtils	windowutils.pa
Writer	writer.pas

Definitions:

Abort [*Except*] exception(Message: String)
AbsoluteDef [*CFileDefs*] const = 13
AbsoluteLocalDef [*CFileDefs*] const = 14
Accent_Version [*AccInt*] function(ServPort: port; AccVersion: DevPartString): GeneralReturn
AccentError [*Dynamic*] exception(R: GeneralReturn; M: String)
AccentVersion [*Dynamic*] const = True
AccErr [*AccentType*] const = 100
Access_Rights [*SesDiskDefs*] type = integer
AddExtension [*PathName*] procedure(var fileName: Path_Name; Extension: String)
AddSearchWord [*CommandParse*] procedure(table: pWord_Search_Table; WordKey: integer; WordString: Cmnd_String)
Adjust [*Spice_String*] procedure(var Str: PString; Len: Integer)
All_Read_Access [*SesDiskDefs*] const = #100
All_Write_Access [*SesDiskDefs*] const = #200
AllocatePort [*AccInt*] function(ServPort: port; var LocalPort: Port; BackLog: Integer): GeneralReturn
AllocCommandNode [*CommandParse*] function(WordClass: Word_Type; WordString: Cmnd_String): pCommand_Word_List
ALLPORTS [*AccentType*] const = -1
ALLPTS [*AccentType*] const = 1
ALoadError [*ALoad*] exception(s: string_255)
AlwaysEof [*CommandParse*] procedure(var ChPool: pCharacter_Pool; var PoolLength: Char_Pool_Index)
APath_Name [*SesameDefs*] type = string[Path_Name_Size]
AppendChar [*Spice_String*] procedure(var Str: PString; c: Char)
AppendString [*Spice_String*] procedure(var Str1: PString; Str2: PString)

ArcCos [*RealFunctions*] function(X: Real): Real
ArcCosLarge [*RealFunctions*] exception(X: Real)
ArcSin [*RealFunctions*] function(X: Real): Real
ArcSinLarge [*RealFunctions*] exception(X: Real)
ArcTan [*RealFunctions*] function(X: Real): Real
ArcTan2 [*RealFunctions*] function(Y, X: Real): Real
ArcTan2Zero [*RealFunctions*] exception(Y, X: Real).
Arguments [*AccentType*] type = record ReturnValue: GeneralReturn; case integer of 1: (Msg: ptrMsg; MaxWait: long; Option: integer; PtOption: integer); 2: (Ports: ptrPortBitArray; MsgType: long); 3: (SrcAddr: VirtualAddress; DstAddr: VirtualAddress; NumWords: long; Delete: boolean; Create: boolean; Mask: long; DontShare: boolean); 4: (NumCmds: integer; GPIBBuffer: GPBuffer); 5: (NormalOrEmergency: boolean; EnableOrDisable: boolean); 6: (SleepID: long); 7: (RectPort: Port; X1: integer; Y1: integer; X2: integer; Y2: integer; Kind: integer); 8: (SrcRect: Port; DstRect: Port; Action: integer; Height: integer; Width: integer; SrcX: integer; SrcY: integer; DstX: integer; DstY: integer); 9: (Rect: Port; FontRect: Port; Funct: integer; FirstX: integer; FirstY: integer; MaxX: integer; FirstChar: integer; MaxChar: integer; StrPtr: Pointer; Rslt: integer); 10: (LockDoLock: boolean; LockPortPtr: ptrLPortArray; LockPortCnt: long); 11: (MsgWType: long; MsgWPortPtr: ptrLPortArray; MsgWPortCnt: long) end
ARunLoad [*ALoad*] procedure(RunFileName: Path_Name; p: pointer; filesize: long; hiskport: port; LoadDebug: boolean)
ASKUSER [*SapphDefs*] const = -32005
ASTInconsistency [*AccentType*] const = AccErr + 46
ASTRecord [*BootInfo*] type = integer
ASyncIO [*IO*] procedure(ServPort: ServerIOPort; Command: IOCommand; CmdBlk: Pointer; CmdBlk_Cnt: long; DataBuf: Pointer; DataBuf_Cnt: long; DataTransferCnt: Long; TimeOut: Long)
Auth_Error_Base [*AuthDefs*] const = 5000
Auth_Var [*AuthDefs*] type = String[Auth_Var_Size]
Auth_Var_Size [*AuthDefs*] const = 30
AuthorizationServerGone [*SesDiskDefs*] const = SesDisk_Error_Base + 3
AuthPortIncorrect [*AuthDefs*] const = Auth_Error_Base + 3
AvailableVM [*AccInt*] function(ServPort: port; var NumBytes: Long): GeneralReturn
BackLogValue [*AccentType*] type = 0..MAXBACKLOG
BadAlignment [*Dynamic*] exception
BadBase [*Stream*] exception(FileName: SName; Base: Integer)
BadCreateMask [*AccentType*] const = AccErr + 52
BadDateTime [*Time*] exception
BadExit [*Except*] exception
BadHeap [*Dynamic*] exception(H: HeapNumber)
BadHeapNumber [*Dynamic*] const = 0
BadIdTable [*Stream*] exception(FileName: SName)
BadIPCName [*AccentType*] const = AccErr + 19
BadKernelMsg [*AccentType*] const = AccErr + 27
BadMsg [*AccentType*] const = AccErr + 22
BADMSGID [*AccentType*] const = 1
BadMsgType [*AccentType*] const = AccErr + 18
BadName [*SesameDefs*] const = Sesame_Error_Base + 4
BadPartitionName [*SesDiskDefs*] const = SesDisk_Error_Base + 1
BadPartitionType [*SesDiskDefs*] const = SesDisk_Error_Base + 2
BadPatterns [*PMatch*] exception
BadPointer [*Dynamic*] exception
BadPriority [*AccentType*] const = AccErr + 30
BadRectangle [*AccentType*] const = AccErr + 53
BADREPLY [*AccentType*] const = 3
BadRights [*AccentType*] const = AccErr + 8

BadSearchlistSyntax [*EnvMgrDefs*] const = Env_Error_Base + 3
BadSegment [*AccentType*] const = AccErr + 34
BadSegType [*AccentType*] const = AccErr + 33
BadTrap [*AccentType*] const = AccErr + 31
BadVPTable [*AccentType*] const = AccErr + 39
BadWildName [*SesameDefs*] const = Sesame_Error_Base + 7
BaseType [*SymDefs*] type = (T_Unknown, T_Char, T_Boolean, T_Integer, T_Enumerated, T_Real, T_Long, T_Pointer, T_Var, T_Routine, T_File, T_String, T_Set, T_Record, T_Array, T_Unused)
BIRecord [*BootInfo*] type = packed record case integer of 1: (IntBlk: array[0..255] of integer); 2: (OvlTable: array[0..11] of VirtualAddress; VP: VirtualAddress; PV: VirtualAddress; PVList: VirtualAddress; Sector: VirtualAddress; PCB: VirtualAddress; AST: VirtualAddress; AccentQueue: VirtualAddress; AccentFont: VirtualAddress; AccentCursor: VirtualAddress; AccentScreen: VirtualAddress; ScreenSize: integer; FreeVP: integer; FreeAST: integer; SchedProc: integer; InitProc: integer; BootChar: integer; NumProc: integer; StackSize: integer; GlobalSize: integer; NumSVReg: integer; TrapCode: integer; TrapArgs: VirtualAddress; MemBoard: integer; AccentStdCursor: VirtualAddress; AccentRoTemp: VirtualAddress; DefaultPartitionName: String[19]; IgnoreRunFile: Boolean; MachineInfo: MachineInfoRec; Filler: array[0..49 – WordSize(AstRecord)] of integer; FirstAst: ASTRecord; EtherIOArea: VirtualAddress; UserPtr: VirtualAddress; SVContext: record SV_CS: integer; SV_GP: integer; SV_LP: integer; SV_LocalSize: integer; SV_TrapCount: integer; SV_FirstRN: integer; SV_PC_Vector: array[0..121] of integer end) end
Bit1 [*AccentType*] type = 0..1
Bit10 [*AccentType*] type = 0..1023
Bit11 [*AccentType*] type = 0..2047
Bit12 [*AccentType*] type = 0..4095
Bit13 [*AccentType*] type = 0..8191
Bit14 [*AccentType*] type = 0..16383
Bit15 [*AccentType*] type = 0..32767
Bit16 [*AccentType*] type = integer
Bit2 [*AccentType*] type = 0..3
Bit3 [*AccentType*] type = 0..7
Bit32 [*AccentType*] type = packed record case integer of 1: (DblWord: array[0..1] of integer); 3: (Bit32Ptr: pBit32); 4: (AnyPtr: pointer); 5: (Byte: packed array[0..3] of Bit8); 6: (PageOffset: Bit8; LswPage: Bit8; MswPage: Bit16); 7: (Lng: Long); 8: (Blk: integer; Index: Bit12; Imag: Bit4); 13: (Field4: Bit8; Field3: Bit8; Field2: Bit8; Field1: Bit7; Field0: Bit1); 14: (Word0: Bit16; Word1: Bit16); 15: (Byte0: Bit8; Byte1: Bit8; Byte2: Bit8; Byte3: Bit8); end
Bit4 [*AccentType*] type = 0..15
Bit5 [*AccentType*] type = 0..31
Bit6 [*AccentType*] type = 0..63
Bit64 [*AccentType*] type = record lsw: long; msw: long; end
Bit7 [*AccentType*] type = 0..127
Bit8 [*AccentType*] type = 0..255
Bit9 [*AccentType*] type = 0..511
BitTable [*Spice_String*] type = set of Char
BlankStarString [*SaltError*] const = ''
BootBlockLocation [*BootInfo*] const = #20000000000
BorderOverhead [*SapphDefs*] const = 5
BOTTOM [*SapphDefs*] const = 32000
BreakKind [*Spice_String*] type = set of BreakType
BreakPointTrap [*AccentType*] const = AccErr + 45
BreakRecord [*Spice_String*] type = record Breakers: set of Char; Omitters: set of Char; Flags: BreakKind end
BreakTable [*Spice_String*] type = ↑BreakRecord
BreakType [*Spice_String*] type = (Append, Retain, Skip, FoldUp, FoldDown, Inclusive, Exclusive)
BusyRectangle [*AccentType*] const = AccErr + 57

ByteSizeOfAbsoluteLocalSymbol [*CFileDefs*] function(var sym: string): long
ByteSizeOfAbsoluteSymbol [*CFileDefs*] function(var sym: string): long
ByteSizeOfLibrarySymbol [*CFileDefs*] function(var libname: string): long
CantFork [*AccentType*] const = AccErr + 29
Cat3 [*Spice_String*] function(Str1, Str2, Str3: pstring): pstring
Cat4 [*Spice_String*] function(Str1, Str2, Str3, Str4: pstring): pstring
Cat5 [*Spice_String*] function(Str1, Str2, Str3, Str4, Str5: pstring): pstring
Cat6 [*Spice_String*] function(Str1, Str2, Str3, Str4, Str5, Str6: pstring): pstring
cChCmd [*SapphDefs*] const = 0
cdBlueDown [*SapphFileDefs*] const = 258
cdBlueUp [*SapphFileDefs*] const = 262
cdDiffPosResponse [*SapphFileDefs*] const = 267
cdGreenDown [*SapphFileDefs*] const = 259
cdGreenUp [*SapphFileDefs*] const = 263
cdListener [*SapphFileDefs*] const = 269
cdNoEvent [*SapphFileDefs*] const = 268
cdPosResponse [*SapphFileDefs*] const = 266
cdRegionExit [*SapphFileDefs*] const = 264
cdTimeout [*SapphFileDefs*] const = 265
cdWhiteDown [*SapphFileDefs*] const = 257
cdWhiteUp [*SapphFileDefs*] const = 261
cdYellowDown [*SapphFileDefs*] const = 256
cdYellowUp [*SapphFileDefs*] const = 260
CF_IOBoard [*Configuration*] function: Cf_IOBoardType
Cf_IOBoardType [*Configuration*] type = (Cf_CIO, Cf_EIO)
Cf_Monitor [*Configuration*] function: Cf_MonitorType
Cf_MonitorType [*Configuration*] type = (Cf_Landscape, Cf_Portrait)
Cf_Network [*Configuration*] function: Cf_NetworkType
Cf_NetworkType [*Configuration*] type = (Cf_CMUNet, Cf_10MBitNet)
Cf_OldZ80 [*Configuration*] function: Boolean
CFileVersion [*CFileDefs*] const = -4
ChainHead [*CFileDefs*] const = 12
ChangeExtensions [*PathName*] procedure(var Name: Path_Name; EList: Extension_List; NewExt: string)
ChangeUserParams [*Auth*] function(ServPort: Port; UserName: Auth_Var; CurrentPassword: Auth_Var; ChangePassword: Boolean; NewPassword: Auth_Var; NewProfile: APath_Name; NewShell: APath_Name): GeneralReturn
Char_Pool_Index [*CommandDefs*] type = long
Character_Pool [*CommandDefs*] type = packed array[0..0] of char
ChArray [*Stream*] type = packed array[1..1] of char
Check_Type [*AuthDefs*] type = (Check_Login, Check_User)
CheckHeap [*Dynamic*] procedure(S: HeapNumber)
CheckIn [*MsgN*] function(ServPort: Port; PortsName: string; Signature: Port; PortsID: Port): GeneralReturn
CheckOut [*MsgN*] function(ServPort: Port; PortsName: string; Signature: Port): GeneralReturn
CheckUser [*Auth*] function(ServPort: Port; UserName: Auth_Var; PassWord: Auth_Var; var UserRec: UserRecord): GeneralReturn
CImpInfo [*Code*] type = record case boolean of true: (ModuleName: SNArray; FileName: FNString); false: (Ary: array[0..0] of integer) end
CImpInfo [*SegDefs*] type = record case boolean of true: (ModuleName: SNArray; FileName: FNString); false: (Ary: array[0..0] of integer) end
CLoadNotCFile [*Cload*] const = -1
CLoadProcess [*Cload*] function(FileName: APath_Name; var FileInMem: pointer; var FileSize: long; Proc: Port; LoadDebug: Boolean): GeneralReturn

CloseIO [*IO*] function(ServPort: ServerIOPort): GeneralReturn
Cmd_EmptyCmdLine [*ExtraCmdParse*] const = -3
Cmd_NotFound [*ExtraCmdParse*] const = WS_NotFound
Cmd_NotInsMaybeSwitches [*ExtraCmdParse*] const = -4
Cmd_NotUnique [*ExtraCmdParse*] const = WS_NotUnique
Cmd_SomeError [*ExtraCmdParse*] const = -5
CmdChar [*CommandParse*] const = Chr(#200+24)
CmdFileChar [*CommandParse*] const = Chr(#200+26)
CmdParse_Error_Base [*CommandDefs*] const = 4200
Cmnd_String [*CommandParse*] type = String[MaxCmndString]
cNoCmd [*SapphDefs*] const = 1
CNTRLOOPS [*SapphFileDefs*] const = #225
CodeVersion [*Code*] const = '1.14'
command_file_leadin_char [*CommandParse*] const = '
Command_File_List [*CommandParse*] type = RECORD cmdFile: Text; isCharDevice: Boolean; next: pCommand_File_List; END
Command_Word_List [*CommandParse*] type = packed record ptrWordString: pWord_String; DeallocWordString: boolean; case WordClass: Word_Type of in_arg, out_arg, command_file: (NextArg: pCommand_Word_List); switch_arg: (NextSwitch: pCommand_Word_List; ValueOfSwitch: pCommand_Word_List; CorrespondingArg: pCommand_Word_List); switch_value: (); end
CommandBlock [*CommandDefs*] type = record WordCount: long; WordDirIndex: Char_Pool_Index; WordArrayPtr: pCharacter_Pool; WordArray_Cnt: Char_Pool_Index; end
CommandParseVersion [*CommandParse*] const = '5.7 of 3 Jun 84'
comment_leadin_char [*CommandParse*] const = '#'
CommentLen [*Code*] const = 80
CommentLen [*SegDefs*] const = 80
CompactIcons [*Sapph*] procedure(ServPort: Window)
CompletePathName [*PathName*] function(var WildPathName: Wild_Path_Name; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean; var Cursor: integer): long
ComputeProgress [*WindowUtils*] procedure(Current, Max: Long)
Concat [*Spice_String*] function(Str1, Str2: PString): PString
Confirm_NO [*ExtraCmdParse*] const = 2
Confirm_Switches [*ExtraCmdParse*] const = 3
Confirm_YES [*ExtraCmdParse*] const = 1
ConfirmUser [*Auth*] function(ServPort: Port; UserAuthPort: Port; var UserID: User_ID; var UserMachineName: Auth_Var): GeneralReturn
ConnectionInheritance [*Spawn*] type = (NewOne, Given, GivenReg)
ControlChar [*Stream*] type = 0..#37
ConvertPoolToString [*CommandParse*] function(ChPool: pCharacter_Pool; FirstChar: Char_Pool_Index; StringLength: Char_Pool_Index): Cmnd_String
ConvertStringToPool [*CommandParse*] procedure(CnvStr: Cmnd_String; var ChPool: pCharacter_Pool; var PoolLength: Char_Pool_Index)
ConvUpper [*Spice_String*] procedure(var Str: PString)
CopyEnvConnection [*EnvMgr*] function(ServPort: Port; OldConnection: Port; var NewConnection: port): GeneralReturn
Cos [*RealFunctions*] function(X: Real): Real
CosH [*RealFunctions*] function(x: real): real
CosHLarge [*RealFunctions*] exception(X: Real)
CosLarge [*RealFunctions*] exception(X: Real)
CoTan [*RealFunctions*] function(X: Real): Real
CoveredRectangle [*AccentType*] const = AccErr+56
CreateHeap [*Dynamic*] function: HeapNumber

CreateProcess [*AccInt*] function(ServPort: port; var HisKernelPort: port; var HisDataPort: port): GeneralReturn

CreateRectangle [*AccInt*] function(ServPort: port; RectPort: port; BaseAddr: VirtualAddress; ScanWidth: Integer; BaseX: Integer; BaseY: Integer; MaxX: Integer; MaxY: Integer; IsFont: Boolean): GeneralReturn

CreateSegment [*AccInt*] function(ServPort: port; ImagSegPort: port; SegmentKind: SpiceSegKind; InitialSize: Integer; MaxSize: Integer; Stable: Boolean; var Segment: SegID): GeneralReturn

CreateWindow [*Sapph*] function(ServPort: Window; fixedPosition: boolean; var leftx: integer; var topy: integer; fixedSize: boolean; var width: integer; var height: integer; hasTitle: boolean; hasborder: boolean; title: TitStr; var progName: ProgStr; hasIcon: boolean; var vp: Viewport): Window

CRLFConvention [*SysType*] const = false

CursorArrayRec [*SapphFileDefs*] type = record firstBlock: Packed Record numCursors: Integer; filler: Integer; offsets: Array[0..126] of record x: Integer; y: Integer; End; End; cursors: array[0..0] of PatternMap; end

CursorFunction [*SapphDefs*] type = (cfScreenOff, cfBroken, cfOR, cfXOR, cfCursorOff)

CursorSet [*SapphDefs*] type = LONG

CVD [*Spice_String*] function(Str: PString): integer

CVH [*Spice_String*] function(Str: PString): integer

CVHS [*Spice_String*] function(I: integer): PString

CVHSS [*Spice_String*] function(I: integer; W: integer): PString

CvInt [*Spice_String*] function(Str: PString; R: integer): integer

CvL [*Spice_String*] function(Str: PString; Radix: integer): long

CvLS [*Spice_String*] function(I: long; W: integer; Radix: integer; Fill: Pstring): Pstring

CVN [*Spice_String*] function(I: integer; W: integer; B: integer; Fill: Pstring): Pstring

CVO [*Spice_String*] function(Str: PString): integer

CVOS [*Spice_String*] function(I: integer): Pstring

CVOSS [*Spice_String*] function(I: integer; W: integer): Pstring

CVS [*Spice_String*] function(I: integer): Pstring

CVSS [*Spice_String*] function(I: integer; W: integer): Pstring

CvUp [*Spice_String*] function(Str: PString): PString

Data2State [*CFileDefs*] const = 2

Data_Format [*SesameDefs*] type = long

DATAPORT [*AccentType*] const = 2

DataState [*CFileDefs*] const = 1

Date_Fields [*TimeDefs*] type = packed record Year: integer; Month: 1..12; Day: 1..31; Weekday: 0..6; end

DateString [*ALoad*] function(date: Internal_Time): String

DeallocatePort [*AccInt*] function(ServPort: port; LocalPort: Port; Reason: Long): GeneralReturn

DeAllocIconVP [*Sapph*] procedure(ServPort: Window)

DebugMessage [*ProcMgrDefs*] type = record Head: Msg; tKPort: TypeType; KPort: Port; tArg1: TypeType; Arg1: Long; tArg2: TypeType; Arg2: Long; end

DEFAULTBACKLOG [*AccentType*] const = 0

DefaultInputName [*Stream*] var: STRING[255]

DefaultOutputName [*Stream*] var: STRING[255]

DEFAULTPTS [*AccentType*] const = 0

DefHeapSize [*Code*] const = #4

DefIncHeap [*Code*] const = #4

DefIncStack [*Code*] const = #4

DefinedGlobal [*CFileDefs*] const = InitializedSymbol

DefinedLocal [*CFileDefs*] const = LocalLabel

DefineFullSize [*Sapph*] procedure(ServPort: Window; exceptW: Window)

DefStackSize [*Code*] const = #20

DeleteChars [*Spice_String*] procedure(Var Str: PString; Index, Size: Integer)

DeleteRegion [*ViewPt*] procedure(ServPort: Viewport; regionNum: integer)
DeleteSearchWord [*CommandParse*] procedure(table: pWord_Search_Table; WordString: Cmnd_String)
DeleteWindow [*Sapph*] procedure(ServPort: Window)
DensityType [*IODefs*] type = (SingleDensity, DoubleDensity)
Deposit [*AccInt*] function(ServPort: port; RegOrStack: Boolean; Index: Integer; Value: Integer): GeneralReturn
DestroyChPool [*CommandParse*] procedure(var ChPool: pCharacter_Pool; var PoolLength: Char_Pool_Index)
DestroyCommandList [*CommandParse*] procedure(var argList: pCommand_Word_List)
DestroyCommandParse [*CommandParse*] procedure
DestroyHeap [*Dynamic*] procedure(S: HeapNumber)
DestroyRectangle [*AccInt*] function(ServPort: port; RectPort: port): GeneralReturn
DestroyRegions [*ViewPt*] procedure(ServPort: Viewport)
DestroySearchTable [*CommandParse*] procedure(var table: pWord_Search_Table)
DestroySegment [*AccInt*] function(ServPort: port; Segment: SegID): GeneralReturn
DestroyViewport [*ViewPt*] procedure(ServPort: Viewport)
DestroyVPCursors [*ViewPt*] procedure(ServPort: CursorSet)
DevPartString [*AccentType*] type = string[MAXDPCHARS]
DForm_16_Bit [*SesameDefs*] const = 16
DForm_32_Bit [*SesameDefs*] const = 32
DForm_36_Bit [*SesameDefs*] const = 36
DForm_8_Bit [*SesameDefs*] const = 8
DForm_CRLF_Text [*SesameDefs*] const = #413
DForm_LF_Text [*SesameDefs*] const = #410
DForm_Press [*SesameDefs*] const = #1000
DForm_Unspecified [*SesameDefs*] const = 0
Dir_Separator [*SesameDefs*] const = '/'
DirectIO [*AccInt*] function(ServPort: port; var CmdBlk: DirectIOArgs; var DataHdr: Header; var Data: DiskBuffer): GeneralReturn
DirectIOArgs [*AccentType*] type = Record IOStatus: Integer; UnitNumber: Integer; PhysAddress: Long; Command: DirIOCommands; end
DirectoryNotEmpty [*SesameDefs*] const = Sesame_Error_Base + 3
DirectoryNotFound [*SesameDefs*] const = Sesame_Error_Base + 2
DirIOCommands [*AccentType*] type = (DirIOInit, DirIORead, DirIOWrite, DirIOReadCheck, DirIOWriteCheck, DirIOTrackRead, DirIOTrackWrite, DirIOParamRead, DirIOBootRead, DirIOBootWrite, DirIOClose)
DisablePrivs [*PascalInit*] function(Proc: PORT): GeneralReturn
DiskAddr [*AccentType*] type = packed record case integer of 1: (lng: long); 2: (byte: packed array[0..3] of Bit8) end
DiskBuffer [*AccentType*] type = Packed Array[0..DISKBUFSIZE*2 – 1] of Bit8
DISKBUFSIZE [*AccentType*] const = PAGEBYTESIZE div 2
DiskErr [*AccentType*] const = AccErr + 32
DiskInterface [*AccentType*] type = (EIO, CIO, FlopDrives, MultiBus, Enet)
DiskParams [*AccentType*] type = Record DskBootSize: Integer; DskSectors: Integer; DskNumHeads: Integer; DskNumCylinders: Integer; DskSecCyl: Long; Case HDiskType: DiskType of D5Inch: (WriteCompCyl: Integer; LandingZone: Integer); D14Inch: (c24MByte: Boolean); End
DiskType [*AccentType*] type = (DUnused, D5Inch, D14Inch, D8Inch, DSMD, DFloppy)
DisposeP [*Dynamic*] procedure(var Where: pointer; L: integer)
DivZero [*Except*] exception
DONTCARE [*SapphDefs*] const = –32004
DONTWAIT [*AccentType*] const = 1
DstryCmdFiles [*CommandParse*] procedure(var inF: pCommand_File_List)

Dummy [*AccentType*] const = AccErr + 0
Dump [*Except*] exception(Message: String)
DynamicVersion [*Dynamic*] const = '1.0'
DynDebug [*Dynamic*] const = false
DynPrint [*Dynamic*] const = false
DynStats [*Dynamic*] procedure(Tr, Dump: boolean)
E10Address [*Ether10Defs*] type = record High: integer; Mid: integer; Low: integer end
E10ByteCount [*Ether10Defs*] const = 3002
E10ByteData [*Ether10Defs*] type = packed array[0..1499] of 0..255
E10BytesInHeader [*Ether10Defs*] const = 14
E10CharData [*Ether10Defs*] type = packed array[0..1499] of char
E10CRC [*Ether10Defs*] const = 3005
E10FilterType [*Ether10Defs*] type = E10Type
E10GetAdd [*Net10MB*] function(ServPort: E10Port; var Addr: E10Address): GeneralReturn
E10IntegerData [*Ether10Defs*] type = packed array[0..749] of integer
E10LongData [*Ether10Defs*] type = packed array[0..374] of long
E10MaxDataBytes [*Ether10Defs*] const = 1500
E10Message [*Ether10Defs*] type = record Head: Msg; Body: array[0..123] of integer; end
E10MinDataBytes [*Ether10Defs*] const = 46
E10NoFreeStructures [*Net10MB*] exception
E10NoNet [*Ether10Defs*] const = 3003
E10OK [*Ether10Defs*] const = 3000
E10Packet [*Ether10Defs*] type = packed record Dest: E10Address; Src: E10Address; PType: E10Type; case integer of 1: (BData: E10ByteData); 2: (CData: E10CharData); 3: (IData: E10IntegerData); 4: (LData: E10LongData); end
E10Port [*Ether10Defs*] type = Port
E10PortClear [*Net10MB*] function(ServPort: E10Port; PacketPort: E10Port): GeneralReturn
E10Receive [*Ether10Defs*] exception(ServPort: E10Port; Buff: pE10Packet; NumBytes: long)
E10RecvFailed [*Net10MB*] exception(Why: Integer)
E10RetVal [*Ether10Defs*] type = integer
E10Send [*Net10MB*] function(ServPort: E10Port; Buff: pE10Packet; Buff_Cnt: long): GeneralReturn
E10SendError [*Ether10Defs*] const = 3004
E10SendFailed [*Net10MB*] exception(Why: Integer)
E10SetFilter [*Net10MB*] function(ServPort: E10Port; PacketPort: E10Port; Which: E10FilterType): GeneralReturn
E10TimeOut [*Ether10Defs*] const = 3001
E10Type [*Ether10Defs*] type = integer
E10TypeClear [*Net10MB*] function(ServPort: E10Port; Which: E10FilterType): GeneralReturn
E10WordsInHeader [*Ether10Defs*] const = 7
EBadReply [*EtherUser*] exception
ELevel1Abort [*SaphEmrExceptions*] exception
ELevel1Debug [*SaphEmrExceptions*] exception
ELevel2Abort [*SaphEmrExceptions*] exception
ELevel3Abort [*SaphEmrExceptions*] exception
EMERGENCYMSG [*AccentType*] const = 1
EmergMsg [*Except*] exception
EMPort [*PascalInit*] var: port
EnableInput [*ViewPt*] procedure(ServPort: Viewport; keytrantab: VPStr255; timeout: integer)
EnableNotifyExceptions [*ViewPt*] procedure(ServPort: Viewport; notifyPort: Port; changed: boolean; exposed: boolean)
EnablePrivs [*PascalInit*] function(Proc: PORT): GeneralReturn
EnableRectangles [*AccInt*] function(ServPort: port; RectList: PtrPortArray; RectList_Cnt: long; Enable: Boolean): GeneralReturn

EnableWinListener [*Sapph*] procedure(ServPort: Window; abortPort: Port; keytranTab: VPStr255; timeOut: integer)
Entry_All [*SesameDefs*] const = 0
Entry_Data [*SesameDefs*] type = record case Entry_Type of Entry_File: (); Entry_Directory: (); Entry_Port: (EDPort: Port); # 400: (EDBytes: packed array[0..255] of bit8); # 401: (EDWords: array[0..127] of integer); # 402: (EDLongs: array[0..63] of long); # 403: (EDString: string[255]); end
Entry_Directory [*SesameDefs*] const = 2
Entry_File [*SesameDefs*] const = 1
Entry_Foreign [*SesDiskDefs*] const = 6
Entry_List [*SesameDefs*] type = ↑Entry_List_Array
Entry_List_Array [*SesameDefs*] type = array[0..0] of Entry_List_Record
Entry_List_Record [*SesameDefs*] type = record EntryName: Entry_Name; EntryVersion: long; EntryType: Entry_Type; NameStatus: Name_Status; end
Entry_Name [*SesameDefs*] type = string[Entry_Name_Size]
Entry_Name_Size [*SesameDefs*] const = 80
Entry_Port [*SesameDefs*] const = 3
Entry_RESERVED [*SesameDefs*] type = 4.. # 377
Entry_Type [*SesameDefs*] type = 0.. # 77777
Entry_UserDefined [*SesameDefs*] type = # 400.. # 77777
Env_Element [*EnvMgrDefs*] type = string[Env_Element_Size]
Env_Element_Array [*EnvMgrDefs*] type = array[0..0] of Env_Element
Env_Element_Size [*EnvMgrDefs*] const = 255
Env_Error_Base [*EnvMgrDefs*] const = 1600
env_quoted_bracket_char [*CommandParse*] const = '"'
Env_Scan_Array [*EnvMgrDefs*] type = array[0..0] of Env_Scan_Record
Env_Scan_List [*EnvMgrDefs*] type = ↑Env_Scan_Array
Env_Scan_Record [*EnvMgrDefs*] type = record VarName: Env_Var_Name; VarType: Env_Var_Type; VarScope: Env_Var_Scope; end
env_var_bracket_char [*CommandParse*] const = '↑'
Env_Var_Name [*EnvMgrDefs*] type = string[Env_VarName_Size]
Env_Var_Scope [*EnvMgrDefs*] type = (Env_Normal, Env_Local, Env_Global)
Env_Var_Type [*EnvMgrDefs*] type = (Env_String, Env_SearchList)
Env_Variable [*EnvMgrDefs*] type = ↑Env_Element_Array
Env_VarName_Size [*EnvMgrDefs*] const = Entry_Name_Size
EnvCompletePathName [*PathName*] function(EnvConnection: Port; var WildPathName: Wild_Path_Name; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean; var Cursor: integer): long
EnvDisconnect [*EnvMgr*] function(ServPort: Port): GeneralReturn
EnvFindWildPathnames [*PathName*] function(EnvConnection: Port; var WildPathName: Path_Name; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean; NameFlags: Name_Flags; EntryType: Entry_Type; var FoundInFirst: boolean; var DirName: APath_Name; var EntryList: Entry_List; var EntryList_Cnt: long): GeneralReturn
EnvVariableNotFound [*EnvMgrDefs*] const = Env_Error_Base + 1
eofChar [*CommandParse*] const = chr(0)
eolnChar [*CommandParse*] const = chr(# 12)
ErAnyError [*CommandDefs*] const = CmdParse_Error_Base + 14
ErBadCmd [*CommandDefs*] const = CmdParse_Error_Base + 2
ErBadQuote [*CommandDefs*] const = CmdParse_Error_Base + 13
ErBadSwitch [*CommandDefs*] const = CmdParse_Error_Base + 1
ErCmdNotUnique [*CommandDefs*] const = CmdParse_Error_Base + 8
ErCmdParam [*CommandDefs*] const = CmdParse_Error_Base + 6
EReceive [*AccCall*] function(var xxmsg: Msg; MaxWait: long; PortOpt: PortOption; Option: ReceiveOption): GeneralReturn

EReceiveFailed [*EtherUser*] exception(Why: GeneralReturn)
ErIllCharAfter [*CommandDefs*] const = CmdParse_Error_Base + 12
ErNoCmdParam [*CommandDefs*] const = CmdParse_Error_Base + 4
ErNoOutFile [*CommandDefs*] const = CmdParse_Error_Base + 9
ErNoSwParam [*CommandDefs*] const = CmdParse_Error_Base + 3
ErOneInput [*CommandDefs*] const = CmdParse_Error_Base + 10
ErOneOutput [*CommandDefs*] const = CmdParse_Error_Base + 11
ErrorMsgPMBroadcast [*SaltError*] procedure(GR: GeneralReturn; ER_Type: GR_Error_Type; ProgName: String; InMsg: PString)
ErSwNotUnique [*CommandDefs*] const = CmdParse_Error_Base + 7
ErSwParam [*CommandDefs*] const = CmdParse_Error_Base + 5
EscKind [*SapphFileDefs*] type = (CodeNormal, CodeSame, EscReturn, EscNoop)
ESendFailed [*EtherUser*] exception(Why: GeneralReturn)
EStack [*Except*] exception
EStackTooDeep [*AccentType*] const = AccErr + 42
EtherHeader [*EtherTypes*] type = PACKED RECORD Src: 0..255; Dst: 0..255; Typ: INTEGER; END
EtherIDBase [*EtherTypes*] const = 800
EtherPacket [*EtherTypes*] type = RECORD CASE Integer OF 0: (Header: EtherHeader; DataWords: ARRAY[0..MaxEtherWords - WordSize(EtherHeader) - 1] OF INTEGER); 1: (Words: ARRAY[0..MaxEtherWords - 1] OF INTEGER); 2: (LongHeader: EtherHeader; LongWords: ARRAY[0..(MaxEtherWords - WordSize(EtherHeader)) div 2 - 1] OF LONG); 3: (ConfigHdr: EtherHeader; SkipC: integer; ConfigBytes: PACKED ARRAY[0..2*(MaxEtherWords - WordSize(EtherHeader) - 1)] OF Bit8) END
EtherPacketBytes [*EtherTypes*] const = 2*WordSize(MsgEtherPacket)
EtherTyp3ConfigTest [*EtherTypes*] const = #220
EtherTypConfigTest [*EtherTypes*] const = #220
EtherTypEchoMe [*EtherTypes*] const = #700
EtherTypIAmAnEcho [*EtherTypes*] const = #701
EtherTypPup [*EtherTypes*] const = #1000
EViewPtChanged [*SaphEmrExceptions*] exception(vp: Viewport; newx, newy, neww, newh, newRank: Integer)
EViewPtExposed [*SaphEmrExceptions*] exception(vp: Viewport; ra: pRectArray; numRectangles: Long)
Examine [*AccInt*] function(ServPort: port; RegOrStack: Boolean; Index: Integer; var Value: Integer): GeneralReturn
ExceptVersion [*Except*] const = '3.4'
Exec [*Spawn*] function(VAR ChildKPort: Port; VAR ChildDPort: Port; ProcessName: APath_Name; HisCommand: CommandBlock): GeneralReturn
ExerciseParseEngine [*CommandParse*] function(ChPool: pCharacter_Pool; PoolLength: Char_Pool_Index; procedure ReadPool(var Pool: pCharacter_Pool; var PLen: Char_Pool_Index); var inputs: pCommand_Word_List; var outputs: pCommand_Word_List; var switches: pCommand_Word_List): GeneralReturn
ExitAllCmdFiles [*CommandParse*] procedure(var inF: pCommand_File_List)
ExitCmdFile [*CommandParse*] procedure(var inF: pCommand_File_List)
ExitGRError [*PascalInit*] exception(GR: GeneralReturn)
ExitProgram [*PascalInit*] exception
Exp [*RealFunctions*] function(X: Real): Real
ExpandPathName [*PathName*] function(var WildPathName: Wild_Path_Name; ImplicitSearchList: Env_Var_Name): GeneralReturn
ExpandWindow [*Sapph*] procedure(ServPort: Window)
ExpLarge [*RealFunctions*] exception(X: Real)
EXPLICITDEALLOC [*AccentType*] const = 0
ExpSmall [*RealFunctions*] exception(X: Real)
Extension_List [*PathName*] type = string[Extension_String_Size]
Extension_String_Size [*PathName*] const = 80

ExtractAllRights [*AccInt*] function(ServPort: port; PortIndex: Long; var PortRight: port; var PortType: Integer): GeneralReturn

ExtractEvent [*ModGetEvent*] function(repMsg: Pointer; var vp: ViewPort; var k: KeyEvent): Boolean

ExtractSimpleName [*PathName*] procedure(Name: Path_Name; var StartTerminal: integer; var StartVersion: integer)

Failure [*AccentType*] const = AccErr + 24

FEarray [*RunDefs*] type = array[FileIndex] of FileEntry

FHdr_Access_Rights_Offset [*SesDiskDefs*] const = 56

File_Data [*SesameDefs*] type = pointer

File_Header [*SesameDefs*] type = packed record FileSize: long; DataFormat: long; PrintName: Print_Name; Author: Group_ID; CreationDate: Internal_Time; AccessID: Group_ID; AccessDate: Internal_Time; FHdr_RESERVED: array[56..64] of integer; end

FileEntry [*RunDefs*] type = record FileType: LinkFileType; FileLocation: long; FileBlocks: long; FileName: APath_Name; WriteDate: Internal_Time; Version: long; end

FileIndex [*RunDefs*] type = 0..RMAXFILE

FileKind [*Stream*] type = (BlockStructured, CharacterStructured)

FileLength [*Code*] const = 100

FileLength [*SegDefs*] const = 100

FileType [*Stream*] type = packed record Flag: packed record case integer of 0: (CharReady: boolean; FEoln: boolean; FEof: boolean; FNotReset: boolean; FNotOpen: boolean; FNotRewrite: boolean; FExternal: boolean; FBusy: boolean; FKind: FileKind; FLastWasEof: boolean; Funused: 0..63); 1: (skip1: 0..3; ReadError: 0..7); 2: (skip2: 0..15; WriteError: 0..3) end; EolCh, EofCh, EraseCh, NoiseCh: ControlChar; OmitCh: set of ControlChar; FileNum: integer; FIndex: integer; Length: integer; BlockNumber: integer; StreamP: pStreamF; LengthInBlocks: integer; LastBlockLength: integer; SizeInWords: integer; SizeInBits: 0..16; ElsPerWord: 0..16; Element: record case integer of 1: (C: char); 2: (W: array[0..0] of integer) end end

FindExtendedFileName [*PathName*] function(var FileName: Path_Name; ExtensionList: Extension_List; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean): GeneralReturn

FindExtendedPathName [*PathName*] function(var PathName: Path_Name; ExtensionList: Extension_List; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean; var EntryType: Entry_Type; var NameStatus: Name_Status): GeneralReturn

FindFileName [*PathName*] function(var FileName: Path_Name; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean): GeneralReturn

FindPathName [*PathName*] function(var PathName: Path_Name; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean; var EntryType: Entry_Type; var NameStatus: Name_Status): GeneralReturn

FindTypedName [*PathName*] function(var PathName: Path_Name; ExtensionList: Extension_List; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean; var EntryType: Entry_Type; var NameStatus: Name_Status): GeneralReturn

FindWildPathnames [*PathName*] function(var WildPathName: Path_Name; ImplicitSearchList: Env_Var_Name; FirstOnly: boolean; NameFlags: Name_Flags; EntryType: Entry_Type; var FoundInFirst: boolean; var DirName: APath_Name; var EntryList: Entry_List; var EntryList_Cnt: long): GeneralReturn

FinEther [*EtherUser*] procedure

First_User [*AuthDefs*] const = 1

FirstBlock [*CFileDefs*] type = record FileVersion: integer; FileSize: long; FileTimeStamp: Internal_Time; SymbolAreaSize: long; TextAreaSize: long; DataAreaSize: long; BSSAreaSize: long; DestAddr: long; StartAddr: long; MainAddr: long; InitialLocalSize: long; StackBaseAddress: long; StackSizeInPages: long; end

FirstItemNotDefined [*EnvMgrDefs*] const = Env_Error_Base + 5

FIRSTNONRESERVEDPORT [*AccentType*] const = 3

FirstUserIndex [*PascalInit*] const = 6

FiveDeep [*AccentType*] const = AccErr + 38

FloppyResultStatus [*IODefs*] type = packed record StatusType: FloppyResultType; Unused: Bit6; case FloppyResultType of NoStatus: (); HeadChange, DriveSense, CmdResults: (Unit: Bit2; Head: Bit1; case FloppyResultType of DriveSense: (TwoSided: boolean; AtTrack0: boolean; DriveReady: boolean; WriteProtected: boolean; DriveFault: boolean); HeadChange, CmdResults: (NotReady: boolean; EquipFault: boolean; SeekEnd: boolean; IntrCode: (Normal, Abnormal, InvalidCmd, DriveRdyChange); case FloppyResultType of HeadChange: (PresentCylinder: Bit8); CmdResults: (NoAddrMark: boolean; NotWritable: boolean; NoData: boolean; Unused1: Bit1; Overrun: boolean; DataError: boolean; Unused2: Bit1; TrackEnd: boolean; NoDataAddrMark: boolean; BadTrack: boolean; ScanFail: boolean; ScanHit: boolean; WrongCylinder: boolean; DataCRCError: boolean; ControlMark: boolean; Unused3: Bit1; CylinderID: Bit8; HeadID: Bit8; SectorID: Bit8; SectorSizeCode: Bit8))) end

FloppyResultType [*IODefs*] type = (NoStatus, HeadChange, DriveSense, CmdResults)

FlushEvents [*ViewPt*] function(ServPort: Viewport): boolean

FNString [*Code*] type = String[FileLength]

FNString [*SegDefs*] type = String[FileLength]

FontCharWidthVector [*ViewPt*] procedure(ServPort: Viewport; ch: char; var dx: integer; var dy: integer)

FontHeadOverhead [*SapphFileDefs*] const = #404

FontMap [*SapphFileDefs*] type = ↑FontMapRec

FontMapRec [*SapphFileDefs*] type = packed record Height: integer; Base: integer; Index: array[0..#177] of packed record Offset: 0..767; Line: 0..63; Width: integer; end; Filler: array[0..1] of integer; Pat: Array[0..0] of integer; end

FontSize [*ViewPt*] procedure(ServPort: Viewport; var name: String; var PointSize: integer; var Rotation: integer; var FaceCode: integer; var maxWidth: integer; var maxHeight: integer; var xOrigin: integer; var yOrigin: integer; var fixedWidth: boolean; var fixedHeight: boolean)

FontStringWidthVector [*ViewPt*] procedure(ServPort: Viewport; str: VPStr255; firstCh: integer; lastch: integer; var dx: integer; var dy: integer)

FontWordWidth [*SapphFileDefs*] const = 48

ForBootFile [*SpawnInitFlags*] const = FALSE

Fork [*AccInt*] function(ServPort: port; var HisKernelPort: port; var HisDataPort: port; var Ports: PtrPortArray; var Ports_Cnt: long): GeneralReturn

ForLibFile [*SpawnInitFlags*] const = NOT ForBootFile

FudgeStack [*Code*] const = #2000

FullLn [*Stream*] function(var F: Text): Boolean

FullWindowState [*Sapph*] procedure(ServPort: Window; var leftx: integer; var topy: integer; var outerwidth: integer; var outerHeigh: integer; var rank: integer; var hasBorder: boolean; var hasTitle: boolean; var isListener: boolean; var name: ProgStr; var title: TitStr)

FwdCode [*EtherTypes*] const = 2

GeneralReturn [*AccentType*] type = integer

GetAbsoluteDef [*CFileDefs*] procedure(var BytePtr: long; var ByteAddress: long; var Name: LString)

GetAbsoluteLocalDef [*CFileDefs*] procedure(var BytePtr: long; var ByteAddress: long; var Name: LString)

GetAsmlong [*CFileDefs*] function(var ByteAddr: long): long

GetAsmString [*CFileDefs*] function(var ByteAddr: long): LString

GetB [*Stream*] procedure(var F: Filetype)

GetBreak [*Spice_String*] function: BreakTable

GetC [*Stream*] procedure(var F: Filetype)

GetChainHead [*CFileDefs*] procedure(var BytePtr: long; var AreaDesg: Bit8; var ByteAreaOff: long)

GetCharacterPool [*ExtraCmdParse*] procedure(prompt: Cmnd_String; var InputFile: Text; var ChPool: pCharacter_Pool; var PoolLength: Char_Pool_Index)

GetCmd [*ExtraCmdParse*] function(prompt: Cmnd_String; SearchTable: pWord_Search_Table; var CmdName: Cmnd_String; var inF: pCommand_File_List; var inputs: pCommand_Word_List; var outputs: pCommand_Word_List; var switches: pCommand_Word_List; var ErrorGR: GeneralReturn): integer

GetCodeByte [*CFileDefs*] function(var ByteAddr: long): Bit8

GetCodeWord [*CFileDefs*] function(var ByteAddr: long): integer

GetConfirm [*ExtraCmdParse*] function(prompt: Cmnd_String; def: integer; var switches: pCommand_Word_List): integer

GetDateTime [*Time*] function(ServPort: port): Internal_Time

GetDiskPartitions [*AccInt*] function(ServPort: port; interface: DiskInterface; log_unit: InterfaceInfo; var unitnum: Integer; var DevName: DevPartString; var PartL: Pointer; var PartL_Cnt: long): GeneralReturn

GetEnvVariable [*EnvMgr*] function(ServPort: Port; Name: Env_Var_Name; SearchScope: Env_Var_Scope; var Variable: Env_Variable; var Variable_Cnt: long; var VarType: Env_Var_Type; var ActualScope: Env_Var_Scope): GeneralReturn

GetEvent [*ViewPt*] function(ServPort: Viewport; howWait: KeyHowWait): KeyEvent

GetEventPort [*ModGetEvent*] procedure(ServPort: Viewport; howWait: KeyHowWait; retPort: Port)

GetFullViewport [*ViewPt*] function(ServPort: Viewport): Viewport

GetFullWindow [*Sapph*] function(ServPort: Window): Window

GetIconViewport [*Sapph*] procedure(ServPort: Window; var iconvp: Viewport; var width: integer; var height: integer)

GetIconWindow [*Sapph*] function(ServPort: Window): Window

GetIOSleepID [*AccCall*] function(var SleepID: long): GeneralReturn

GetIthWordPtr [*CommandParse*] function(i: long; CmndBlock: CommandBlock): pWord_String

GetLibraryDef [*CFileDefs*] procedure(var BytePtr: long; var LibraryFileName: LString; var LibraryTimeStamp: Internal_Time; var LibraryLoc: long)

GetListenerWindow [*Sapph*] function(ServPort: Window): Window

GetOffsetDef [*CFileDefs*] procedure(var BytePtr: long; var WdOffsetAmt: long)

GetParsedUserInput [*ExtraCmdParse*] function(prompt: Cmnd_String; var inF: pCommand_File_List; var inputs: pCommand_Word_List; var outputs: pCommand_Word_List; var switches: pCommand_Word_List): GeneralReturn

GetPortIndexStatus [*AccInt*] function(ServPort: port; PortIndex: Long; var Backlog: Integer; var NWaitingMsgs: Integer; var EWaitingMsgs: Integer; var PortRight: port; var PortType: Integer): GeneralReturn

GetPortStatus [*AccInt*] function(ServPort: port; PortRight: port; var Backlog: Integer; var NWaitingMsgs: Integer; var EWaitingMsgs: Integer; var PortIndex: Long; var PortType: Integer): GeneralReturn

GetPrimaryDef [*CFileDefs*] procedure(var BytePtr: long; var SimKind: Bit8; var AreaDesg: Bit8; var AreaOffset: long; var SymSize: long; var SymName: LString)

GetRectangleParms [*AccInt*] function(ServPort: port; RectPort: port; var BaseAddr: VirtualAddress; var ScanWidth: Integer; var BaseX: Integer; var BaseY: Integer; var MaxX: Integer; var MaxY: Integer; var IsFont: Boolean): GeneralReturn

GetRegionCursor [*ViewPt*] procedure(ServPort: Viewport; regionNum: integer; var cursorImage: CursorSet; var cursIndex: integer; var cursFunc: CursorFunction; var track: boolean)

GetRegionParms [*ViewPt*] procedure(ServPort: Viewport; regionNum: integer; var absolute: boolean; var speed: integer; var minx: integer; var maxx: integer; var miny: integer; var maxy: integer; var modx: integer; var posx: integer; var mody: integer; var posy: integer)

GetScreenParameters [*Sapph*] procedure(ServPort: Window; var width: integer; var height: integer)

GetShellCmd [*ExtraCmdParse*] function(SearchTable: pWord_Search_Table; var CmdName: Cmnd_String; var inF: pCommand_File_List; var inputs: pCommand_Word_List; var outputs: pCommand_Word_List; var switches: pCommand_Word_List; var ErrorGR: GeneralReturn): integer

GetStringTime [*Time*] function(ServPort: port; TimeFormat: integer): String

GetSysFont [*ViewPt*] function(ServPort: Viewport): Viewport

GetUserName [*Auth*] function(ServPort: Port; UserID: User_ID; var UserName: Auth_Var): GeneralReturn

GetUserTime [*Time*] function(ServPort: port): User_Time

GetViewportBit [*ViewPt*] function(ServPort: Viewport; x: integer; y: integer; var value: boolean): boolean

GetViewportRectangle [*ViewPt*] function(ServPort: Viewport; x: integer; y: integer; width: integer; height: integer; var Data: pVPIntegerArray; var Data_Cnt: long; var WordsAcross: integer; ux: integer; uy: integer): boolean

GetVPRank [*ViewPt*] function(ServPort: Viewport): integer

GetWinNames [*Sapph*] procedure(ServPort: Window; var names: pWinNameArray; var names_Cnt: long; var curListenIndex: integer)

GetWinProcess [*Sapph*] function(ServPort: Window): Port

GlobalProcedure [*CFileDefs*] const = 3

GPBuffer [*AccentType*] type = packed array[1..8] of Bit8

GPIBDevCmdHead [*IODefs*] type = packed record Options: packed record SetInt0Mask: boolean; SetInt1Mask: boolean; OmitBusConfig: boolean; OmitUnListen: boolean; case boolean of true: (HoldOffOnEOI: boolean; unused2: Bit3); false: (OmitGoToStandby: boolean; WaitOnData: boolean; ForceEOI: boolean; unused1: Bit1); end; Int0Mask: Bit8; Int1Mask: Bit8; PrimAddr: Bit8; SecAddr: Bit8; case boolean of true: (ReadCount: Bit8); false: (); end

GPIBSenseStatus [*IODefs*] type = packed record IntStat0: Bit8; IntStat1: Bit8; IntAddrStat: Bit8; IntBusStat: Bit8; IntAddrSwch: Bit8; IntCmdPass: Bit8; CurAddrStat: Bit8; CurBusStat: Bit8; CurAddrSwch: Bit8; CurCmdPass: Bit8; end

GPIBWriteRegister [*IODefs*] type = packed record case RegNum: Bit8 of 0: (); 1: (RegVal: Bit8) end

GR_Error_Type [*SaltError*] type = (GR_Warning, GR_Error, GR_FatalError)

GRError [*PascalInit*] exception(GR: GeneralReturn)

GRErrorMsg [*SaltError*] procedure(GR: GeneralReturn; ER_Type: GR_Error_Type; ProgName: String; InMsg: PString; var OutMsg: PString)

Group_ID [*SesameDefs*] type = long

GRStdErr [*SaltError*] function(GR: GeneralReturn; ER_Type: GR_Error_Type; InMsg: PString; var OutMsg: PString): boolean

GRStdError [*SaltError*] procedure(GR: GeneralReturn; ER_Type: GR_Error_Type; InMsg: PString; var OutMsg: PString)

GRWriteErrorMsg [*SaltError*] procedure(GR: GeneralReturn; ER_Type: GR_Error_Type; ProgName: String; InMsg: PString)

GRWriteStdError [*SaltError*] procedure(GR: GeneralReturn; ER_Type: GR_Error_Type; InMsg: PString)

Header [*AccentType*] type = packed Array[0..15] of Bit8

HeapAddress [*Dynamic*] type = record case integer of 0: (Offset: HeapOffset; Segment: HeapNumber); 1: (AnyPtr: Pointer); 2: (LongInt: Long); end

HeapNumber [*Dynamic*] type = integer

HeapOffset [*Dynamic*] type = integer

IconAutoUpdate [*Sapph*] procedure(ServPort: Window; allowed: boolean)

IconHeight [*SapphDefs*] const = 64

IconWidth [*SapphDefs*] const = 64

IDCIrEtherFilter [*EtherTypes*] const = EtherIDBase + 3

IDCIrPupFilter [*EtherTypes*] const = EtherIDBase + 6

Identifier [*Stream*] type = string[IdentLength]

IdentifyWindow [*Sapph*] procedure(ServPort: Window)

IdentLength [*Stream*] const = 8

IdentTable [*Stream*] type = array[0..1] of Identifier

IDGetEtherAddress [*EtherTypes*] const = EtherIDBase + 1

IdNotDefined [*Stream*] exception(FileName: SName; Id: Identifier)

IdNotUnique [*Stream*] exception(FileName: SName; Id: Identifier)

IDRCIrEtherFilter [*EtherTypes*] const = EtherIDBase + 103

IDRCIrPupFilter [*EtherTypes*] const = EtherIDBase + 106

IDRecvEtherPacket [*EtherTypes*] const = EtherIDBase + 104

IDRecvPupPacket [*EtherTypes*] const = EtherIDBase + 107

IDRGetEtherAddress [*EtherTypes*] const = EtherIDBase + 101

IDRSetEtherFilter [*EtherTypes*] const = EtherIDBase + 102

IDRSetPupFilter [*EtherTypes*] const = EtherIDBase + 105

IDSendEtherPacket [*EtherTypes*] const = EtherIDBase + 4
IDSendPupPacket [*EtherTypes*] const = EtherIDBase + 7
IDSetEtherFilter [*EtherTypes*] const = EtherIDBase + 2
IDSetPupFilter [*EtherTypes*] const = EtherIDBase + 5
IllegalBacklog [*AccentType*] const = AccErr + 10
IllegalScanWidth [*AccentType*] const = AccErr + 55
ImpArray [*RunDefs*] type = packed array[0..4096] of SegIndex
ImpIndex [*RunDefs*] type = 0..RMAXIMPORT
ImpNode [*Code*] type = record SId: SNArray; FilN: pFNString; XGP: integer; XSN: integer; Seg: pSegNode; Next: pImpNode end
Impossible [*ViewPt*] exception(s: String)
ImproperEntryType [*SesameDefs*] const = Sesame_Error_Base + 11
in_out_separator_char [*CommandParse*] const = '~'
InactiveSegment [*AccentType*] const = AccErr + 47
Index1Unquoted [*PathName*] function(S: Wild_Path_Name; C: char): integer
IndexInterpose [*AccInt*] function(ServPort: port; MyPort: Port; HisIndex: Long; var HisPort: Port): GeneralReturn
Inf [*Spice_String*] const = -32742
InitAccInt [*AccInt*] procedure(RPort: port)
InitAuth [*Auth*] procedure(RPort: port)
InitCmdFile [*CommandParse*] procedure(var inF: pCommand_File_List)
InitCommandParse [*CommandParse*] procedure
InitDynamic [*Dynamic*] procedure
InitEnvMgr [*EnvMgr*] procedure(RPort: port)
InitEther [*EtherUser*] procedure(RPort: port; Heap: integer)
InitExceptions [*Except*] procedure
Initial [*Spice_String*] function(Str1, Str2: PString): boolean
InitializedSymbol [*CFileDefs*] const = 5
InitIO [*IO*] procedure(RPort: port)
InitMsgId [*PascalInit*] const = 32896
InitMsgN [*MsgN*] procedure(RPort: port)
InitMsgSize [*PascalInit*] const = WORDSIZE(InitMsgType)*2
InitMsgType [*PascalInit*] type = RECORD Head: Msg; DefaultType: TypeType; DefInName: STRING[255]; DefOutName: STRING[255]; PassPortType: TypeType; PassedPorts: PtrPortArray; UWinSharedType: TypeType; UWinShared: Boolean; EMType: TypeType; EMPort: Port; WindowType: TypeType; UserWindow: Port; TSType: TypeType; UserTS: Port; WordCountType: TypeType; WordCount: long; WordDirIndexType: TypeType; WordDirIndex: long; WordArrayPtrType: TypeType; WordArrayPtrEltSize: integer; WordArrayPtrTName: integer; WordArray_Cnt: long; WordArrayPtr: pCharacter_Pool; ComputeEnvType: TypeType; ComputingEnvironment: Long; END
InitNet10MB [*Net10MB*] procedure(RPort: port)
InitPascal [*PascalInit*] procedure
InitProcess [*PascalInit*] procedure(AmIClone: BOOLEAN)
InitProcMgr [*ProcMgr*] procedure(RPort: port)
InitSapph [*Sapph*] procedure
InitSesame [*Sesame*] procedure(RPort: port)
InitSesDisk [*SesDisk*] procedure(RPort: port)
InitStream [*Stream*] procedure
InitTime [*Time*] procedure(RPort: port)
InitTS [*TS*] procedure(RPort: port)
InitViewPt [*ViewPt*] procedure
InitWordSearchTable [*CommandParse*] procedure(var table: pWord_Search_Table; CaseSensitive: boolean)
InPorts [*PascalInit*] var: ptrPortArray

InPorts_Cnt [*PascalInit*] var: long
InsertAllRights [*AccInt*] function(ServPort: port; PortIndex: Long; PortRight: port; PortType: Integer): GeneralReturn
InsertChars [*Spice_String*] procedure(Source: PString; var Dest: PString; Index: Integer)
InterceptSegmentCalls [*AccInt*] function(ServPort: port; var OldSysPorts: PtrAllPortArray; var OldSysPorts_Cnt: long; var SysPorts: PtrPortArray; var SysPorts_Cnt: long): GeneralReturn
InterfaceInfo [*AccentType*] type = Packed Record Case DiskInterface Of EIO, CIO, FlopDrives: (Unit: Integer); ENet: (EAddr: Packed Array[0..2] of integer); End
Internal_Time [*TimeDefs*] type = record Weeks: integer; MSecInWeek: long; end
Intr [*AccentType*] const = AccErr + 12
InvalidateMemory [*AccInt*] function(ServPort: port; Address: VirtualAddress; NumBytes: Long): GeneralReturn
InvalidDirectoryVersion [*SesameDefs*] const = Sesame_Error_Base + 6
InvalidVersion [*SesameDefs*] const = Sesame_Error_Base + 5
InxCase [*Except*] exception
IO_Version [*IO*] function(ServPort: ServerNamePort): String
IOBadBaudRate [*IODefs*] const = IOErr + 12
IOBadBufferSize [*IODefs*] const = IOErr + 35
IOBadCmdBlkCount [*IODefs*] const = IOErr + 8
IOBadCylinderNumber [*IODefs*] const = IOErr + 21
IOBadDataByteCount [*IODefs*] const = IOErr + 9
IOBadHeadNumber [*IODefs*] const = IOErr + 22
IOBadPortReference [*IODefs*] const = IOErr + 6
IOBadRegisterNumber [*IODefs*] const = IOErr + 10
IOBadSectorNumber [*IODefs*] const = IOErr + 20
IOBadTrack [*IODefs*] const = IOErr + 31
IOBadUserEventPort [*IODefs*] const = IOErr + 5
IOBaseMsgID [*IODefs*] const = 4000
IOCircBufOverFlow [*IODefs*] const = IOErr + 17
IOCmdBlk [*IODefs*] type = packed record CmdIDTag: long; case integer of 0: (case integer of 0: (CmdByte: packed array[0..0] of Bit8); 1: (CmdWord: packed array[0..0] of integer); 2: (WriteReg: packed array[stretch(0)..stretch(0)] of packed record RegNum: Bit8; RegVal: Bit8 end)); 1: (case IOCommand of IOSetAttention: (GPIBEnableATN: boolean); IOSetStream: (GPIBEnableStream: boolean; GPIBBlockingFactor: integer); IOSetBufferSize: (GPIBBufferSize: long); IOWriteRegisters: (GPIBWriteReg: packed array[stretch(0)..stretch(0)] of GPIBWriteRegister); IODevRead, IODevWrite: (GPIBDevCmdBlk: GPIBDevCmdHead)); 2: (case IOCommand of IOSetAttention: (RS232EnableATN: boolean); IOSetStream: (RS232EnableStream: boolean; RS232BlockingFactor: integer); IOSetBufferSize: (RS232BufferSize: long); IOSetBaud: (RS232TxBaud: Bit8; RS232RxBaud: Bit8); IOWriteRegisters: (RS232WriteReg: packed array[stretch(0)..stretch(0)] of SIOWriteRegister)); 3: (case IOCommand of IOSetAttention: (SpeechEnableATN: boolean); IOSetBufferSize: (SpeechBufferSize: long); IOSetBaud: (SpeechTxRate: integer); IOWriteRegisters: (SpeechWriteReg: packed array[stretch(0)..stretch(0)] of SIOWriteRegister)); 4: (case IOCommand of IOSetAttention: (FloppyEnableATN: boolean); IOSetDensity: (FloppyDensity: DensityType); IORead, IOWrite, IOFormat, IOSeek, IORecalibrate, IOReadID, IOSenseDrive: (FloppyUnit: Bit8; FloppyHead: Bit8; case IOCommand of IORead, IOWrite, IOFormat, IOSeek: (FloppyCylinder: Bit8; case IOCommand of IORead, IOWrite: (FloppySector: Bit8); IOFormat: (FloppyFmtData: Bit8)))) end
IOCommand [*IODefs*] type = (IOSense, IOReset, IOWriteRegisters, IOFlushInput, IOFlushOutPut, IORead, IOWrite, IOWriteEOI, IOReadHiVol, IOWriteHiVol, IODevRead, IODevWrite, IOSetBaud, IOSetStream, IOSetAttention, IOAbort, IOSuspend, IOResume, IOSeek, IORecalibrate, IOFormat, IOReadID, IOSenseDrive, IOSetDensity, IOSetBufferSize, IONullCmd)
IOCylinderMisMatch [*IODefs*] const = IOErr + 32
IODataCRCError [*IODefs*] const = IOErr + 29
IODeviceNotFree [*IODefs*] const = IOErr + 3
IODeviceNotReady [*IODefs*] const = IOErr + 24

IODeviceNotWritable [*IODefs*] const = IOErr + 27
IODriveReadyChanged [*IODefs*] const = IOErr + 33
IOEBSE [*EtherTypes*] const = #4
IOEFUZ [*EtherTypes*] const = #5
IOEIOC [*EtherTypes*] const = #0
IOEndOfFrame [*IODefs*] const = IOErr + 18
IOEndOfInput [*IODefs*] const = IOErr + 19
IOEPTL [*EtherTypes*] const = #2
IOERNT [*EtherTypes*] const = #3
IOErr [*IODefs*] const = IOBaseMsgID
IOETIM [*EtherTypes*] const = #1
IOEvent [*IODefs*] type = (IOReply, AsyncData, Attention, Distress, Acknowledge)
IOFramingError [*IODefs*] const = IOErr + 16
IOGetTime [*Clock*] function: long
IOHeaderCRCError [*IODefs*] const = IOErr + 30
IOIllegalCommand [*IODefs*] const = IOErr + 7
IOInvalidIOPort [*IODefs*] const = IOErr + 4
IOMessage [*IODefs*] type = record Head: Msg; Body: array[0..1023] of integer; end
IOMissingDataAddrMark [*IODefs*] const = IOErr + 25
IOMissingHeaderAddrMark [*IODefs*] const = IOErr + 26
IONoDataFound [*IODefs*] const = IOErr + 13
IONotEnoughData [*IODefs*] const = IOErr + 34
IONotEnoughRoom [*IODefs*] const = IOErr + 11
IOOverRun [*IODefs*] const = IOErr + 14
IOParityError [*IODefs*] const = IOErr + 15
IOSectorNotFound [*IODefs*] const = IOErr + 28
IOSenseStatusBlk [*IODefs*] type = packed record StatusCnt: Bit8; case integer of 1: (GPIBStatus: GPIBSenseStatus); 2: (SIOStatus: SIOSenseStatus); 3: (FloppyStatus: FloppyResultStatus); 0: (case integer of 1: (StatusByte: packed array[stretch(1)..stretch(1)] of Bit8); 2: (StatusWord: packed array[stretch(1)..stretch(1)] of integer); 3: (SByte: packed array[stretch(1)..stretch(12)] of Bit8)) end
IOStatusBlk [*IODefs*] type = packed record CmdIDTag: long; HardStatus: integer; SoftStatus: integer; CmdBytesTransferred: long; DataBytesTransferred: long; DeviceStatus: IOSenseStatusBlk; end
IOSuccess [*IODefs*] const = SUCCESS
IOTimeOut [*IODefs*] const = IOErr + 2
IOUndefinedError [*IODefs*] const = IOErr + 1
IOUndeterminedEquipFault [*IODefs*] const = IOErr + 23
IsChild [*AccentType*] const = AccErr + 36
IsParent [*AccentType*] const = AccErr + 35
IsPattern [*PMatch*] function(var str: pms255): boolean
IsQuotedChar [*PathName*] function(S: Wild_Path_Name; Index: integer): boolean
IsStreamDevice [*Stream*] function(S: SName): integer
KBFlushBoardOutput [*Stream*] procedure(var F: FileType)
KERNELPORT [*AccentType*] const = 1
KeyEvent [*SapphDefs*] type = record Cmd: 0..255; Ch: char; region: Integer; X, Y: integer; end
KeyHowWait [*SapphDefs*] type = (KeyWaitDiffPos, KeyDontWait, KeyWaitEvent)
KeyKind [*SapphFileDefs*] type = (Standard, Control, Mouse, NonStandard)
KeyMap [*SapphFileDefs*] type = packed record hashkey: KeyState; cmd: 0..255; chval: char; escty: EscKind; next: 0..MAXMAP; region: 0..WILDREGION; end
KeyState [*SapphFileDefs*] type = packed record case integer of 1: (code: 0..#177; cntrlon: boolean; special: boolean; EscCl: 0..#7; mbuttons: 0..#17); 2: (fullCode: 0..#777; escClAndMButtons: 0..#177); 3: (ks: integer); end

KeyTransTab [*SapphFileDefs*] type = packed record Version: 0..255; CmdState: 0..255; RawKeyBoard: boolean; EscState: 0..7; NumMaps: 0..MAXMAP; HashMask: integer; Len: integer; Class: packed array[0..MaxCode] of KeyKind; Map: packed array[1..1] of KeyMap; end
KeyTranVersion [*SapphFileDefs*] const = 3
KOEName [*CFileDefs*] function(x: Bit8): string
LandScapeBitHeight [*SapphDefs*] const = 1024
LandScapeBitWidth [*SapphDefs*] const = 1280
Language [*Code*] type = (Pascal, Fortran, Imp)
Language [*SegDefs*] type = (Pascal, Fortran, Imp)
LargeNumber [*Stream*] exception(FileName: SName)
LargeReal [*Stream*] exception(FileName: SName)
LastRecMsg [*IPCRecordIO*] var: RRecMsg
LibraryDef [*CFileDefs*] const = 90
LineFunct [*SapphDefs*] type = (DrawLine, EraseLine, XORLine)
LinkFileType [*RunDefs*] type = (SegFile, RunFile, DataFile, SymsFileType, QMapFileType)
LinkTypeStr [*ALoad*] function(typ: LinkFileType): string
ListLoggedInUsers [*Auth*] function(ServPort: Port; var UserList: Logged_User_List; var UserList_Cnt: long): GeneralReturn
LMsg [*AccCall*] type = record H: Msg; D: array[1..MAXMSGDATA] of integer; end
Ln [*RealFunctions*] function(X: Real): Real
LoadFont [*ViewPt*] function(ServPort: Viewport; fileName: VPStr255): Viewport
LoadVPCursors [*ViewPt*] function(ServPort: Viewport; fileName: VPStr255; var numCursors: integer): CursorSet
LoadVPPicture [*ViewPt*] function(ServPort: Viewport; fileName: VPStr255; width: integer; height: integer): Viewport
LocalLabel [*CFileDefs*] const = 6
LocalProcedure [*CFileDefs*] const = 2
LOCALPT [*AccentType*] const = 2
LockPorts [*AccCall*] function(LockThem: boolean; Ports: ptrLPortArray; PortsCount: long): GeneralReturn
Log10 [*RealFunctions*] function(X: Real): Real
Logged_User [*AuthDefs*] type = record UserID: User_ID; UserName: Auth_Var; MachineName: Auth_Var; End
Logged_User_Array [*AuthDefs*] type = array[0..0] of Logged_User
Logged_User_List [*AuthDefs*] type = ↑Logged_User_Array
LoginUser [*Auth*] function(ServPort: Port; UserName: Auth_Var; Password: Auth_Var; MachineName: Auth_Var; var UserAuthPort: Port; var UserRec: UserRecord): GeneralReturn
LOGMSGS [*AccCall*] const = false
LogoutUser [*Auth*] function(ServPort: Port): GeneralReturn
LogSmall [*RealFunctions*] exception(X: Real)
Lookup [*MsgN*] function(ServPort: Port; PortsName: string; var PortsID: Port): GeneralReturn
Lop [*Spice_String*] function(var Str: PString): PString
LPortArray [*AccentType*] type = array[stretch(0)..stretch(#77777)] of Port
LString [*CFileDefs*] type = String[255]
M_CHILDFORKREPLY [*AccentType*] const = #100 + #11
M_DEBUGMSG [*AccentType*] const = #100 + #12
M_GENERALKERNELREPLY [*AccentType*] const = #100 + 6
M_KERNELMSGERROR [*AccentType*] const = #100 + 7
M_MSGACCEPTED [*AccentType*] const = #100 + 2
M_OWNERSHIPRIGHTS [*AccentType*] const = #100 + 3
M_PARENTFORKREPLY [*AccentType*] const = #100 + #10
M_PORTDELETED [*AccentType*] const = #100 + 1
M_RECEIVERIGHTS [*AccentType*] const = #100 + 4
Machine_Name [*AuthDefs*] type = String[255]

MachineInfoRec [*BootInfo*] type = packed record case boolean of false: (int: integer); true: (WCSSize: 0..15; Reserved: 0..3; IsPortrait: Boolean; BoardRev: 0..31; OldZ80: boolean; CMUNet: boolean; Reserved2: 0..3) end
MAINKLUDGE [*RunDefs*] const = 255
MakeViewport [*ViewPt*] function(ServPort: Viewport; x: integer; y: integer; w: integer; h: integer; rank: integer; memory: boolean; courteous: boolean; transparent: boolean): Viewport
MakeWinListener [*Sapph*] procedure(ServPort: Window)
MapFull [*AccentType*] const = AccErr + 25
MAPNIL [*SapphFileDefs*] const = 0
Max_QmapRoutines [*QMapDefs*] const = 251
Max_SymRoutines [*SymDefs*] const = 251
Max_Users [*AuthDefs*] const = 1023
MAXBACKLOG [*AccentType*] const = 63
MaxCmndString [*CommandParse*] const = 255
MaxCode [*SapphFileDefs*] const = 127 + 16
MaxCoord [*SapphDefs*] const = 16000
MAXDEVICES [*AccentType*] const = 5
MAXDISKS [*AccentType*] const = 4
MAXDPCHARS [*AccentType*] const = 25
MaxEtherWords [*EtherTypes*] const = 748
MAXLOGMESS [*AccCall*] const = 20
MAXMAP [*SapphFileDefs*] const = 511
MAXMSGDATA [*AccCall*] const = 20
MaxNumRectangles [*SapphDefs*] const = 256 div 4
MAXPARTCHARS [*AccentType*] const = 8
MAXPARTITIONS [*AccentType*] const = 10
MAXPORTS [*AccentType*] const = 256
MAXPROCS [*AccentType*] const = 63
MaxPStringSize [*Spice_String*] const = 255
MaxPupWords [*EtherTypes*] const = 533
MaxSignal [*ProcMgrDefs*] const = SignalBase + 63
MaxSize [*SapphDefs*] const = 16000
MemFault [*AccentType*] const = AccErr + 6
MemProtection [*AccentType*] type = READONLY..READWRITE
MessagesWaiting [*AccCall*] function(MsgType: long; var Ports: ptrLPortArray; var PortsCount: long): GeneralReturn
MicroFailure [*AccentType*] const = AccErr + 41
MicroSeconds [*AccentType*] type = long
MILLIPERTIC [*SapphFileDefs*] const = 17
MinCoord [*SapphDefs*] const = - 16000
MinEtherWords [*EtherTypes*] const = WordSize(EtherHeader)
MinPupWords [*EtherTypes*] const = WordSize(PupHeader) + 1
MinSignal [*ProcMgrDefs*] const = SignalBase
ModifyRegion [*ViewPt*] procedure(ServPort: Viewport; regionNum: integer; leftx: integer; topy: integer; width: integer; height: integer)
ModifyVP [*ViewPt*] procedure(ServPort: Viewport; newlx: integer; newty: integer; newwidth: integer; newheight: integer; newrank: integer; wantVpChEx: boolean)
ModifyWindow [*Sapph*] procedure(ServPort: Window; newleftx: integer; newtopy: integer; newouterwidth: integer; newouterheight: integer; newRank: integer)
ModNameLength [*RunDefs*] const = 19
ModString [*RunDefs*] type = String[ModNameLength]
MoveWords [*AccCall*] function(SrcAddr: VirtualAddress; var DstAddr: VirtualAddress; NumWords: long; Delete: boolean; Create: boolean; Mask: long; DontShare: boolean): GeneralReturn
MParity [*Except*] exception

Msg [*AccentType*] type = record SimpleMsg: boolean; MsgSize: long; MsgType: long; LocalPort: Port; RemotePort: Port; ID: long; end

MsgClrEtherFilter [*EtherTypes*] type = RECORD Head: Msg; TTyp: TypeType; Typ: INTEGER; TProcessPort: TypeType; ProcessPort: Port END

MsgClrPupFilter [*EtherTypes*] type = RECORD Head: Msg; TSocket: TypeType; Socket: Long; TProcessPort: TypeType; ProcessPort: Port END

MsgEtherPacket [*EtherTypes*] type = RECORD Head: Msg; TPacket: TypeType; Packet: EtherPacket; END

MsgGetEtherAddress [*EtherTypes*] type = RECORD Head: Msg END

MsgIndex [*PascalInit*] const = 5

MsgInterrupt [*AccentType*] const = AccErr + 43

MsgLog [*AccCall*] type = record Init: integer; MsgsSent: long; MsgsRec: long; NxtMsg: integer; LMsgs: array[0..MAXLOGMESS] of record Sent: boolean; InProg: boolean; GR: GeneralReturn; M: LMsg; end; end

MsgPacket [*EtherTypes*] type = RECORD Head: Msg; TPacket: TypeType; END

MsgPortStatus [*MsgN*] function(ServPort: Port; PortsID: Port; var GlobalPort: long; var Owner: long; var Receiver: long; var SrcID: long; var SeqNum: long; var NetWaiting: boolean; var NumQueued: integer; var Blocked: boolean; var Locked: boolean; var RecvQueue: integer; var DataOffset: long; var InSrcID: long; var InSeqNum: long): GeneralReturn

MsgPupPacket [*EtherTypes*] type = RECORD Head: Msg; TPacket: TypeType; Packet: PupPacket; END

MsgRClrEtherFilter [*EtherTypes*] type = RECORD Head: Msg; TTyp: TypeType; Typ: INTEGER; TAnswer: TypeType; Answer: BOOLEAN END

MsgRClrPupFilter [*EtherTypes*] type = RECORD Head: Msg; TSocket: TypeType; Socket: Long; TAnswer: TypeType; Answer: Boolean END

MsgRecvEtherPacket [*EtherTypes*] type = MsgEtherPacket

MsgRecvPupPacket [*EtherTypes*] type = MsgPupPacket

MsgRGetEtherAddress [*EtherTypes*] type = RECORD Head: Msg; TEtherAddress: TypeType; EtherAddress: INTEGER END

MsgRSetEtherFilter [*EtherTypes*] type = RECORD Head: Msg; TTyp: TypeType; Typ: INTEGER; TAnswer: TypeType; Answer: BOOLEAN END

MsgRSetPupFilter [*EtherTypes*] type = RECORD Head: Msg; TSocket: TypeType; Socket: Long; TAnswer: TypeType; Answer: Boolean END

MsgSendEtherPacket [*EtherTypes*] type = MsgEtherPacket

MsgSendPupPacket [*EtherTypes*] type = MsgPupPacket

MsgSetEtherFilter [*EtherTypes*] type = RECORD Head: Msg; TTyp: TypeType; Typ: INTEGER; TProcessPort: TypeType; ProcessPort: Port END

MsgSetPupFilter [*EtherTypes*] type = RECORD Head: Msg; TSocket: TypeType; Socket: Long; TProcessPort: TypeType; ProcessPort: Port END

MsgTooBig [*AccentType*] const = AccErr + 20

MulOvfl [*Except*] exception

MultiLevelProgress [*WindowUtils*] procedure(Level: integer; Current, Max: Long)

MultiStreamProgress [*WindowUtils*] procedure(Level: integer; var F: File)

Name_Flags [*SesameDefs*] type = 0..#3

Name_Status [*SesameDefs*] type = 0..#7

NameAmbiguous [*ProcMgrDefs*] const = ProcMgrBase + 8

NameBase [*NameErrors*] const = 1000

NameNotCheckedIn [*NameErrors*] const = NameBase + 1

NameNotFound [*SesameDefs*] const = Sesame_Error_Base + 1

NameNotYours [*NameErrors*] const = NameBase + 0

NameServerPort [*PascalInit*] var: port

Net10MBRecvServer [*Net10MBRecvServer*] function(InP, RepP: POINTER): boolean

Net_Version [*Net10MB*] function(ServPort: E10Port): String

NetFail [*AccentType*] const = AccErr + 11

NETWORKTROUBLE [*AccentType*] const = 2

NewP [*Dynamic*] procedure(S: HeapNumber; A: integer; var Where: pointer; L: integer)
NewToOldTime [*OldTimeStamp*] function(NewTime: Internal_Time): TimeStamp
NextExtension [*PathName*] function(var EList: Extension_List): string
NextOp [*Except*] exception
NFlag_Deleted [*SesameDefs*] const = #000001
NFlag_NoNormal [*SesameDefs*] const = #000002
NFlag_RESERVED [*SesameDefs*] const = #177774
NilPointer [*Dynamic*] exception
No_User [*AuthDefs*] const = 0
NoAccess [*SesameDefs*] const = Sesame_Error_Base + 9
NoAvailablePages [*AccentType*] const = AccErr + 37
NoChildren [*ProcMgrDefs*] const = ProcMgrBase + 6
NoMorePorts [*AccentType*] const = AccErr + 9
NOREPLY [*AccentType*] const = 4
NORMALMSG [*AccentType*] const = 0
NormMsg [*Except*] exception
NotADirectory [*SesameDefs*] const = Sesame_Error_Base + 12
NotAFile [*SesameDefs*] const = Sesame_Error_Base + 8
NotAFont [*AccentType*] const = AccErr + 58
NotAnIPCCall [*AccentType*] const = AccErr + 17
NotAPort [*AccentType*] const = AccErr + 7
NotASystemAddress [*AccentType*] const = AccErr + 50
NotAUserAddress [*AccentType*] const = AccErr + 51
NotBoolean [*Stream*] exception(FileName: SName)
NotCurrentProcess [*AccentType*] const = AccErr + 28
NotEnoughRoom [*AccentType*] const = AccErr + 16
NotIdentifier [*Stream*] exception(FileName: SName)
NotNumber [*Stream*] exception(FileName: SName)
NotOpen [*Stream*] exception
NotPortReceiver [*AccentType*] const = AccErr + 14
NotReal [*Stream*] exception(FileName: SName)
NotReset [*Stream*] exception(FileName: SName)
NotRewrite [*Stream*] exception(FileName: SName)
NotSamePartition [*SesameDefs*] const = Sesame_Error_Base + 10
NotTextFile [*Stream*] exception(FileName: SName)
NotYourChild [*AccentType*] const = AccErr + 21
NStat_Deleted [*SesameDefs*] const = #000001
NStat_High [*SesameDefs*] const = #000002
NStat_Low [*SesameDefs*] const = #000004
NStat_RESERVED [*SesameDefs*] const = #177770
NTIMERS [*SapphFileDefs*] const = 20
Null_CommandBlock [*CommandDefs*] function: CommandBlock
NULLPORT [*AccentType*] const = 0
NULLViewPort [*SapphDefs*] const = NullPort
NullWindow [*SapphDefs*] const = NullPort
NUMMSGTYPES [*AccentType*] const = 2
NUMPRIORITIES [*AccentType*] const = 16
NumProgressBars [*SapphDefs*] const = 2
NUMQUEUES [*AccentType*] const = NUMSLEEPQS + NUMPRIORITIES + 5
NUMSLEEPQS [*AccentType*] const = 32
OFFSCREEN [*SapphDefs*] const = -32002
OFFSETBASE [*CFileDefs*] const = 16000
OffsetDef [*CFileDefs*] const = 11

OldCurrentTime [*OldTimeStamp*] function: TimeStamp
OldToNewTime [*OldTimeStamp*] function(OldTime: TimeStamp): Internal_Time
OpenCmdFile [*CommandParse*] function(FileName: pWord_String; var inF: pCommand_File_List): GeneralReturn
OpenIO [*IO*] function(ServPort: ServerNamePort; var IOPort: ServerIOPort; UserPort: UserEventPort): GeneralReturn
Other [*AccentType*] const = AccErr + 13
OutOfImagSegments [*AccentType*] const = AccErr + 49
OutOfIPCSpace [*AccentType*] const = AccErr + 23
OutOfRectangleBounds [*AccentType*] const = AccErr + 54
OUTREGION [*SapphDefs*] const = 0
OvfILI [*Except*] exception
OvrReal [*Except*] exception
Owner_Read_Access [*SesDiskDefs*] const = #1
Owner_Write_Access [*SesDiskDefs*] const = #2
PacketBytes [*EtherTypes*] const = 2*WordSize(MsgPacket)
Pad [*Spice_String*] function(Str: PString; TotalLen: integer; PadCh: Char; Where: integer): Pstring
PAGEBITS [*AccentType*] const = 8
PAGEBYTESIZE [*AccentType*] const = 512
PAGEWORDSIZE [*AccentType*] const = PAGEBYTESIZE div 2
ParseChPool [*CommandParse*] function(ChPool: pCharacter_Pool; PoolLength: Char_Pool_Index; var inputs: pCommand_Word_List; var outputs: pCommand_Word_List; var switches: pCommand_Word_List): GeneralReturn
ParseCommand [*CommandParse*] function(var inputs: pCommand_Word_List; var outputs: pCommand_Word_List; var switches: pCommand_Word_List): GeneralReturn
ParseEtherAddress [*EtherUser*] function(MsgP: pMsgRGetEtherAddress): INTEGER
ParseEtherClear [*EtherUser*] function(MsgP: pMsgRClrEtherFilter; VAR Typ: INTEGER): BOOLEAN
ParseEtherFilter [*EtherUser*] function(MsgP: pMsgRSetEtherFilter; VAR Typ: INTEGER): BOOLEAN
ParseEtherPacket [*EtherUser*] function(MsgP: pMsgRecvEtherPacket; Packet: POINTER): INTEGER
ParseIllegalCharInEnvNam [*CommandDefs*] const = CmdParse_Error_Base + 19
ParseIllegalCharInInRed [*CommandDefs*] const = CmdParse_Error_Base + 22
ParseIllegalCharInOutRed [*CommandDefs*] const = CmdParse_Error_Base + 23
ParseIllegalCharInQuoted [*CommandDefs*] const = CmdParse_Error_Base + 20
ParseIllegalCharInShPara [*CommandDefs*] const = CmdParse_Error_Base + 24
ParseIllegalCharInSwName [*CommandDefs*] const = CmdParse_Error_Base + 17
ParseIllegalCharInSwVal [*CommandDefs*] const = CmdParse_Error_Base + 18
ParseInternalFault [*CommandDefs*] const = CmdParse_Error_Base + 15
ParseOnlyCmdAllowed [*CommandDefs*] const = CmdParse_Error_Base + 21
ParsePupClear [*EtherUser*] function(MsgP: pMsgRClrPupFilter; VAR Socket: Long): BOOLEAN
ParsePupFilter [*EtherUser*] function(MsgP: pMsgRSetPupFilter; VAR Socket: Long): BOOLEAN
ParsePupPacket [*EtherUser*] procedure(MsgP: pMsgRecvPupPacket; PupPacket: POINTER)
ParseWordTooLong [*CommandDefs*] const = CmdParse_Error_Base + 16
PartData [*SesDiskDefs*] type = record PartName: DevPartString; DevName: DevPartString; PartRootDir: SegID; PartNumFree: long; PartMounted: boolean; PartStart: DiskAddr; PartEnd: DiskAddr; PartKind: PartitionType; PartExUse: boolean; end
PartDisMount [*AccInt*] function(ServPort: port): GeneralReturn
PartDList [*SesDiskDefs*] type = array[1..MAXPARTITIONS] of PartData
PartInfo [*AccentType*] type = record PartHeadFree: DiskAddr; PartTailFree: DiskAddr; PartInfoBlk: DiskAddr; PartRootDir: SegID; PartNumOps: integer; PartNumFree: long; PartInUse: boolean; PartMounted: boolean; PartDevice: integer; PartStart: DiskAddr; PartEnd: DiskAddr; PartKind: PartitionType; PartName: PartString; PartExUse: boolean; Unused: long; PartDiskRel: boolean; end
PartitionFull [*AccentType*] const = AccErr + 59

PartitionType [*AccentType*] type = (Root, UnUsed, Segment, PLX)
PartList [*AccentType*] type = array[1..MAXPARTITIONS] of PartInfo
PartMount [*AccInt*] function(ServPort: port; PartName: DevPartString; ExUse: Boolean; var RootId: SegID; var PartKind: PartitionType; var PartPort: port; var PartS: DiskAddr; var PartE: DiskAddr): GeneralReturn
PartString [*AccentType*] type = string[MAXPARTCHARS]
PascalProcedure [*CFileDefs*] const = 1
PASCALRECORD [*IPCRecordIO*] const = #12345
PassType [*AuthDefs*] type = Long
PassWordIncorrect [*AuthDefs*] const = Auth_Error_Base+2
PastEof [*Stream*] exception(FileName: SName)
Path_Name [*PathName*] type = string[Path_Name_Size]
Path_Name_Size [*SesameDefs*] const = 255
PattDebug [*PMatch*] procedure(v: boolean)
Pattern [*SapphFileDefs*] type = ↑PatternMap
PatternMap [*SapphFileDefs*] type = array[0..63, 0..3] of integer
PattMap [*PMatch*] function(var str, inpatt, outpatt, outstr: pms255; fold: boolean): boolean
PattMatch [*PMatch*] function(var str, pattern: pms255; fold: boolean): boolean
pBit32 [*AccentType*] type = ↑Bit32
pCharacter_Pool [*CommandDefs*] type = ↑Character_Pool
PCInData [*CFileDefs*] const = DataState
PCInData2 [*CFileDefs*] const = Data2State
PCInText [*CFileDefs*] const = TextState
pCmnd_String [*CommandParse*] type = ↑Cmnd_String
pCommand_File_List [*CommandParse*] type = ↑Command_File_List
pCommand_Word_List [*CommandParse*] type = ↑Command_Word_List
PCStateName [*CFileDefs*] function(x: Bit8): string
pCursorArrayRec [*SapphFileDefs*] type = ↑CursorArrayRec
pE10Message [*Ether10Defs*] type = ↑E10Message
pE10Packet [*Ether10Defs*] type = ↑E10Packet
pEtherPacket [*EtherTypes*] type = ↑EtherPacket
pFEarray [*RunDefs*] type = ↑FEarray
pFirstBlock [*CFileDefs*] type = ↑FirstBlock
pFNString [*Code*] type = ↑FNString
PhysicalAddress [*AccentType*] type = long
pImpArray [*RunDefs*] type = ↑ImpArray
pImpInfo [*SegDefs*] type = ↑CImpInfo
pImpNode [*Code*] type = ↑ImpNode
pIOCmdBlk [*IODefs*] type = ↑IOCmdBlk
pIOMessage [*IODefs*] type = ↑IOMessage
pKeyTab [*SapphFileDefs*] type = ↑KeyTransTab
pLMsg [*AccCall*] type = ↑LMsg
PMAddCtlWindow [*ProcMgr*] function(ServPort: port; CtlWindow: window; NewCtlWindow: window): GeneralReturn
PMBroadcast [*ProcMgr*] function(ServPort: port; s: string): GeneralReturn
PMChangeGroup [*ProcMgr*] function(ServPort: port; ProcPort: port; NewWindow: window): GeneralReturn
PMDebug [*ProcMgr*] function(ServPort: port; ProcID: string): GeneralReturn
PMDebugProcess [*ProcMgr*] function(ServPort: port; ProcPort: port; Reason: long): GeneralReturn
PMGetProcPorts [*ProcMgr*] function(ServPort: port; ProcPort: port; var hisWindow: window; var hisTypescript: typescript; var hisEMConn: port): GeneralReturn
PMGetStatus [*ProcMgr*] function(ServPort: port; ProcID: string; var Stats: StatList; var Stats_Cnt: long): GeneralReturn

PMGetTimes [*ProcMgr*] function(ServPort: port; ProcPort: port; var LoadTime: long; var RunTime: long; var ElapsedTime: long): GeneralReturn
PMGetWaitID [*ProcMgr*] function(ServPort: port; ProcPort: port; var WaitID: long): GeneralReturn
PMGroupSignal [*ProcMgr*] procedure(ServPort: port; CtlWindow: window; Signal: SignalName)
PMIndex [*PascalInit*] const = 4
PMKill [*ProcMgr*] function(ServPort: port; ProcID: string): GeneralReturn
PMPort [*PascalInit*] var: port
PMProcessSignal [*ProcMgr*] procedure(ServPort: port; ProcPort: port; Signal: SignalName)
PMRegisterProcess [*ProcMgr*] function(ServPort: port; HisKPort: port; HisDPort: port; ProgName: string; HisWindow: window; HisTypescript: typescript; EMConn: port; Parent: port): GeneralReturn
PMRemoveCtlWindow [*ProcMgr*] function(ServPort: port; CtlWindow: window): GeneralReturn
PMResume [*ProcMgr*] function(ServPort: port; ProcID: string): GeneralReturn
pms255 [*PMatch*] type = String[255]
PMSaveLoadTime [*ProcMgr*] function(ServPort: port; ProcPort: port; LoadTime: long): GeneralReturn
PMSetDebugPort [*ProcMgr*] function(ServPort: port; ProcPort: port; DebugPort: port; DebugSignalOnly: boolean): GeneralReturn
PMSetPriority [*ProcMgr*] function(ServPort: port; ProcID: string; priority: integer): GeneralReturn
PMSetSignal [*ProcMgr*] function(ServPort: port; ProcPort: port; Signal: SignalName; Action: SignalAction): GeneralReturn
PMSetSignalPort [*ProcMgr*] function(ServPort: port; ProcPort: port; SignalPort: port): GeneralReturn
pMsgClrEtherFilter [*EtherTypes*] type = ↑MsgClrEtherFilter
pMsgClrPupFilter [*EtherTypes*] type = ↑MsgClrPupFilter
pMsgGetEtherAddress [*EtherTypes*] type = ↑MsgGetEtherAddress
pMsgLog [*AccCall*] type = ↑MsgLog
pMsgRClrEtherFilter [*EtherTypes*] type = ↑MsgRClrEtherFilter
pMsgRClrPupFilter [*EtherTypes*] type = ↑MsgRClrPupFilter
pMsgRecvEtherPacket [*EtherTypes*] type = ↑MsgRecvEtherPacket
pMsgRecvPupPacket [*EtherTypes*] type = ↑MsgRecvPupPacket
pMsgRGetEtherAddress [*EtherTypes*] type = ↑MsgRGetEtherAddress
pMsgRSetEtherFilter [*EtherTypes*] type = ↑MsgRSetEtherFilter
pMsgRSetPupFilter [*EtherTypes*] type = ↑MsgRSetPupFilter
pMsgSendEtherPacket [*EtherTypes*] type = ↑MsgSendEtherPacket
pMsgSendPupPacket [*EtherTypes*] type = ↑MsgSendPupPacket
pMsgSetEtherFilter [*EtherTypes*] type = ↑MsgSetEtherFilter
pMsgSetPupFilter [*EtherTypes*] type = ↑MsgSetPupFilter
PMSuspend [*ProcMgr*] function(ServPort: port; ProcID: string): GeneralReturn
PMTerminate [*ProcMgr*] function(ServPort: port; ProcPort: port; Reason: long): GeneralReturn
Port [*AccentType*] type = long
PortArray [*AccentType*] type = array[0..MAXPORTS – 1] of Port
PortBitArray [*AccentType*] type = packed array[0..MAXPORTS – 1] of boolean
PortDeath [*AccentType*] type = EXPLICITDEALLOC..NETWORKTROUBLE
PortFull [*AccentType*] const = AccErr + 3
PortInterpose [*AccInt*] function(ServPort: port; MyPort: Port; HisPort: port; var MyNewPort: Port): GeneralReturn
PortOption [*AccentType*] type = DEFAULTPTS..LOCALPT
PortraitBitHeight [*SapphDefs*] const = 1024
PortraitBitWidth [*SapphDefs*] const = 768
PortsWithMessages [*AccCall*] function(MsgType: long; var ports: PortBitArray): GeneralReturn
PosC [*Spice_String*] function(Str: PString; c: char): integer
PosString [*Spice_String*] function(Source, Mask: PString): Integer
PosSystem [*SysType*] const = false

Power [*RealFunctions*] function(X, Y: Real): Real
PowerBig [*RealFunctions*] exception(X, Y: Real)
PowerI [*RealFunctions*] function(X: Real; Y: Integer): Real
PowerNeg [*RealFunctions*] exception(X, Y: Real)
PowerSmall [*RealFunctions*] exception(X, Y: Real)
PowerZero [*RealFunctions*] exception(X, Y: Real)
pPupData [*EtherTypes*] type = ↑PupData
pPupPacket [*EtherTypes*] type = ↑PupPacket
pQMap [*QMapDefs*] type = ↑QCodeMap
pQMap_Buffer [*QMapDefs*] type = ↑QMap_Buffer
pQMap_ByteBuffer [*QMapDefs*] type = ↑QMap_ByteBuffer
pQMapDict [*QMapDefs*] type = ↑QMapDict
PReadIn [*Stream*] procedure(var F: Filetype)
pRectArray [*SapphDefs*] type = ↑RectArray
PREVIEW [*AccentType*] const = 0
PrimaryDef [*CFileDefs*] const = 10
Print_Name [*SesameDefs*] type = string[Print_Name_Size]
Print_Name_Size [*SesameDefs*] const = 80
PriorID [*AccentType*] type = 0..NUMPRIORITIES - 1
PROCESSDEATH [*AccentType*] const = 1
ProcessDeathMsg [*ProcMgrDefs*] type = record Head: Msg; tWaitID: TypeType; WaitID: long; tReason: TypeType; Reason: long; tLoadTime: TypeType; LoadTime: long; tRunTime: TypeType; RunTime: long; tElapsedTime: TypeType; ElapsedTime: long; end
ProcessDeathMsgID [*ProcMgrDefs*] const = 3800
ProcessDisowned [*ProcMgrDefs*] const = ProcMgrBase + 9
ProcID [*AccentType*] type = integer
ProcMgr_Version [*ProcMgr*] function(ServPort: port): string
ProcMgrBase [*ProcMgrDefs*] const = 3600
ProcState [*AccentType*] type = (Supervisor, Privileged, BadSupervisor, User)
ProgressInTitle [*SapphDefs*] const = 1
ProgStr [*SapphDefs*] type = String[ProgStrLength]
ProgStrLength [*SapphDefs*] const = (IconWidth - 2*BorderOverhead) div SysFontWidth
Prompt_Indention_String [*ExtraCmdParse*] const = ".
pRunHead [*RunDefs*] type = ↑RunHead
pSEarray [*RunDefs*] type = ↑SEarray
pSegBlock [*Code*] type = ↑SegBlock
pSegBlock [*SegDefs*] type = ↑SegBlock
pSegNode [*Code*] type = ↑SegNode
pSourceMap [*QMapDefs*] type = ↑SourceMap
PStatus [*AccentType*] type = record State: ProcState; Priority: PriorID; MsgPending: boolean; EMsgPending: boolean; MsgEnable: boolean; EMsgEnable: boolean; LimitSet: boolean; SVStkInCore: boolean; QueueID: QID; SleepID: ptrInteger; RunTime: long; LimitTime: long end
pStreamBuffer [*Stream*] type = ↑StreamBuffer
pStreamF [*Stream*] type = ↑StreamF
PString [*Spice_String*] type = String[MaxPStringSize]
ptrAllPortArray [*AccentType*] type = ↑PortArray
ptrArguments [*AccentType*] type = ↑Arguments
ptrBIRecord [*BootInfo*] type = ↑BIRecord
ptrBoolean [*AccentType*] type = ↑boolean
ptrDirIOArgs [*AccentType*] type = ↑DirectioArgs
ptrInteger [*AccentType*] type = ↑integer
ptrLPortArray [*AccentType*] type = ↑LPortArray
ptrMsg [*AccentType*] type = ↑Msg

ptrPartDList [*SesDiskDefs*] type = ↑PartDList
ptrPartList [*AccentType*] type = ↑PartList
ptrPort [*AccentType*] type = ↑Port
ptrPortArray [*AccentType*] type = ↑PortArray
ptrPortBitArray [*AccentType*] type = ↑PortBitArray
pTSCharArray [*TSDefs*] type = ↑TSCharArray
PupCMUNet [*EtherTypes*] const = #52
PupData [*EtherTypes*] type = RECORD CASE INTEGER OF 0: (Chars: PACKED ARRAY[0..1043] OF CHAR); 1: (Bytes: PACKED ARRAY[0..1043] OF 0..255); 2: (Words: PACKED ARRAY[0..521] OF INTEGER); 3: (HLongs: PACKED ARRAY[0..260] OF PupHLong); 4: (Ports: PACKED ARRAY[0..260] OF PupPort); END
PupEFTPAck [*EtherTypes*] const = #31
PupEFTPBort [*EtherTypes*] const = #33
PupEFTPData [*EtherTypes*] const = #30
PupEFTPEnd [*EtherTypes*] const = #32
PupEFTPSocket [*EtherTypes*] const = #20*#200000
PupError [*EtherTypes*] const = #4
PupHeader [*EtherTypes*] type = PACKED RECORD Len: INTEGER; Typ: 0..255; TC: 0..255; Id: PupHLong; Dst: PupPort; Src: PupPort END
PupHLong [*EtherTypes*] type = RECORD CASE INTEGER OF 1: (Lng: Long); 2: (Hgh: Integer; Low: Integer); END
PupLLong [*EtherTypes*] type = RECORD CASE INTEGER OF 1: (Lng: Long); 2: (Low: Integer; Hgh: Integer) END
PupNameSocket [*EtherTypes*] const = #4*#200000
PupPacket [*EtherTypes*] type = RECORD CASE INTEGER OF 0: (Header: PupHeader; Data: PupData); 1: (ChkSums: ARRAY[0..532] OF Integer) END
PupPacketBytes [*EtherTypes*] const = 2*WordSize(MsgPupPacket)
PupPort [*EtherTypes*] type = PACKED RECORD CASE integer OF 0: (Host: 0..255; Net: 0..255; Soc: PupHLong); 1: (All: PACKED ARRAY[0..5] OF Char) END
PushRegion [*ViewPt*] procedure(ServPort: Viewport; regionNum: integer; leftx: integer; topy: integer; width: integer; height: integer)
PutAbsoluteDef [*CFileDefs*] procedure(var BytePtr: long; ByteAddress: long; Name: LString)
PutAbsoluteLocalDef [*CFileDefs*] procedure(var BytePtr: long; ByteAddress: long; Name: LString)
PutB [*Stream*] procedure(var F: Filetype)
PutC [*Stream*] procedure(var F: FileType)
PutChainHead [*CFileDefs*] procedure(var BytePtr: long; AreaDesg: Bit8; ByteAreaOff: long)
PutLibraryDef [*CFileDefs*] procedure(var BytePtr: long; LibraryFileName: LString; LibraryTimeStamp: Internal_Time; LibraryLoc: long)
PutOffsetDef [*CFileDefs*] procedure(var BytePtr: long; WdOffsetAmt: long)
PutPrimaryDef [*CFileDefs*] procedure(var BytePtr: long; SimKind: Bit8; AreaDesg: Bit8; AreaOffset: long; SymSize: long; SymName: LString)
PutViewportBit [*ViewPt*] procedure(ServPort: Viewport; x: integer; y: integer; value: boolean)
PutViewportRectangle [*ViewPt*] procedure(ServPort: Viewport; Funct: RopFunct; x: integer; y: integer; width: integer; height: integer; Data: pVPIntegerArray; Data_Cnt: long; WordsAcross: integer; ux: integer; uy: integer)
pVar_CharAr [*SymDefs*] type = ↑Var_CharAr
pVar_IntAr [*SymDefs*] type = ↑Var_IntAr
pVarDictEntry [*SymDefs*] type = ↑VarDictEntry
pVPCharArray [*SapphDefs*] type = ↑VPCharArray
pVPIntegerArray [*SapphDefs*] type = ↑VPIntegerArray
pVPPortArray [*SapphDefs*] type = ↑VPPortArray
pWinNameArray [*SapphDefs*] type = ↑WinNameArray
pWord_Search_Table [*CommandParse*] type = POINTER
pWord_String [*CommandParse*] type = ↑Word_String
PWriteln [*Stream*] procedure(var F: Filetype)

QCodeMap [*QMapDefs*] type = Array[0..QMap_entries_per_block – 1] of QMapInfoRecord
QCodeVersion [*Code*] const = 4
QCodeVersion [*SegDefs*] const = 4
QID [*AccentType*] type = 0..NUMQUEUES
QMap_Buffer [*QMapDefs*] type = array[0..PageWordSize – 1] of integer
QMap_ByteBuffer [*QMapDefs*] type = array[0..PageByteSize – 1] of bit8
QMap_entries_per_block [*QMapDefs*] const = 256 div WSQMapInfoRecord
QMap_ptr_type [*QMapDefs*] type = record case integer of 0: (P: POINTER); 1: (Buffer: pQMap_Buffer); 2: (ByteBuffer: pQMap_ByteBuffer); 2: (p: pDirBlk); 2: (pDict: pQMapDict); 3: (pMap: pQMap); 4: (pSource: pSourceMap); end
QMapDict [*QMapDefs*] type = Record CompID: Internal_Time; SourceFileBlock: Integer; Routines: array[0..Max_QmapRoutines] of integer; CompID: TimeStamp; SourceFileBlock: Integer; Routines: array[0..252] of integer; End
QMapInfoRecord [*QMapDefs*] type = packed record QCodeNumber: integer; SourceLineNumber: integer; BlockNumber: integer; Offset: 0..511; SourceFileNumber: 0..127; end
QuitMultiProgress [*WindowUtils*] procedure(Level: integer)
QuitProgress [*WindowUtils*] procedure
Quote_Char [*SesameDefs*] const = '\'
quoted_text_bracket_char [*CommandParse*] const = ''''
QVerRange [*Code*] type = 0..255
QVerRange [*SegDefs*] type = 0..255
RaiseP [*Except*] procedure(ES, ER, PStart, PEnd: Integer)
RandomProgress [*WindowUtils*] procedure
ReadBoolean [*Reader*] procedure(var F: FileType; var X: boolean)
ReadCh [*Reader*] procedure(var F: FileType; var X: char; Field: integer)
ReadChArray [*Reader*] procedure(var F: FileType; var X: ChArray; Max, Len: integer)
ReadD [*PasLong*] procedure(var F: FileType; var X: long; B: integer)
ReadExtendedFile [*PathName*] function(var PathName: Path_Name; ExtensionList: Extension_List; ImplicitSearchList: Env_Var_Name; var Data: File_Data; var ByteCount: long): GeneralReturn
ReadFile [*PathName*] function(var PathName: Path_Name; var Data: File_Data; var ByteCount: long): GeneralReturn
ReadIdentifier [*Reader*] procedure(var F: FileType; var X: integer; var IT: IdentTable; L: integer)
ReadInteger [*Reader*] procedure(var F: FileType; var X: integer)
READONLY [*AccentType*] const = 0
ReadProcessMemory [*AccInt*] function(ServPort: port; Address: VirtualAddress; NumBytes: Long; var Data: Pointer; var Data_Cnt: long): GeneralReturn
ReadR [*PasReal*] procedure(var F: FileType; var value: real)
ReadSegment [*AccInt*] function(ServPort: port; Segment: SegID; Offset: Integer; NumPages: Integer; var Data: Pointer; var Data_Cnt: long): GeneralReturn
ReadString [*Reader*] procedure(var F: FileType; var X: String; Max, Len: integer)
READWRITE [*AccentType*] const = 1
ReadX [*Reader*] procedure(var F: FileType; var X: integer; B: integer)
RealDivZero [*Except*] exception
RealMIndefinite [*RealFunctions*] const = Recast(#20000000001, Real)
RealMInfinity [*RealFunctions*] const = Recast(#37740000000, Real)
RealMLargest [*RealFunctions*] const = Recast(#37737777777, Real)
RealMSmallest [*RealFunctions*] const = Recast(#20040000000, Real)
RealPIndefinite [*RealFunctions*] const = Recast(#00000000001, Real)
RealPInfinity [*RealFunctions*] const = Recast(#17740000000, Real)
RealPLargest [*RealFunctions*] const = Recast(#17737777777, Real)
RealPSmallest [*RealFunctions*] const = Recast(#00040000000, Real)
RealWriteError [*Stream*] exception(FileName: SName)
Receive [*AccCall*] function(var xxmsg: Msg; MaxWait: long; PortOpt: PortOption; Option: ReceiveOption): GeneralReturn

RECEIVEIT [*AccentType*] const = 1
ReceiveOption [*AccentType*] type = PREVIEW..RECEIVEWAIT
RECEIVEWAIT [*AccentType*] const = 2
RecMsg [*IPCRecordIO*] type = record Head: Msg; RecType: TypeType; RecName: integer; RecSize: integer; RecElts: long; RecPtr: pointer end
RECMSGSIZE [*IPCRecordIO*] const = WordSize(RecMsg)*2
RecRecord [*IPCRecordIO*] function(var localport: Port; var remoteport: Port; var id: long; var MsgType: long; var recptr: Pointer; var recsize: integer): GeneralReturn
Rectangle [*SapphDefs*] type = record lx, ty, w, h: integer; end
RectArray [*SapphDefs*] type = Array[1..MaxNumRectangles] of Rectangle
RectColor [*AccCall*] function(Rectangle: Port; Action: integer; X: integer; Y: integer; Width: integer; Height: integer): GeneralReturn
RectColorFunct [*SapphDefs*] type = (RectBlack, RectWhite, RectInvert)
RectDrawLine [*AccCall*] function(DstRectangle: Port; Kind: integer; X1, Y1, X2, Y2: integer): GeneralReturn
RectPutString [*AccCall*] function(DstRectangle: Port; FontRectangle: Port; Action: integer; var FirstX: integer; var FirstY: integer; StrPtr: Pointer; FirstChar: integer; var MaxChar: integer): GeneralReturn
RectRasterOp [*AccCall*] function(DstRectangle: Port; Action: integer; DstX: integer; DstY: integer; Width: integer; Height: integer; SrcRectangle: Port; SrcX: integer; SrcY: integer): GeneralReturn
RectScroll [*AccCall*] function(Rectangle: Port; X: integer; Y: integer; Width: integer; Height: integer; Xamt: integer; Yamt: integer): GeneralReturn
RemoveExtension [*PathName*] procedure(var fileName: Path_Name; Extension: String)
RemoveWindow [*Sapph*] procedure(ServPort: Window)
RemoveWindowAttentionFlag [*WindowUtils*] procedure
RemoveWindowErrorFlag [*WindowUtils*] procedure
RemoveWindowRequestFlag [*WindowUtils*] procedure
ReplaceChars [*Spice_String*] procedure(var Str: PString; NewS: PString; Index: integer)
REPLY [*AccentType*] const = 2
ReplyCode [*EtherTypes*] const = 1
ReserveCursor [*ViewPt*] procedure(ServPort: Viewport; reserve: boolean)
ReserveScreen [*ViewPt*] procedure(ServPort: Viewport; reserve: boolean)
ResetError [*Stream*] exception(FileName: SName)
ResetHeap [*Dynamic*] procedure(S: HeapNumber)
ResolveSearchList [*EnvMgr*] function(ServPort: Port; Name: Env_Var_Name; FirstOnly: boolean; var Variable: Env_Variable; var Variable_Cnt: long; var FirstDefined: boolean): GeneralReturn
RestoreWindow [*Sapph*] procedure(ServPort: Window)
Resume [*AccInt*] function(ServPort: port): GeneralReturn
RevPosC [*Spice_String*] function(Str: PString; c: char): integer
RevPosString [*Spice_String*] function(Source, Mask: PString): Integer
RewriteError [*Stream*] exception(FileName: SName)
RFDELAYEDMAIN [*RunDefs*] const = 255
RFFORMAT [*RunDefs*] const = 5
RFileFormat [*Code*] const = 2
RMAXFILE [*RunDefs*] const = 255
RMAXIMPORT [*RunDefs*] const = 4095
RMAXSEG [*RunDefs*] const = RFDELAYEDMAIN – 1
RopFunct [*SapphDefs*] type = (RRpl, RNot, RAnd, RAndNot, ROr, ROrNot, RXor, RXNor)
RoundTo [*CFileDefs*] procedure(var Value: long; Boundary: integer)
RRecMsg [*IPCRecordIO*] type = record Head: Msg; RecType: TypeType; RecName: integer; RecSize: integer; RecElts: long; RecPtr: pointer; filler: array[1..100] of integer end
RRECMSGSIZE [*IPCRecordIO*] const = WordSize(RRecMsg)*2
RS110 [*IODefs*] const = 1
RS1200 [*IODefs*] const = 5

RS150 [*IODefs*] const = 2
RS19200 [*IODefs*] const = 9
RS2400 [*IODefs*] const = 6
RS300 [*IODefs*] const = 3
RS4800 [*IODefs*] const = 7
RS600 [*IODefs*] const = 4
RS9600 [*IODefs*] const = 8
RSExt [*IODefs*] const = 0
RtoIOvfl [*Except*] exception
RunElement [*Code*] type = (RunHeader, SysSegment, UserSegment, Import, SegFileNames)
RunFileType [*Code*] type = file of Integer
RunHead [*RunDefs*] type = record RFileFormat: integer; InitSeg: SegIndex; MainSeg: SegIndex; LinkVersion: string[20]; LinkLocation: string[20]; LinkDate: Internal_Time; LinkCommand: string; Version: long; SegEntryOffset: long; NumSegs: SegIndex; ImpIndexOffset: long; NumImports: ImpIndex; FileEntryOffset: long; NumFiles: FileIndex; end
RUNHEADLOC [*RunDefs*] const = PageWordSize
RunInfo [*Code*] type = record RFileFormat: integer; Version: integer; System: boolean; InitialGP: integer; CurOffset: integer; StackSize: integer; StackIncr: integer; HeapSize: integer; HeapIncr: integer; ProgramSN: integer; SegCount: integer end
SaphEmrServer [*SaphEmrServer*] function(InP, RepP: POINTER): boolean
Sapph_Version [*Sapph*] function(ServPort: Window): String
SapphIndex [*PascalInit*] const = 2
SapphPort [*PascalInit*] var: Port
SapphSystem [*SysType*] const = true
Scan [*Spice_String*] function(var S: Pstring; BT: breaktable; var BRK: Pstring): Pstring
ScanEnvVariables [*EnvMgr*] function(ServPort: Port; SearchScope: Env_Var_Scope; var EnvScanList: Env_Scan_List; var EnvScanList_Cnt: long): GeneralReturn
ScreenToVPCoords [*ViewPt*] procedure(ServPort: Viewport; scrX: integer; scrY: integer; var x: integer; var y: integer)
Searchlist_Separator [*EnvMgrDefs*] const = ':'
SearchlistLoop [*EnvMgrDefs*] const = Env_Error_Base + 4
SEarray [*RunDefs*] type = array[SegIndex] of SegEntry
SECName [*CFileDefs*] function(x: Bit8): string
SegBaseAddr [*CFileDefs*] function(Segnum: integer): long
SegBlock [*Code*] type = packed record case integer of 0: (ProgramSegment: boolean; LongIds: boolean; DbgInfoExists: boolean; OptimizedCode: boolean; SegBlkFiller: 0..15; QVersion: QVerRange; ModuleName: SNArray; FileName: FNString; NumSeg: integer; ImportBlock: integer; GDBSize: integer; Version: String[CommentLen]; Comment: String[CommentLen]; Source: Language; PreLinkBlock: integer; RoutDescBlock: integer; DiagBlock: integer; QMapFileName: FNString; SymFileName: FNString; CompId: TimeStamp); 1: (OffsetRD: integer; RoutsThisSeg: integer); 2: (Block: array[0..255] of integer) end
SegBlock [*SegDefs*] type = packed record case integer of 0: (ProgramSegment: boolean; LongIds: boolean; DbgInfoExists: boolean; OptimizedCode: boolean; ThirtyChIds: boolean; SegBlkFiller: 0..7; QVersion: QVerRange; ModuleName: SNArray; FileName: FNString; NumSeg: integer; ImportBlock: integer; GDBSize: integer; Version: String[CommentLen]; Comment: String[CommentLen]; Source: Language; PreLinkBlock: integer; RoutDescBlock: integer; DiagBlock: integer; QMapFileName: FNString; SymFileName: FNString; CompId: TimeStamp); 1: (OffsetRD: integer; RoutsThisSeg: integer); 2: (Block: array[0..255] of integer) end
SegEntry [*RunDefs*] type = record ModuleName: ModString; GDBSize: long; GDBLocation: long; SegFile: FileIndex; SegHdrOffset: long; SegHdrSize: long; CodeOffset: long; CodeSize: long; ImportsOffset: long; ImportsSize: long; ProcNamesOffset: long; ProcNamesSize: long; SymsFile: FileIndex; SymsOffset: long; SymsSize: long; QMapFile: FileIndex; QMapOffset: long; QMapSize: long; FirstImport: ImpIndex; NumImports: integer; end
SegFileType [*Code*] type = file of SegBlock
SegFileType [*SegDefs*] type = file of SegBlock

SegHint [*Code*] type = record case Integer of 1: (Fid: Integer; Update: TimeStamp); 2: (Word1: Integer; Word2: Integer; Word3: Integer) end

SegID [*AccentType*] type = long

SegIndex [*RunDefs*] type = 0..RFDELAYEDMAIN

SegLength [*Code*] const = 8

SegLength [*SegDefs*] const = 8

SegmentAlreadyExists [*AccentType*] const = AccErr + 48

SegmentOf [*CFileDefs*] function(Addr: long): integer

SegNode [*Code*] type = record SegId: SNArray; RootNam: pFNString; Hint: SegHint; GDBSize: integer; XSTSize: integer; GDBOff: integer; ISN: integer; CodeSize: integer; SSN: integer; UsageCnt: integer; ImpList: pImpNode; Next: pSegNode end

Send [*AccCall*] function(var xxmsg: Msg; MaxWait: long; Option: SendOption): GeneralReturn

SendEtherAddress [*EtherUser*] function(Reply: Port; MaxWait: long; Option: SendOption): GeneralReturn

SendEtherClear [*EtherUser*] function(Typ: INTEGER; Listener: Port; Reply: Port; MaxWait: long; Option: SendOption): GeneralReturn

SendEtherFilter [*EtherUser*] function(Typ: INTEGER; Listener: Port; Reply: Port; MaxWait: long; Option: SendOption): GeneralReturn

SendEtherPacket [*EtherUser*] function(Packet: POINTER; PacketWords: INTEGER; MaxWait: long; Option: SendOption): GeneralReturn

SendOption [*AccentType*] type = WAIT..REPLY

SendPupClear [*EtherUser*] function(Socket: Long; Listener: Port; Reply: Port; MaxWait: long; Option: SendOption): GeneralReturn

SendPupFilter [*EtherUser*] function(Socket: Long; Listener: Port; Reply: Port; MaxWait: long; Option: SendOption): GeneralReturn

SendPupPacket [*EtherUser*] function(PupPacket: POINTER; MaxWait: long; Option: SendOption): GeneralReturn

SendRecord [*IPCRecordIO*] function(localport: Port; remoteport: Port; id: long; MsgType: long; recptr: Pointer; recsize: integer): GeneralReturn

ServerIOPort [*IODefs*] type = Port

ServerNamePort [*IODefs*] type = Port

Sesame_Error_Base [*SesameDefs*] const = 1200

SesameIndex [*PascalInit*] const = 1

SesAuthServerPort [*SesDisk*] function(ServPort: Port; AuthServerPort: Port): GeneralReturn

SesConnect [*SesDisk*] function(ServPort: Port; RegPort: Port): GeneralReturn

SesDirectio [*SesDisk*] function(ServPort: Port; var CmdBlk: DirectIOArgs; var DataHdr: Header; var Data: DiskBuffer): GeneralReturn

SesDisk_Error_Base [*SesDiskDefs*] const = 29400

SesDiskDisMount [*SesDisk*] function(ServPort: Port; PartName: string): GeneralReturn

SesDiskMount [*SesDisk*] function(ServPort: Port; PartName: string): GeneralReturn

SesEnterForeignSesamoid [*SesDisk*] function(ServPort: Port; APathName: APath_Name; ForeignPort: Port; ForeignPrefix: APath_Name): GeneralReturn

SesEnterSegID [*SesDisk*] function(ServPort: Port; var APathName: APath_Name; SegmentID: SegID): GeneralReturn

SesGetAccessRights [*SesDisk*] function(ServPort: Port; var APathName: APath_Name; var Owner: User_ID; var Rights: Access_Rights): GeneralReturn

SesGetDefaultAccess [*SesDisk*] function(ServPort: Port; var DefaultAccess: Access_Rights): GeneralReturn

SesGetDiskPartitions [*SesDisk*] function(ServPort: Port; var PartL: ptrPartDList; var NumElts: integer): GeneralReturn

SesGetFileHeader [*Sesame*] function(ServPort: port; APathName: APath_Name; var FileHeader: File_Header): GeneralReturn

SesGetNetPort [*SesDisk*] function(ServPort: Port; var SesNetPort: Port): GeneralReturn

SesGetSegID [*SesDisk*] function(ServPort: Port; var APathName: APath_Name; var SegmentID: SegID): GeneralReturn

SesGetSegName [*SesDisk*] function(ServPort: Port; SegmentID: SegID; var APathName: APath_Name): GeneralReturn

SesLookupForeignSesamoid [*SesDisk*] function(ServPort: Port; APathName: APath_Name; var ForeignPort: Port; var ForeignPrefix: APath_Name): GeneralReturn

SesMountDevice [*SesDisk*] function(ServPort: Port; interface: DiskInterface; log_unit: InterfaceInfo; var unitnum: integer; var PartL: ptrPartDList; var NumElts: integer): GeneralReturn

SesMsgServerPort [*SesDisk*] function(ServPort: Port; MsgServerPort: Port): GeneralReturn

SesPort [*PascalInit*] var: port

SesReadBoth [*Sesame*] function(ServPort: port; var APathName: APath_Name; var Data: File_Data; var Data_Cnt: long; var FileHeader: File_Header; var NameStatus: Name_Status): GeneralReturn

SesReadFile [*Sesame*] function(ServPort: port; var APathName: APath_Name; var Data: File_Data; var Data_Cnt: long; var DataFormat: Data_Format; var CreationDate: Internal_Time; var NameStatus: Name_Status): GeneralReturn

SesScanNames [*Sesame*] function(ServPort: port; WildAPathName: Wild_APath_Name; NameFlags: Name_Flags; EntryType: Entry_Type; var DirectoryName: APath_Name; var EntryList: Entry_List; var EntryList_Cnt: long): GeneralReturn

SesSetAccessRights [*SesDisk*] function(ServPort: Port; var APathName: APath_Name; NewOwner: User_ID; NewRights: Access_Rights): GeneralReturn

SesSetDefaultAccess [*SesDisk*] function(ServPort: Port; NewDefaultAccess: Access_Rights): GeneralReturn

SesSetPOSWriteDate [*SesDisk*] function(ServPort: Port; APathName: APath_Name; WriteDate: Internal_Time): GeneralReturn

SetAsmLong [*CFileDefs*] procedure(var ByteAddr: long; NewValue: long)

SetAsmString [*CFileDefs*] procedure(var ByteAddr: long; NewValue: LString)

SetBackLog [*AccInt*] function(ServPort: port; LocalPort: port; BackLog: Integer): GeneralReturn

SetBreak [*Spice_String*] procedure(BT: BreakTable; Break, Omit: PString; Options: BreakKind)

SetCodeByte [*CFileDefs*] procedure(var ByteAddr: long; NewValue: Bit8)

SetCodeLong [*CFileDefs*] procedure(var ByteAddr: long; NewValue: long)

SetCodeWord [*CFileDefs*] procedure(var ByteAddr: long; NewValue: integer)

SetCursorPos [*ViewPt*] procedure(ServPort: Viewport; x: integer; y: integer)

SetDataLong [*CFileDefs*] procedure(var ByteAddr: long; NewValue: long)

SetDateTime [*Time*] procedure(ServPort: port; ITime: Internal_Time)

SetDebugPort [*AccInt*] function(ServPort: port; DebugPort: port): GeneralReturn

SetEnvVariable [*EnvMgr*] function(ServPort: Port; Name: Env_Var_Name; VarType: Env_Var_Type; VarScope: Env_Var_Scope; Variable: Env_Variable; Variable_Cnt: long): GeneralReturn

SetKernelWindow [*AccInt*] function(ServPort: port; LeftX: Integer; TopY: Integer; Width: Integer; Height: Integer; Inverted: Boolean): GeneralReturn

SetLimit [*AccInt*] function(ServPort: port; ReplyPort: port; Limit: Long): GeneralReturn

SetListener [*ViewPt*] procedure(ServPort: Viewport)

SetPagingSegment [*AccInt*] function(ServPort: port; Segment: SegID): GeneralReturn

SetPMPort [*Sapph*] procedure(ServPort: Window; ProcCtlPort: Port)

SetPortsWaiting [*AccCall*] function(var ports: PortBitArray): GeneralReturn

SetPriority [*AccInt*] function(ServPort: port; Priority: PriorID): GeneralReturn

SetProtection [*AccInt*] function(ServPort: port; Address: VirtualAddress; NumBytes: Long; Protection: Integer): GeneralReturn

SetRegionCursor [*ViewPt*] procedure(ServPort: Viewport; regionNum: integer; cursorImage: CursorSet; cursIndex: integer; cursFunc: CursorFunction; track: boolean)

SetRegionParms [*ViewPt*] procedure(ServPort: Viewport; regionNum: integer; absolute: boolean; speed: integer; minx: integer; maxx: integer; miny: integer; maxy: integer; modx: integer; posx: integer; mody: integer; posy: integer)

SetSystemZone [*Time*] procedure(ServPort: port; TimeZone: integer; DSTWhenTimely: boolean)

SetTempSegPartition [*AccInt*] function(ServPort: port; PartName: DevPartString): GeneralReturn

SetWindowAttention [*Sapph*] procedure(ServPort: Window; attn: boolean)

SetWindowError [*Sapph*] procedure(ServPort: Window; error: boolean)

SetWindowName [*Sapph*] procedure(ServPort: Window; var progName: ProgStr)
SetWindowProgress [*Sapph*] procedure(ServPort: Window; nestLevel: integer; value: Long; max: Long)
SetWindowRequest [*Sapph*] procedure(ServPort: Window; requesting: boolean)
SetWindowTitle [*Sapph*] procedure(ServPort: Window; title: TitStr)
shell_input_redirect [*CommandParse*] const = '<'
shell_output_redirect [*CommandParse*] const = '>'
shell_parallel_command [*CommandParse*] const = '&'
shell_piped_command [*CommandParse*] const = '|'
shell_special_args_start [*CommandParse*] const = '['
shell_special_args_stop [*CommandParse*] const = ']'
ShowBreak [*Spice_String*] function(BT: BreakTable): PString
ShowPathAndTitle [*WindowUtils*] procedure(s: TitStr)
ShowRun [*ALoad*] procedure(p: pointer; MapFileName: Path_Name)
ShowWindowAttentionFlag [*WindowUtils*] procedure
ShowWindowErrorFlag [*WindowUtils*] procedure
ShowWindowRequestFlag [*WindowUtils*] procedure
ShrinkWindow [*Sapph*] procedure(ServPort: Window)
SigDebug [*ProcMgrDefs*] const = MinSignal + 36
SigLevel1Abort [*ProcMgrDefs*] const = MinSignal + 37
SigLevel2Abort [*ProcMgrDefs*] const = MinSignal + 38
SigLevel3Abort [*ProcMgrDefs*] const = MinSignal + 39
SignalAction [*ProcMgrDefs*] type = (SigDefault, SigIgnore, SigSend)
SignalBase [*ProcMgrDefs*] const = 3800
SignalMsg [*ProcMgrDefs*] type = record Head: Msg; tCtlWindow: TypeType; CtlWindow: Window; tSignal: TypeType; Signal: SignalName; end
SignalMsgID [*ProcMgrDefs*] const = 3801
SignalName [*ProcMgrDefs*] type = integer
SigResume [*ProcMgrDefs*] const = MinSignal + 34
SigStatus [*ProcMgrDefs*] const = MinSignal + 35
SigSuspend [*ProcMgrDefs*] const = MinSignal + 33
SimpleName [*PathName*] function(PathName: Path_Name): Entry_Name
Sin [*RealFunctions*] function(X: Real): Real
single_char_quote [*CommandParse*] const = '\'
SinH [*RealFunctions*] function(x: real): real
SinHLarge [*RealFunctions*] exception(X: Real)
SinLarge [*RealFunctions*] exception(X: Real)
SIOSenseStatus [*IODefs*] type = packed record RxCharAvailable: boolean; IntPending: boolean; TxBufferEmpty: boolean; DCD: boolean; SyncHunt: boolean; CTS: boolean; TransmitUnderRun: boolean; BreakAbort: boolean; AllSent: boolean; Residue: Bit3; ParityError: boolean; RxOverRun: boolean; CrcFramingError: boolean; EndOfFrame: boolean; end
SIOWriteRegister [*IODefs*] type = packed record case RegNum: Bit8 of 0: (); 1: (RegVal: Bit8) end
SmallReal [*Stream*] exception(FileName: SName)
SName [*Stream*] type = string[255]
SNArray [*Code*] type = packed array[1..SegLength] of Char
SNArray [*SegDefs*] type = packed array[1..SegLength] of Char
SoftEnable [*AccCall*] function(NormOrEmerg: boolean; EnOrDis: boolean): GeneralReturn
SoftInterrupt [*AccInt*] function(ServPort: port; NormOrEmerg: Boolean; var EnOrDisable: Boolean): GeneralReturn
Source_entries_per_block [*QMapDefs*] const = 256 div WSSourceMapRecord
SourceMap [*QMapDefs*] type = Array[0..Source_entries_per_block – 1] of SourceMapRecord
SourceMapRecord [*QMapDefs*] type = record FileName: APath_Name; CompID: Internal_Time; Filler: array[0..124] of integer; FileName: PathName; fileBlocks, fileBits: Integer; end

Spawn [*Spawn*] function(VAR ChildKPort: Port; VAR ChildDPort: Port; ProgName: APath_Name; ProcName: STRING; HisCommand: CommandBlock; DebugIt: BOOLEAN; ProtectChild: BOOLEAN; SapphConn: ConnectionInheritance; pWindow: Port; pTypeScript: Port; EMConn: ConnectionInheritance; pEMPort: Port; PassedPorts: ptrPortArray; NPorts: LONG; LoaderDebug: BOOLEAN): GeneralReturn

SpiceSegKind [*AccentType*] type = (Temporary, Permanent, Bad, SegPhysical, Imaginary, Shadow)

Split [*Spawn*] function(VAR ChildKPort: Port; VAR ChildDPort: Port): GeneralReturn

Sqrt [*RealFunctions*] function(X: Real): Real

SqrtNeg [*RealFunctions*] exception(X: Real)

Squeeze [*Spice_String*] function(Str: PString): PString

StackLeader [*Code*] const = 2

StatArray [*ProcMgrDefs*] type = array[0..0] of StatRecord

StatList [*ProcMgrDefs*] type = ↑StatArray

StatRecord [*ProcMgrDefs*] type = record RunTime: long; LoadTime: long; ElapsedTime: long; KernelPort: long; Priority: integer; QueueID: integer; ProcName: string; IconName: ProgStr; State: ProcState; end

Status [*AccInt*] function(ServPort: port; var NStats: PStatus): GeneralReturn

str [*Spice_String*] function(Ch: char): PString

StrBadParm [*Spice_String*] exception(FuncName, StringArgument: PString; ParmValue: integer)

StreamBuffer [*Stream*] type = record case integer of 0: (W: array[0..255] of integer); 1: (B1: packed array[0..0] of 0..1); 2: (B2: packed array[0..0] of 0..3); 3: (B3: packed array[0..0] of 0..7); 4: (B4: packed array[0..0] of 0..15); 5: (B5: packed array[0..0] of 0..31); 6: (B6: packed array[0..0] of 0..63); 7: (B7: packed array[0..0] of 0..127); 8: (B8: packed array[0..0] of 0..255); 9: (C: packed array[0..255] of char); end

StreamClose [*Stream*] procedure(var F: FileType)

StreamF [*Stream*] type = record FName: SName; Buffer: pStreamBuffer; BufSize: long; WordIndex: long; LastElt: integer; EltIndex: integer; end

StreamFlushOutput [*Stream*] procedure(var F: Text)

StreamInit [*Stream*] procedure(var F: FileType; WordSize, BitSize: integer; CharFile: boolean)

StreamKeyBoardReset [*Stream*] procedure(var F: Text)

StreamName [*Stream*] function(var F: FileType): SName

StreamOpen [*Stream*] procedure(var F: FileType; var Name: SName; WordSize, BitSize: integer; CharFile: boolean; OpenWrite: boolean)

StreamProgress [*WindowUtils*] procedure(var F: File)

StreamVersion [*Stream*] const = '3.9'

StrIndx [*Except*] exception

String_255 [*TimeDefs*] type = string[255]

Strip [*Spice_String*] function(Str: PString): PString

StripCurrent [*PathName*] function(var WildPathName: Wild_Path_Name): GeneralReturn

StrLong [*Except*] exception

STSChangeEnv [*TS*] procedure(ServPort: Typescript; env: Port)

STSFlushInput [*TS*] procedure(ServPort: Typescript)

STSFlushOutput [*TS*] procedure(ServPort: Typescript)

STSFullLine [*TS*] function(ServPort: Typescript): Boolean

STSFullOpen [*TS*] function(ServPort: Port; vp: ViewPort; env: Port; fontName: TString255; doWrap: Boolean; dispPages: Integer): Typescript

STSFullOpenWindow [*TS*] function(ServPort: Port; w: Window; env: Port; fontName: TString255; doWrap: Boolean; dispPages: Integer): Typescript

STSGetChar [*TS*] function(ServPort: Typescript): Char

STSGetString [*TS*] function(ServPort: Typescript): TString255

STSGrabWindow [*TS*] function(ServPort: Typescript; kPort: Port): Window

STSOpen [*TS*] function(ServPort: Port; vp: ViewPort; env: Port): Typescript

STSOpenWindow [*TS*] function(ServPort: Port; w: Window; env: Port): Typescript

STSPutChar [*TS*] procedure(ServPort: Typescript; ch: Char)

STSPutCharArray [*TS*] procedure(ServPort: Typescript; chars: pTSCharArray; chars_Cnt: long; firstCh: Integer; lastCh: Integer)
STSPutString [*TS*] procedure(ServPort: Typescript; s: TString255)
SubDeleteName [*Sesame*] function(ServPort: port; APathName: APath_Name): GeneralReturn
SubEnterName [*Sesame*] function(ServPort: port; var APathName: APath_Name; EntryType: Entry_Type; EntryData: Entry_Data): GeneralReturn
SubLookUpName [*Sesame*] function(ServPort: port; var APathName: APath_Name; var EntryType: Entry_Type; var EntryData: Entry_Data; var NameStatus: Name_Status): GeneralReturn
SubReadFile [*Sesame*] function(ServPort: port; APathName: APath_Name; var Data: File_Data; var Data_Cnt: long): GeneralReturn
SubReName [*Sesame*] function(ServPort: port; OldAPathName: APath_Name; var NewAPathName: APath_Name): GeneralReturn
SubstrFor [*Spice_String*] function(Source: PString; Index, Size: Integer): PString
SubstrTo [*Spice_String*] function(Source: PString; Index, EndIndex: Integer): PString
SubTestName [*Sesame*] function(ServPort: port; var APathName: APath_Name; var EntryType: Entry_Type; var NameStatus: Name_Status): GeneralReturn
SubWriteFile [*Sesame*] function(ServPort: port; var APathName: APath_Name; Data: File_Data; Data_Cnt: long; DataFormat: Data_Format; var CreationDate: Internal_Time): GeneralReturn
Success [*AccentType*] const = AccErr + 1
Suspend [*AccInt*] function(ServPort: port): GeneralReturn
switch_leadin_char [*CommandParse*] const = '-'
SyncIO [*IO*] function(ServPort: ServerIOPort; Command: IOCommand; CmdBlk: Pointer; CmdBlk_Cnt: long; var DataBuf: Pointer; var DataBuf_Cnt: long; DataTransferCnt: Long; TimeOut: Long; var Status: IOStatusBlk): GeneralReturn
SysFontHeight [*SapphDefs*] const = 13
SysFontName [*SapphDefs*] const = 'Fix13.Kst'
SysFontWidth [*SapphDefs*] const = 9
T_IntToString [*Time*] function(ServPort: port; ITime: Internal_Time; TimeFormat: integer): String
T_IntToUser [*Time*] function(ServPort: port; ITime: Internal_Time): User_Time
T_IntToZone [*Time*] function(ServPort: port; ITime: Internal_Time; TZone: Zone_Info): User_Time
T_Never [*Time*] function(ServPort: port): Internal_Time
T_StringToInt [*Time*] function(ServPort: port; STime: String_255; var Index: integer; var WhatIFound: integer): Internal_Time
T_StringToUser [*Time*] function(ServPort: port; STime: String_255; var Index: integer; var WhatIFound: integer): User_Time
T_UserToInt [*Time*] function(ServPort: port; UTime: User_Time): Internal_Time
T_UserToString [*Time*] function(ServPort: port; UTime: User_Time; TimeFormat: integer): String
Tan [*RealFunctions*] function(X: Real): Real
TanH [*RealFunctions*] function(x: real): real
TanLarge [*RealFunctions*] exception(X: Real)
Terminate [*AccInt*] function(ServPort: port; Reason: Long): GeneralReturn
TextState [*CFileDefs*] const = 0
TF_12_Hour [*TimeDefs*] const = #004000
TF_ANSI [*TimeDefs*] const = #000300
TF_ANSI_Ordinal [*TimeDefs*] const = #000340
TF_BlankPad [*TimeDefs*] const = #020000
TF_Dashes [*TimeDefs*] const = #000000
TF_DateFormat [*TimeDefs*] const = #000340
TF_FullMonth [*TimeDefs*] const = #000010
TF_FullWeekday [*TimeDefs*] const = #000002
TF_FullYear [*TimeDefs*] const = #000020
TF_Milliseconds [*TimeDefs*] const = #002000
TF_Never [*TimeDefs*] const = #100000
TF_NoColumns [*TimeDefs*] const = #040000

TF_NoDate [*TimeDefs*] const = #000004
TF_NoSeconds [*TimeDefs*] const = #001000
TF_NoTime [*TimeDefs*] const = #000400
TF_Reversed [*TimeDefs*] const = #000100
TF_Slashes [*TimeDefs*] const = #000140
TF_Spaces [*TimeDefs*] const = #000040
TF_TimeZone [*TimeDefs*] const = #010000
TF_Weekday [*TimeDefs*] const = #000001
This_Directory [*SesameDefs*] const = './'
Time_Fields [*TimeDefs*] type = packed record Hour: 0..24; Minute: 0..59; Second: 0..59; Millisecond: 0..999; end
TimeIndex [*PascalInit*] const = 0
TimeNotInitialized [*Time*] exception
TimeOut [*AccentType*] const = AccErr + 2
TimeOutError [*Stream*] exception(FileName: SName)
TIMEOUTUSECS [*SapphFileDefs*] const = 16666
TimePort [*PascalInit*] var: port
TimeStamp [*OldTimeStamp*] type = packed record Hour: 0..23; Day: 1..31; Second: 0..59; Minute: 0..59; Month: 1..12; Year: 0..63; end
TitleOverhead [*SapphDefs*] const = SysFontHeight + 6
TitStr [*SapphDefs*] type = String[TitStrLength]
TitStrLength [*SapphDefs*] const = LandScapeBitWidth div SysFontWidth
TooManyHeaps [*Dynamic*] exception
TooManyReplies [*AccentType*] const = AccErr + 5
Touch [*AccInt*] function(ServPort: port; Address: VirtualAddress): GeneralReturn
TP_Date [*TimeDefs*] const = #000002
TP_Never [*TimeDefs*] const = #000020
TP_RESERVED [*TimeDefs*] const = #177740
TP_Time [*TimeDefs*] const = #000004
TP_Weekday [*TimeDefs*] const = #000001
TP_Zone [*TimeDefs*] const = #000010
TraceHeap [*Dynamic*] procedure(S: HeapNumber; Trace: boolean)
TrapCodes [*AccentType*] type = (TrapInit, TrapReadFault, TrapWriteFault, TrapSend, TrapReceive, TrapSetPortsWaiting, TrapPortsWithMessages, TrapDebugWrite, TrapException, TrapNothing, TrapRectDrawLine, TrapRectRasterOp, TrapCharRead, TrapFull, TrapFlush, TrapMoveWords, TrapRectPutString, TrapError, TrapClockEnable, TrapGPRead, TrapGPWrite, TrapSoftEnable, TrapGetIOSleepID, TrapRectColor, TrapRectScroll, TrapLockPorts, TrapMessagesWaiting)
Trim [*Spice_String*] function(Str: PString): PString
TruncateSegment [*AccInt*] function(ServPort: port; Segment: SegID; NewSize: Integer): GeneralReturn
TSCharArray [*TSDefs*] type = packed array[0..1] of Char
TString255 [*TSDefs*] type = String[255]
TYPEBIT [*AccentType*] const = 0
TYPEBOOLEAN [*AccentType*] const = 0
TYPEBYTE [*AccentType*] const = 9
TYPECHAR [*AccentType*] const = 8
TYPEINT16 [*AccentType*] const = 1
TYPEINT32 [*AccentType*] const = 2
TYPEINT8 [*AccentType*] const = 9
TYPEPAGE [*AccentType*] const = 14
TYPEPSTAT [*AccentType*] const = 11
TYPEPT [*AccentType*] const = 6
TYPEPTALL [*AccentType*] const = 5
TYPEPTOWNERSHIP [*AccentType*] const = 3

TYPEPTRECEIVE [*AccentType*] const = 4
TYPEREAL [*AccentType*] const = 10
Typescript [*TSDefs*] type = Port
TypescriptIndex [*PascalInit*] const = 3
TypescriptPort [*PascalInit*] var: Port
TYPESEGID [*AccentType*] const = 13
TYPESTRING [*AccentType*] const = 12
TypeType [*AccentType*] type = packed record case integer of 1: (TypeName: Bit8; TypeSizeInBits: Bit8; NumObjects: Bit12; InLine: boolean; LongForm: boolean; Deallocate: boolean); 2: (LongInteger: long) end
TYPEUNSTRUCTURED [*AccentType*] const = 0
ULInitial [*Spice_String*] function(Str1, Str2: PString): boolean
ULPosString [*Spice_String*] function(Source, Mask: PString): Integer
UncaughtException [*AccentType*] const = AccErr + 44
UNCHANGED [*SapphDefs*] const = -32001
UndefinedGlobal [*CFileDefs*] const = UndefinedSymbol
UndefinedSymbol [*CFileDefs*] const = 7
UndfDevice [*Stream*] exception
UndfInt [*Except*] exception
UndfQcd [*Except*] exception
UndReal [*Except*] exception
UniqueWordIndex [*CommandParse*] function(table: pWord_Search_Table; ptrWordString: pWord_String; var WordText: Cmnd_String): integer
UnitIOError [*Stream*] exception(FileName: SName)
UnknownAction [*ProcMgrDefs*] const = ProcMgrBase + 3
UnknownPort [*ProcMgrDefs*] const = ProcMgrBase + 7
UnknownProcess [*ProcMgrDefs*] const = ProcMgrBase + 1
UnknownSignal [*ProcMgrDefs*] const = ProcMgrBase + 2
UnknownWindow [*ProcMgrDefs*] const = ProcMgrBase + 4
UnrecognizedMsgType [*AccentType*] const = AccErr + 15
UNSPECEXCEPTION [*AccentType*] const = 5
Up_one_Directory [*SesameDefs*] const = '../'
UpChar [*Spice_String*] function(C: Char): Char
UpEQU [*Spice_String*] function(Str1: PString; Str2: PString): boolean
User_ID [*AuthDefs*] type = No_User..Max_Users
User_Time [*TimeDefs*] type = packed record Date: Date_Fields; Time: Time_Fields; Zone: Zone_Info; end
UserCommand [*PascalInit*] var: CommandBlock
UserError [*ViewPt*] exception(s: String)
UserEventPort [*IODefs*] type = Port
UserNameNotFound [*AuthDefs*] const = Auth_Error_Base + 1
UserRecord [*AuthDefs*] type = record Name: Auth_Var; UserID: User_ID; EncryptPass: PassType; Profile: APath_Name; NameOfShell: APath_Name; End
UserTypescript [*PascalInit*] var: Port
UserWindow [*PascalInit*] var: Port
UserWindowShared [*PascalInit*] var: Boolean
Valid_Name_Chars [*SesameDefs*] const = '$-.+0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz'
ValidateMemory [*AccInt*] function(ServPort: port; var Address: VirtualAddress; NumBytes: Long; CreateMask: Long): GeneralReturn
value_marker_char [*CommandParse*] const = '='
Var_CharAr [*SymDefs*] type = Packed Array[0..511] of Char
Var_IntAr [*SymDefs*] type = Array[0..255] of Integer

var_ptr [*SymDefs*] type = Record Case Integer Of 0: (P: POINTER); 1: (p: pDirBlk); 2: (dict: pVarDictEntry); 3: (chars: pVar_CharAr); 4: (int: pVar_IntAr); End

VarDescriptor [*SymDefs*] type = Packed Record mainType: BaseType; subType: BaseType; etc: 0..255; End

VarDictEntry [*SymDefs*] type = Record CompID: Internal_Time; Globals: Integer; Routines: Array[0..Max_SymRoutines] of Integer; End

VarNameArray [*SymDefs*] type = Record TotNumChars: integer; VarNames: Packed Array[0..0] of Char; End

Ver_Separator [*SesameDefs*] const = '#'

ViewChar [*ViewPt*] procedure(ServPort: Viewport; fontVP: Viewport; funct: RopFunct; var dx: integer; var dy: integer; ch: char)

ViewChArray [*ViewPt*] procedure(ServPort: Viewport; fontVP: Viewport; funct: RopFunct; var dx: integer; var dy: integer; chars: pVPCharArray; chars_Cnt: long; firstCh: integer; var lastch: integer)

ViewColorRect [*ViewPt*] procedure(ServPort: Viewport; funct: RectColorFunct; x: integer; y: integer; width: integer; height: integer)

ViewLine [*ViewPt*] procedure(ServPort: Viewport; funct: LineFunct; x1: integer; y1: integer; x2: integer; y2: integer)

Viewport [*SapphDefs*] type = Port

ViewportState [*ViewPt*] procedure(ServPort: Viewport; var curIx: integer; var curty: integer; var curwidth: integer; var curheight: integer; var curRank: integer; var memory: boolean; var courteous: boolean; var transparent: boolean)

ViewPutChar [*ViewPt*] procedure(ServPort: Viewport; fontVP: Viewport; funct: RopFunct; dx: integer; dy: integer; ch: char)

ViewPutChArray [*ViewPt*] procedure(ServPort: Viewport; fontVP: Viewport; funct: RopFunct; dx: integer; dy: integer; chars: pVPCharArray; chars_Cnt: long; firstCh: integer; lastch: integer)

ViewPutString [*ViewPt*] procedure(ServPort: Viewport; fontVP: Viewport; funct: RopFunct; dx: integer; dy: integer; str: VPStr255; firstCh: integer; lastch: integer)

ViewROP [*ViewPt*] procedure(ServPort: Viewport; funct: RopFunct; dx: integer; dy: integer; width: integer; height: integer; srcVP: Viewport; sx: integer; sy: integer)

ViewScroll [*ViewPt*] procedure(ServPort: Viewport; x: integer; y: integer; width: integer; height: integer; Xamt: integer; Yamt: integer)

ViewString [*ViewPt*] procedure(ServPort: Viewport; fontVP: Viewport; funct: RopFunct; var dx: integer; var dy: integer; str: VPStr255; firstCh: integer; var lastch: integer)

VirtualAddress [*AccentType*] type = long

VPChar [*ViewKern*] procedure(destvp, fontVP: Viewport; funct: RopFunct; var dx, dy: Integer; ch: Char)

VPCharArray [*SapphDefs*] type = Packed Array[0..1] of Char

VPChArray [*ViewKern*] procedure(destvp, fontVP: Viewport; funct: RopFunct; var dx, dy: Integer; chars: pVPCharArray; arSize: Long; firstCh: Integer; var lastch: Integer)

VPColorRect [*ViewKern*] procedure(vp: Viewport; funct: RectColorFunct; x, y, width, height: Integer)

VPExclusionFailure [*AccentType*] const = AccErr + 40

VPIntegerArray [*SapphDefs*] type = Array[0..0] of Integer

VPLine [*ViewKern*] procedure(destvp: Viewport; funct: LineFunct; x1, y1, x2, y2: Integer)

VPNIY [*ViewPt*] exception

VPPortArray [*SapphDefs*] type = Record num: Integer; ar: Array[0..0] of Port; end

VPPutChar [*ViewKern*] procedure(destvp, fontVP: Viewport; funct: RopFunct; dx: Integer; dy: Integer; ch: Char)

VPPutChArray [*ViewKern*] procedure(destvp, fontVP: Viewport; funct: RopFunct; dx, dy: Integer; chars: pVPCharArray; arSize: Long; firstCh, lastch: Integer)

VPPutString [*ViewKern*] procedure(destvp, fontVP: Viewport; funct: RopFunct; dx, dy: Integer; var str: VPStr255; firstCh, lastch: Integer)

VPREGION [*SapphDefs*] const = 1

VPROP [*ViewKern*] procedure(destvp: Viewport; funct: RopFunct; dx, dy, width, height: Integer; srcVP: Viewport; sx, sy: Integer)

VPScroll [*ViewKern*] procedure(destvp: Viewport; x, y, width, height, Xamt, Yamt: Integer)

VPStr255 [*SapphDefs*] type = string[255]
VPString [*ViewKern*] procedure(destvp, fontVP: Viewport; funct: RopFunct; var dx, dy: Integer; var str: VPStr255; firstCh: Integer; var lastch: Integer)
VPtoScreenCoords [*ViewPt*] procedure(ServPort: Viewport; x: integer; y: integer; var scrX: integer; var scrY: integer)
WAIT [*AccentType*] const = 0
WaitEtherAddress [*EtherUser*] function: INTEGER
WaitEtherClear [*EtherUser*] function(Typ: INTEGER; Listener: Port): BOOLEAN
WaitEtherFilter [*EtherUser*] function(Typ: INTEGER; Listener: Port): BOOLEAN
WaitPupClear [*EtherUser*] function(VAR Socket: Long; Listener: Port): BOOLEAN
WaitPupFilter [*EtherUser*] function(VAR Socket: Long; Listener: Port): BOOLEAN
Wild_APath_Name [*SesameDefs*] type = string[Path_Name_Size]
Wild_Path_Name [*PathName*] type = string[Path_Name_Size]
WILDREGION [*SapphDefs*] const = 31
WillReply [*AccentType*] const = AccErr + 4
Window [*SapphDefs*] type = Port
WindowInUse [*ProcMgrDefs*] const = ProcMgrBase + 5
WindowViewport [*Sapph*] procedure(ServPort: Window; var vp: Viewport; var vpWidth: integer; var vpHeight: integer)
WinForName [*Sapph*] function(ServPort: Window; name: ProgStr): Window
WinForViewPort [*Sapph*] function(ServPort: Window; vp: Viewport; var isouter: boolean): Window
WinNameArray [*SapphDefs*] type = Array[0..0] of ProgStr
Word_String [*CommandParse*] type = String[1]
Word_Type [*CommandParse*] type = (in_arg, out_arg, switch_arg, switch_value, command_file)
WordifyPool [*CommandParse*] function(ChPool: pCharacter_Pool; PoolLength: Char_Pool_Index; var WordStruct: CommandBlock): GeneralReturn
WriteBoolean [*Writer*] procedure(var F: FileType; X: Boolean; Field: integer)
WriteCh [*Writer*] procedure(Var F: FileType; X: char; Field: integer)
WriteChArray [*Writer*] procedure(var F: FileType; var X: ChArray; Max, Field: integer)
WriteChars [*Stream*] procedure(VAR F: FileType; VAR S: String)
WriteD [*PasLong*] procedure(var F: FileType; X: long; Field, B: integer)
WriteFault [*AccentType*] const = AccErr + 26
WriteFile [*PathName*] function(var PathName: Path_Name; Data: File_Data; ByteCount: long): GeneralReturn
WriteIdentifier [*Writer*] procedure(var F: FileType; X: integer; var IT: IdentTable; L, Field: integer)
WriteInteger [*Writer*] procedure(var F: FileType; X: integer; Field: integer)
WriteNChars [*Stream*] procedure(VAR F: FileType; c: char; N: Integer)
WriteProcessMemory [*AccInt*] function(ServPort: port; Address: VirtualAddress; NumBytes: Long; Data: Pointer; Data_Cnt: long): GeneralReturn
WriteR [*PasReal*] procedure(var F: FileType; e: real; TotalWidth: integer; FracDigits: integer; format: integer)
WriteSegment [*AccInt*] function(ServPort: port; Segment: SegID; Offset: Integer; Data: Pointer; Data_Cnt: long): GeneralReturn
WriteString [*Writer*] procedure(var F: FileType; var X: String; Field: integer)
WriteX [*Writer*] procedure(var F: FileType; X, Field, B: integer)
WRONGARGS [*AccentType*] const = 2
WrongEnvVarType [*EnvMgrDefs*] const = Env_Error_Base + 2
WS_NotFound [*CommandParse*] const = −1
WS_NotUnique [*CommandParse*] const = −2
WSQMapInfoRecord [*QMapDefs*] const = WordSize(QMapInfoRecord)
WSSourceMapRecord [*QMapDefs*] const = WordSize(SourceMapRecord)
Zone_Info [*TimeDefs*] type = packed record TimeZone: integer; UseTimeZone: boolean; Daylight: boolean; UseDaylight: boolean; end