PERQ Systems Corporation

C SYSTEM INTERFACES

Part Two

—

June 30, 1985

This document is for use with C Version 2.0, which runs under Accent Version S6 with Amendment No. 2.

—

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

6.30. MoveBytes

6.30.1. Function MoveBytes

Code:

```
extern void movebytes();
 /* char *dst;
    char *src;
    int numbytes;
  */
```

Abstract:

Move numbytes bytes from dst to src quickly using the movebytes qcode.

Parameters:

dst	Destination byte address
src	Source byte addres
numbytes	Number of bytes to be moved

Design:

MoveBytes first transforms its given src and dst c-style byte pointers into the pascal-style (3-word) byte pointers used by the qcode. Note that pascal-style byte pointers are formed by concatenating a 32-bit virtual word address with a 16-bit character offset from that base address. Thus a 48-bit (3-word) byte pointer is formed.

MoveBytes simply casts the c-style 32-bit byte pointer into a pointer to an int to form a word address. If the original byte pointer is odd then the character offset is one, otherwise it is zero.

The QCode is only able to move 32000 (approx.) bytes at a time. Thus MoveBytes loops moving 32000 byte chunks and then moves a final small

chunk. In practice, MoveBytes is probably never asked to move very large arrays of bytes.

6.31. MsgN

File MsgN.h provides routines to manipulate the network-wide name-port mapping. See the document "The Name Server" in the *Accent Programming Manual.*

Code:

```
#include <AccentType.h>

extern void InitMsgN () ;
  /*  Port RPort;  */

extern GeneralReturn CheckIn ( );
  /*  Port ServPort;
      String PortsName;
      Port Signature;
      Port PortsID;
  */


extern GeneralReturn Lookup ( );
  /*  Port ServPort;
      String PortsName;
      Port  * PortsID;
  */


extern GeneralReturn CheckOut ( );
  /*  Port ServPort;
      String PortsName;
      Port Signature;
  */


extern GeneralReturn MsgPortStatus ( );
  /*  Port ServPort;
      Port PortsID;
```

```
        long  * GlobalPort;
        long  * Owner;
        long  * Receiver;
        long  * SrcID;
        long  * SeqNum;
        Boolean  * NetWaiting;
        short  * NumQueued;
        Boolean  * Blocked;
        Boolean  * Locked;
        short  * RecvQueue;
        long  * DataOffset;
        long  * InSrcID;
        long  * InSeqNum;
    */

extern GeneralReturn PacketStats ();
    /*  Port ServPort;
        Boolean Reset;
        long  * PacketsQued;
        long  * PacketsSent;
        long  * PacketsRecd;
        long  * AcksSent;
        long  * AcksRecd;
        long  * ReXmitsSent;
        long  * ReXmitsRecd;
        long  * BlocksSent;
        long  * BlocksRecd;
    */

extern GeneralReturn Msg_Version ();
    /*  Port ServPort;
        String  * Version;
    */
```

6.32. NameErrors

File NameErrors.h exports the MsgServer (NameServer) errors.

Code:

```
#define NameBase        1000
#define NameNotYours    NameBase + 0
#define NameNotCheckedIn    NameBase + 1
```

6.33. OldBuiltinDefs

File OldBuiltinDefs provides typedefs for Matchmaker-generated interfaces.

Code:

```
#include <AccentType.h>

typedef short Short;
typedef char  Byte;
typedef Byte  ByteArray[];
typedef ByteArray *ptrByteArray;
typedef Port Port_All;
typedef Port Port_Receive;
typedef Port Port_Ownership;

/* see MoveBytes.h for documentation on this function */
extern void movebytes ();
  /* char *dst;
     char *src;
     int numbytes;
   */
```

6.34. OldTime

File OldTime.h is a compatibility file for converting between new time values and POS timestamps. Also see the section on "Time."

Code:

```
#include <timedefs.h>
/* typedef struct
        {
           unsigned int Hour : 5;
           unsigned int Day : 5;
           unsigned int Second : 6;
           unsigned int Minute : 6;
           unsigned int Month : 4;
           unsigned int Year : 6;
        }
         TimeStamp;

 /* TimeStamp declaration if bit fields */

typedef int TimeStamp;
  /*     No parameters     */

extern TimeStamp OldCurrentTime();
  /*

extern TimeStamp NewToOldTime();
  /* Internal_Time NewTime */

extern Internal_Time OldToNewTime();
  /* TimeStamp OldTime */
```

6.35. PasExtens

6.35.1. Function ROTATE

Code:

```
extern short ROTATE();
 /*     short value;
        short distance;
  */
```

Abstract:

Does a left or right rotate of a 16 bit number.

Parameters:

value — Contains number which is to be rotated

dist — Number of positions to rotate value. If dist is > 0 then a right shift of dist bits occurs, otherwise a left shift of dist bits occurs.

Returns:

The rotated number

6.36. PathName/PathNameDefs

Files PathName.h and PathNameDefs.h provide routines and definitions for a non-primitive interface to the file server. These routines make use of both environment manager functions and name server functions. All pathnames should be handled by routines at this level.

See the document "The File System" in the *Accent Programming Manual.*

6.36.1. PathName

Include Files:

```
#include <pathnamedefs.h>
```

6.36.1.1. Function WriteFile

Code:

```
extern GeneralReturn WriteFile();
  /*     char * PathName;
         File_Data Data;
         int ByteCount;
   */
```

Abstract:

This is equivalent to the composition of ExpandPathName with SubWriteFile.

Parameters:

PathName
: Name of File to be written

Data
: File Data to be written into PathName

ByteCount

Number of Bytes to be written

Returns:

Error Code from ExpandPathName or (if Successful) Error or Success code from SubWriteFile

6.36.1.2. Function ReadFile

Code:

```
extern GeneralReturn ReadFile();
  /*     char * PathName;
         File_Data * Data;
         int * ByteCount;
   */
```

Abstract:

This is equivalent to the composition of FindFileName with SubReadFile.

Parameters:

PathName
: The Name of the file to be read

File_Data
: Points to the area where the data read should be stored

ByteCount
: The number of bytes read

Returns:

Error code from FindTypedName (unless this error code was ImproperEntryType, then NotAFile is returned), or Success or Error Code from SubReadFile

6.36.1.3. Function FindExtendedPathName

Code:

```
extern GeneralReturn FindExtendedPathName();
  /*     char * PathName;
         char * ExtensionList;
         char * ImplicitSearchList;
         Boolean FirstOnly;
         Entry_Type *EntryType;
        _Name_Status *NameStatus;
   */
```

Abstract:

Finds a name of any type using an extension list and a search list. Should be used for names of arbitrary type before doing a lookup-like operation.

Parameters:

PathName The path name to look for. Changed to the name actually found. The user must make sure that the amount of space allocated for this string is large enough to hold the complete extended pathname. (i.e., passing a char * which has been assigned a string constant is not a good idea).

ExtensionList
The list of extensions. Each name in the extension list is terminated by a semicolon (';'). Note that the extensions must be of the form ".ext" and not "ext" (i.e., make sure that a "." begins the extension. A null name implies a null extension. An extensionlist might look like: ".o;.s;.c;.exe".

ImplicitSearchList
Name of the search list to use if a partial path name is supplied. If blank, the list "Default" is used.

FirstOnly If true, only look in the first item of the search list. Should

ordinarily be false.

EntryType
: Returns the entry type of the name

NameStatus
: Returns the version status of the name

Returns:

Success

SearchlistNotFound

NameNotFound

6.36.1.4. Function StripCurrent

Code:

```
extern GeneralReturn StripCurrent();
  /*     char * WildPathName;     */
```

Abstract:

Shortens a path name if it is accessible from the current search list.

Parameters:

WildPathName
: The path name to check. If the first component of 'current' is a prefix of the path name, it is stripped off and the rest is returned.

Returns:

Any return value from ResolveSearchList, FirstItemNotDefined, or Failure

6.36.1.5. Function AddExtension

Code:

```
extern void AddExtension();
  /*     char * filename;
         char * Extension;
   */
```

Abstract:

Adds an extension to a file name if it is not already there.

Parameters:

FileName Name to check. It is changed if the extension is added.

Extension The extension to add.

Warning:!!!! Unlike the Pascal version of this function, you must make sure that the length of filename points to enough space to hold both FileName and Extension.

6.36.1.6. Function RemoveExtension

Code:

```
extern void RemoveExtension();
  /*     char * filename;
         char * Extension;
   */
```

Abstract:

Removes an extension from a file name if it is there.

Parameters:

FileName The file name to check. It is altered to remove the extension if it is there.

Extension The extension to remove

6.36.1.7. Function ChangeExtensions

Code:

```
extern void ChangeExtensions();
  /*     char * Name;
         char * EList;
         char * NewExt;
   */
```

Abstract:

Removes any of a list of extensions from a file name if they are there and then adds NewExt to the end.

Examples:

Name	ExtensionList	NewExt	Result
'foo'	'.PAS;.ADA;'	'.SEG'	=> 'foo.SEG'
'bar.ada'	'.PAS;.ADA;'	'.Map'	=> 'bar.Map'
'mumble.c'	'.c;.slisp;'	'.dump'	=> 'mumble.dump'

Parameters:

Name The path name to be modified

EList The list of extensions to check for and replace

NewExt The extension to add

Warning:!!! The space that name points to must be large enough to hold Name - extension + NewExt.

6.36.1.8. Function IsQuotedChar

Code:

```
extern Boolean IsQuotedChar();
  /*     char *str;
         int index;
   */
```

Abstract:

Checks whether Wild_Path_Name[Index] is quoted by preceding ' characters.

Parameters:

S — The string to check

Index — The index of character that we care about in that string

Returns:

TRUE — If the character is quoted by preceding ' characters

FALSE — Otherwise

6.36.1.9. Function Index1Unquoted

Code:

```
extern short Index1Unquoted();
  /*     char *str,ch;     */
```

Abstract:

Find the first occurrence of C in S that is not quoted. Note: The C version of this function differs from the Pascal version in that the C version returns -1 on failure (since 0 is a valid location in C).

Parameters:

str The string to search

ch The character to find; it should not be ""

Returns:

The index of the matching character, or -1 if none

6.36.1.10. Function EnvFindWildPathName

Code:

```
extern GeneralReturn EnvFindWildPathNames();
  /*     Port EnvConnection;
         char *WildPathName;
         char *ImplicitSearchList;
         Boolean FirstOnly;
         Name_Flags NameFlags;
         Entry_Type EntryType;
         Boolean *FoundInFirst;
         char *DirName;
         Entry_List *EntryList;
         int *EntryList_Cnt;
   */
```

Abstract:

Finds matches for a wildcarded name using a search list and a supplied EnvMgr connection. For each item in the search list, the name is looked up using SesScanNames. If any matches are found, the search stops and the results from SesScanNames are returned. Note that this routine may not be quite what you want -- it returns first non-empty match-set, NOT the union of all the match-sets. Think about the distinction before deciding whether this routine is appropriate for your application.

Parameters:

EnvConnection
The Environment Manager connection to use

WildPathName
The pattern to look for. Changed to be pattern found.

ImplicitSearchList
Name of the search list to use if a partial path name supplied. If blank, the list "Default" is used.

FirstOnly If true, only look in the first item of the search list. Should ordinarily be false.

NameFlags
Passed through to SesScanNames

EntryType
Passed through to SesScanNames

FoundInFirst
Returned TRUE if the first item in the search list produced the match.

DirName Passed through to SesScanNames

EntryList Passed through to SesScanNames

EntryList_Cnt
Passed through to SesScanNames

Returns:

Success

SearchlistNotFound

Any other errors returned by SesScanNames.

6.36.1.11. Function FindWildPathNames

Code:

```
extern GeneralReturn FindWildPathNames();
  /*     char * WildPathName;
         char * ImplicitSearchList;
         Boolean FirstOnly;
         Name_Flags NameFlags;
         Entry_Type EntryType;
         Boolean * FoundInFirst;
         char * DirName;
         Entry_List * EntryList;
         int * EntryList_Cnt;
   */
```

Abstract:

Finds matches for a wildcarded name using a search list. For each item in the search list, the name is looked up using SesScanNames. If any matches are found, the search stops and the results from SesScanNames are returned. Note that this routine may not be quite what you want -- it returns first non-empty match-set, NOT the union of all the match-sets. Think about the distinction before deciding whether this routine is appropriate for your application.

Parameters:

WildPathName
: The pattern to look for. Changed to be pattern found.

ImplicitSearchList
: Name of the search list to use if a partial path name supplied. If blank, the list "Default" is used.

FirstOnly
: If true, only look in the first item of the search list. Should ordinarily be false.

NameFlags

Passed through to SesScanNames

EntryType
Passed through to SesScanNames

FoundInFirst
Returned TRUE if the first item in the search list produced the match.

DirName Passed through to SesScanNames

EntryList Passed through to SesScanNames

EntryList_Cnt
Passed through to SesScanNames

Returns:

Success

SearchlistNotFound

Any other errors returned by SesScanNames.

6.36.1.12. Function EnvCompletePathName

Code:

```
extern int  EnvCompletePathName();
  /*     Port EnvConnection;
         char * WildPathName;
         char * ImplicitSearchList;
         Boolean FirstOnly;
         short * Cursor;
    */
```

Abstract:

Do filename completion on the filename indicated by WildPathName, relative to a supplied EnvMgr connection.

Parameters:

EnvConnection
: The Environment Manager connection to use

WildPathName
: The partial filename to be expanded. Changed to indicate the (partially) completed filename.

ImplicitSearchList
: Name of the search list to use if a partial path name is supplied.

FirstOnly
: If true, only look in the first item of the search list. Should ordinarily be false.

Cursor
: Position in WildPathName AFTER which the implicit '*' should be inserted. Changed to indicate the corresponding position in the expanded WildPathName. The corresponding position will never be less than the original position. Most common usage is Cursor = length(WildPathName).

Returns:

Number of names that matched WildPathName.

0 => no matches at all; WildPathName unchanged

1 => unique match; WildPathName is the expanded name.

n => several matches; WildPathName is the part that matches them all.

Warning!!!! WildPathName must be long enough to hold the expanded pathname.

6.36.1.13. Function CompletePathName

Code:

```
extern int CompletePathName();
  /*     char * WildPathName;
         char * ImplicitSearchList;
         Boolean FirstOnly;
         short * Cursor;
   */
```

Abstract:

Do filename completion on the filename indicated by WildPathName.

Parameters:

WildPathName
: The partial filename to be expanded. Changed to indicate the (partially) completed filename.

ImplicitSearchList
: Name of the search list to use if a partial path name is supplied.

FirstOnly
: If true, only look in the first item of the search list. Should ordinarily be false.

Cursor
: Position in WildPathName AFTER which the implicit '*' should be inserted. Changed to indicate the corresponding position in the expanded WildPathName. For now, the corresponding position is FORCED to be at the END of an entry name. Most common usage is Cursor = length(WildPathName).

Returns:

Number of names that matched WildPathName.

0 => no matches at all; WildPathName unchanged

1 => unique match; WildPathName is the expanded name.

n => several matches; WildPathName is the part that matches them all.

6.36.1.14. Function FindTypedName

Code:

```
extern GeneralReturn FindTypedName();
  /*     char *PathName;
         char *ExtensionList;
         char *ImplicitSearchList;
         Boolean FirstOnly;
         Entry_Type *pEntryType;
         Name_Status *pNameStatus;
   */
```

Abstract:

Finds a name of a specified type or of any type using an (optionally empty) extension list and a search list. This is the general call from which all the below "Find<xxxx>" calls are built.

For each item in the search list, the name is looked up with each extension in turn. The search proceeds across the extension list, then down the search list.

Parameters:

PathName The path name to look for. Changed to the name actually found.

ExtensionList
: The list of extensions. Each name in the extension list is terminated by a semicolon (';'). A null name implies a null extension.

ImplicitSearchList
: Name of the search list to use if a partial path name supplied. If blank, the list "Default" is used.

FirstOnly If true, only look in the first item of the search list. Should ordinarily be false.

pEntryType
Entry type being searched for. Entry_All finds first one.

pNameStatus
Passed on to SubTestName as last parameter.

Returns:

pEntryType
Returns the entry type of the name

pNameStatus
Returns the version status of the name

Returns:

Success

SearchlistNotFound

NameNotFound

ImproperEntryType

6.36.1.15. Function FindFileName

Code:

```
extern GeneralReturn FindFileName();
  /*     char * FileName;
         char * ImplicitSearchList;
         Boolean  FirstOnly;
   */
```

Abstract:

Finds a name of type Entry_File using a search list. Should be used for names of arbitrary type before doing a lookup-like operation.

Parameters:

FileName The path name to look for. Changed to the name actually found.

ImplicitSearchList
Name of the search list to use if a partial path name supplied. If blank, the list "Default" is used.

FirstOnly If true, only look in the first item of the search list. Should ordinarily be false.

Returns:

Success

SearchlistNotFound

NameNotFound

NotAFile

6.36.1.16. Function FindExtendedFileName

Code:

```
extern GeneralReturn FindExtendedFileName();
  /*     char * FileName;
         char * ExtensionList;
         char * ImplicitSearchList;
         Boolean FirstOnly;
   */
```

Abstract:

Finds a name of type Entry_File using an extension list and a search list. Should be used for names of arbitrary type before doing a lookup-like operation.

Parameters:

FileName The path name to look for. Changed to the name actually found.

ExtensionList
: The list of extensions. Each name in the extension list is terminated by a semicolon (';'). A null name implies a null extension.

ImplicitSearchList
: Name of the search list to use if a partial path name supplied. If blank, the list "Default" is used.

FirstOnly If true, only look in the first item of the search list. Should ordinarily be false.

Returns:

Success

SearchlistNotFound

NameNotFound

NotAFile

6.36.1.17. Function ReadExtendedFile

Code:

```
extern GeneralReturn ReadExtendedFile();
  /*     char *PathName;
         char *ExtensionList;
         char *ImplicitSearchList;
         File_Data *pData;
         long *pByteCount;
   */
```

Abstract:

This is equivalent to the composition of FindTypedName with SubReadFile.

Parameters:

PathName Name of file to be read

ExtensionList
List of extensions to be tried when looking for the file (for example, ".o;.i;.exe")

ImplicitSearchList
Search List to use when looking for the file

pData Points to area where data read is to be placed

pByteCount
Points to area where number of bytes read is to be stored

Returns:

Any Error from FindTypedName (unless error is ImproperEntryType, then NotAFile is returned), Success or AnyError from SubReadFile.

6.36.1.18. Function NextExtension

Code:

```
extern char *NextExtension();
  /*     char *Result;
         char *EList;
   */
```

Abstract:

Retrieves the next extension from a list of extensions which are terminated by semicolons. Removes the extension from the list.

Parameters:

EList	The list of extensions to be modified
Result	The next extension (make sure that the space that Result points to is large enough to hold any of the extensions in the list).

Returns:

The "Result" parameter (i.e., the pointer that you pass into the routine is actually returned as the value of the function)

6.36.1.19. Function SimpleName

Code:

```
extern char *SimpleName();
  /*     char *Result;
         char *PathName;
   */
```

Abstract:

Returns the terminal node of a name without a version number.

Parameters:

PathName The path name we want the terminal node for

Result Points to a place where we can put the terminal node. (Note that this must be large enough to hold the string).

Returns:

The terminal Entry_Name for the Path_Name. The Result parameter is returned through the function result.

6.36.1.20. Function ExpandPathName

Code:

```
extern GeneralReturn ExpandPathName();
  /*     char *WildPathName;
         char *ImplicitSearchList;
   */
```

Abstract:

Expands a path name into a full pathname

Parameters:

WildPathName
The path name to expand. Changed to full path name.

ImplicitSearchList
Name of the search list to use if a partial path name is supplied. If blank then the list 'Default' is used.

Returns:

Success

BadName

FirstItemNotDefined

6.36.1.21. Function FindPathName

Code:

```
extern GeneralReturn FindPathName();
  /*     char *PathName;
         char *ImplicitSearchList;
         Boolean FirstOnly;
         Entry_Type *pEntryType;
         Name_Status *pNameStatus;
   */
```

Abstract:

Finds a name using a search list.

Parameters:

PathName The path name to look for. Changed to the name actually found.

ImplicitSearchList
Name of the search list to use if a partial path name supplied. If blank, the list "Default" is used.

FirstOnly If true, only look in the first item of the search list.

EntryType Returns the entry type of the name.

NameStatus
Returns the version status of the name.

Returns:

Success

SearchlistNotFound

NameNotFound

6.36.1.22. Function ExtractSimpleName

Code:

```
extern void ExtractSimpleName();
  /*     char *name;
         short *start_terminal;
         short *start_version;
   */
```

Abstract:

Extracts some often-used components of a path name.

Parameters:

Name The path name to check

StartTerminal
Returns the index of the first character of the terminal component of the name. Will be length(Name) + 1 if the name is a directory name (ends with '>').

StartVersion
Returns the index of the version suffix of name (at the ';'). Will be length(Name) + 1 if the name has no version.

The terminal component of the name is then SubstrTo(Name, StartTerminal, StartVersion-1) and the version number string is SubstrTo(Name, StartVersion, INF)

6.36.2. PathNameDefs

Code:

```
#include <AccentType.h>
#include <SesameDefs.h>
#include <EnvMgrDefs.h>
#include <C_Types.h>


/* Path_Name:          An absolute, relative, or logical
                       non-wild pathname. */
/* Wild_Path_Name:     A potentially wild pathname.          */

typedef char Path_Name[Path_Name_Size];
typedef char Wild_Path_Name[Path_Name_Size];

#define Extension_String_Size 80


/* Extension_List:  a list of name extensions to tack onto   */
/*                  a pathname (before any version number)   */
/*                  when doing an extension search.  Each    */
/*                  extension  in the list must be terminated*/
/*                  by the semicolon (;) character.          */

typedef char Extension_List[Extension_String_Size];
```

6.37. Perq.QCodes

File Perq.QCodes.h defines the PERQ Q-code opcodes for use in C Assembly language code inserts.

Code:

```
#define LDC0      " .byte 0"   /* load constant      0    */
#define LDC1      " .byte 1"   /*    "     "         1    */
#define LDC2      " .byte 2"   /*    "     "         2    */
#define LDC3      " .byte 3"   /*    "     "         3    */
#define LDC4      " .byte 4"   /*    "     "         4    */
#define LDC5      " .byte 5"   /*    "     "         5    */
#define LDC6      " .byte 6"   /*    "     "         6    */
#define LDC7      " .byte 7"   /*    "     "         7    */
#define LDC8      " .byte 8"   /*    "     "         8    */
#define LDC9      " .byte 9"   /*    "     "         9    */
#define LDC10     " .byte 10"  /*    "     "        10    */
#define LDC11     " .byte 11"  /*    "     "        11    */
#define LDC12     " .byte 12"  /*    "     "        12    */
#define LDC13     " .byte 13"  /*    "     "        13    */
#define LDC14     " .byte 14"  /*    "     "        14    */
#define LDC15     " .byte 15"  /*    "     "        15    */
#define LDCM0     " .byte 16"  /*    "     "        -1    */
#define LDCB      " .byte 17"  /*    "     "      byte    */
#define LDCW      " .byte 18"  /*    "     "      word    */
#define LDCN      " .byte 19"  /* load long const 0 (nil)*/
#define LDLC1     " .byte 20"  /* load long constant 1    */
#define LDLCM1    " .byte 21"  /* load long constant -1   */
#define LDLCB     " .byte 22"  /* load long constant byte*/
#define LDDC      " .byte 23"  /* load long constant      */
#define LSA       " .byte 24"  /* load string address     */
#define QAND      " .byte 25"  /* integer logical and     */
#define QOR       " .byte 26"  /* integer logical or      */
#define QNOT      " .byte 27"  /* integer 1's complement */
#define QXOR      " .byte 28"  /* integer logical excl-or*/
#define EQUI      " .byte 29"  /* intg equality           */
#define NEQI      " .byte 30"  /* intg inequality         */
#define LEQI      " .byte 31"  /* intg lesser or equal    */
```

```
#define LESI      " .byte 32"  /* intg lesser              */
#define GEQI      " .byte 33"  /* intg greater or equal    */
#define GTRI      " .byte 34"  /* intg greater             */
#define ABI       " .byte 35"  /* intg absolute value      */
#define ADI       " .byte 36"  /* intg add                 */
#define NGI       " .byte 37"  /* intg negate              */
#define SBI       " .byte 38"  /* intg subtract            */
#define MPI       " .byte 39"  /* intg multiply            */
#define DVI       " .byte 40"  /* intg divide              */
#define MODI      " .byte 41"  /* intg modulus             */
#define CHK       " .byte 42"  /* intg range check         */
#define ROTSHI    " .byte 43"  /* rotate or shift integer*/
#define nCVTLI    " .byte 44"  /* long -> integer          */
#define nCVTIL    " .byte 45"  /* integer -> long          */
#define nADL      " .byte 46"  /* long add                 */
#define nSBL      " .byte 47"  /* long subtract            */
#define nEQULONG  " .byte 48"  /* long equality            */
#define nNEQLONG  " .byte 49"  /* long inequality          */
#define nLEQLONG  " .byte 50"  /* long lesser or equal     */
#define nLESLONG  " .byte 51"  /* long lesser              */
#define nGEQLONG  " .byte 52"  /* long greater or equal    */
#define nGTRLONG  " .byte 53"  /* long greater             */
#define EQNIL     " .byte 54"  /* test for equal to nil    */
#define EXCH      " .byte 55"  /* exchange e-stack intgs   */
#define EXCH2     " .byte 56"  /* exchange e-stack longs   */
#define REPL      " .byte 57"  /* replicate integer        */
#define REPL2     " .byte 58"  /* replicate long           */
#define MMS       " .byte 59"  /* push integer to m-stsck*/
#define MMS2      " .byte 60"  /* push long onto m-stack   */
#define MES       " .byte 61"  /* pop intg from m-stsck    */
#define MES2      " .byte 62"  /* pop long from m-stack    */
#define EXOP      " .byte 63"  /* two-byte ops banner      */
#define LLLB      " .byte 64"  /* load local long    byte*/
#define LLLW      " .byte 65"  /*   "     "     "      word*/
#define SLLB      " .byte 66"  /* store local long  byte   */
#define SLLW      " .byte 67"  /*   "     "     "    word  */
#define LILB      " .byte 68"  /* load intermed long byte*/
#define LILW      " .byte 69"  /*   "     "     "    word*/
#define SILB      " .byte 70"  /* store intermed long byt*/
```

```
#define SILW      " .byte 71"  /*    "       "       "  wrd*/
#define LDLB      " .byte 80"  /* load local integer byte*/
#define LDLW      " .byte 81"  /*    "     "       "    word*/
#define LDL0      " .byte 82"  /*    "     "       "      0 */
#define LDL1      " .byte 83"  /*    "     "       "      1 */
#define LDL2      " .byte 84"  /*    "     "       "      2 */
#define LDL3      " .byte 85"  /*    "     "       "      3 */
#define LDL4      " .byte 86"  /*    "     "       "      4 */
#define LDL5      " .byte 87"  /*    "     "       "      5 */
#define LDL6      " .byte 88"  /*    "     "       "      6 */
#define LDL7      " .byte 89"  /*    "     "       "      7 */
#define LDL8      " .byte 90"  /*    "     "       "      8 */
#define LDL9      " .byte 91"  /*    "     "       "      9 */
#define LDL10     " .byte 92"  /*    "     "       "     10 */
#define LDL11     " .byte 93"  /*    "     "       "     11 */
#define LDL12     " .byte 94"  /*    "     "       "     12 */
#define LDL13     " .byte 95"  /*    "     "       "     13 */
#define LDL14     " .byte 96"  /*    "     "       "     14 */
#define LDL15     " .byte 97"  /*    "     "       "     15 */
#define LLAB      " .byte 98"  /* load local address byte*/
#define LLAW      " .byte 99"  /*    "     "       "    word*/
#define STLB      " .byte 100" /* store local intg   byte*/
#define STLW      " .byte 101" /*    "     "       "    word*/
#define STL0      " .byte 102" /*    "     "       "      0 */
#define STL1      " .byte 103" /*    "     "       "      1 */
#define STL2      " .byte 104" /*    "     "       "      2 */
#define STL3      " .byte 105" /*    "     "       "      3 */
#define STL4      " .byte 106" /*    "     "       "      4 */
#define STL5      " .byte 107" /*    "     "       "      5 */
#define STL6      " .byte 108" /*    "     "       "      6 */
#define STL7      " .byte 109" /*    "     "       "      7 */
#define LLLLB     " .byte 128" /* load aligned local long*/
#define SLR0      " .byte 129" /* store long register 0  */
#define SLR1      " .byte 130" /*    "     "       "      1  */
#define SLR2      " .byte 131" /*    "     "       "      2  */
#define SLR3      " .byte 132" /*    "     "       "      3  */
#define LLR0      " .byte 133" /* load long register  0  */
#define LLR1      " .byte 134" /*    "     "       "      1  */
#define LLR2      " .byte 135" /*    "     "       "      2  */
```

```
#define LLR3      " .byte 136" /*   "     "     "      3  */
#define SIR0      " .byte 137" /* store intg register 0  */
#define SIR1      " .byte 138" /*   "     "     "      1  */
#define SIR2      " .byte 139" /*   "     "     "      2  */
#define SIR3      " .byte 140" /*   "     "     "      3  */
#define LIR0      " .byte 141" /* load intg register  0  */
#define LIR1      " .byte 142" /*   "     "     "      1  */
#define LIR2      " .byte 143" /*   "     "     "      2  */
#define LIR3      " .byte 144" /*   "     "     "      3  */
#define IXAB      " .byte 145" /* build index       byte */
#define IXAW      " .byte 146" /*   "     "         word */
#define IXA1      " .byte 147" /*   "     "           1 */
#define IXA2      " .byte 148" /*   "     "           2 */
#define IXA3      " .byte 149" /*   "     "           3 */
#define IXA4      " .byte 150" /*   "     "           4 */
#define INCB      " .byte 151" /* increment e-stack addr */
#define INCW      " .byte 152" /*     "         "     "   */
#define INDB      " .byte 153" /* load intg indirect byte*/
#define INDW      " .byte 154" /*   "     "     "    word*/
#define IND0      " .byte 155" /*   "     "     "      0 */
#define IND1      " .byte 156" /*   "     "     "      1 */
#define IND2      " .byte 157" /*   "     "     "      2 */
#define IND3      " .byte 158" /*   "     "     "      3 */
#define IND4      " .byte 159" /*   "     "     "      4 */
#define IND5      " .byte 160" /*   "     "     "      5 */
#define IND6      " .byte 161" /*   "     "     "      6 */
#define IND7      " .byte 162" /*   "     "     "      7 */
#define STIND     " .byte 163" /* store integer indirect */
#define LBLB      " .byte 164" /* load based long es+byte*/
#define LBL0      " .byte 165" /*   "     "     "  es+0   */
#define STDW      " .byte 166" /* store long indirect    */
#define LDLIND    " .byte 167" /* load long indexed      */
#define LDB       " .byte 168" /* load byt indir intg idx*/
#define LDBLong   " .byte 169" /*   "     "     "   long idx*/
#define LDBIND    " .byte 170" /*   "     "     "   byte ptr*/
#define STB       " .byte 171" /* store byt indr intg idx*/
#define STBLong   " .byte 172" /*   "     "     "  long idx*/
#define STBIND    " .byte 173" /*   "     "     "  byte ptr*/
#define LDCH      " .byte 174" /* load char indirect     */
```

```
#define STCH     " .byte 175" /* store char indirect    */
#define IXP      " .byte 176" /* build packed fld index */
#define IXPLong  " .byte 177" /* build packed fld index */
#define LDP      " .byte 178" /* load packed fld indir  */
#define STPF     " .byte 179" /* store packed fld indir */
#define SAS      " .byte 180" /* string copy            */
#define LXAB     " .byte 181" /* long index address byte*/
#define LXAW     " .byte 182" /*    "     "       "    word*/
#define LXA2     " .byte 183" /*    "     "       "    2   */
#define LXA3     " .byte 184" /*    "     "       "    3   */
#define LXA4     " .byte 185" /*    "     "       "    4   */
#define LBAR0B   " .byte 186" /* load based reg adr byte*/
#define LBAR0W   " .byte 187" /*    "     "     "   "  word*/
#define LBAR1B   " .byte 188" /* load based reg adr byte*/
#define LBAR1W   " .byte 189" /*    "     "     "   "  word*/
#define LBAR2B   " .byte 190" /* load based reg adr byte*/
#define LBAR2W   " .byte 191" /*    "     "     "   "  word*/
#define LBAR3B   " .byte 192" /* load based reg adr byte*/
#define LBAR3W   " .byte 193" /*    "     "     "   "  word*/
#define LBIR0B   " .byte 194" /* load based intg   byte */
#define LBIR1B   " .byte 195" /*    "     "      byte */
#define LBIR2B   " .byte 196" /*    "     "      byte */
#define LBIR3B   " .byte 197" /*    "     "      byte */
#define LBLR0B   " .byte 198" /* load based long   byte */
#define LBLR0W   " .byte 199" /*    "     "      word */
#define LBLR1B   " .byte 200" /*    "     "      byte */
#define LBLR1W   " .byte 201" /*    "     "      word */
#define LBLR2B   " .byte 202" /*    "     "      byte */
#define LBLR2W   " .byte 203" /*    "     "      word */
#define LBLR3B   " .byte 204" /*    "     "      byte */
#define LBLR3W   " .byte 205" /*    "     "      word */
#define IXAR0B   " .byte 206" /* intg index reg    byte */
#define IXAR0W   " .byte 207" /*    "     "      word */
#define IXAR1B   " .byte 208" /*    "     "      byte */
#define IXAR1W   " .byte 209" /*    "     "      word */
#define IXAR2B   " .byte 210" /*    "     "      byte */
#define IXAR2W   " .byte 211" /*    "     "      word */
#define IXAR3B   " .byte 212" /*    "     "      byte */
#define IXAR3W   " .byte 213" /*    "     "      word */
```

```
#define LXAR0B   " .byte 214" /* long index reg    byte */
#define LXAR0W   " .byte 215" /*   "     "     "     word */
#define LXAR1B   " .byte 216" /*   "     "     "     byte */
#define LXAR1W   " .byte 217" /*   "     "     "     word */
#define LXAR2B   " .byte 218" /*   "     "     "     byte */
#define LXAR2W   " .byte 219" /*   "     "     "     word */
#define LXAR3B   " .byte 220" /*   "     "     "     byte */
#define LXAR3W   " .byte 221" /*   "     "     "     word */
#define XJP      " .byte 222" /* jump through table      */
#define JMPB     " .byte 223" /* unconditional jump      */
#define JMPW     " .byte 224" /*      "          "        */
#define JFB      " .byte 225" /* jump if e-stack zero    */
#define JFW      " .byte 226" /*   "    "    "      "     */
#define JTB      " .byte 227" /* jump, e-stack non-zero */
#define JTW      " .byte 228" /*   "       "       "     */
#define JEQB     " .byte 229" /* jump, top two intg equ */
#define JEQW     " .byte 230" /*   "    "   "   "   "  */
#define JNEB     " .byte 231" /* jump, top two intg neq */
#define JNEW     " .byte 232" /*   "    "   "   "   "  */
#define LDRET1   " .byte 233" /* load intg c-return val */
#define LDRET2   " .byte 234" /* load long c-return val */
#define CCALL    " .byte 235" /* start C stack frame     */
#define CENTER   " .byte 236" /* finish C stack frame    */
#define CRET     " .byte 237" /* C subr return           */
#define ATPB     " .byte 244" /* add intg to top-pointer*/
#define ATPW     " .byte 245" /*   "    "   "     "      */
#define LDAP     " .byte 246" /* load activation pointer*/
#define LDTP     " .byte 247" /* load top-pointer        */
#define EVENT    " .byte 248" /* ????                    */
#define WCS      " .byte 249" /* write control store     */
#define JCS      " .byte 250" /* jump to control store   */
#define GOTOOVL  " .byte 251" /* enter micro-code overly*/
#define KOPS     " .byte 252" /* accent kernel ops       */
#define NOOP     " .byte 253" /* no-operation            */
#define BREAK    " .byte 254" /* break point trap        */
#define REFILLOP " .byte 255" /* refill op-file          */

/*************************************************************
 *
```

```
 * Spice Kernel Operations
 *
 *        There exist extensive inlinbyte's of KOPS/Kernel Op
 *        Take care in renumbering.
 *
 ****************************************************************/

#define KSVRETURN           " .byte 0"
#define KCURPROCESS         " .byte 1"
#define KHASH               " .byte 2"
#define KSETSOFT            " .byte 3"
#define KCLOCK              " .byte 4"
#define KGETCB              " .byte 5"
#define KSETVMTABLES        " .byte 6"
#define KSVCALL             " .byte 9"
#define KADDTOQUEUE         " .byte 10"
#define KREMOVEFROMQUEUE    " .byte 11"
#define KWAKEUP             " .byte 12"
#define KSLEEP              " .byte 13"
#define KUNBLOCK            " .byte 14"
#define KBLOCK              " .byte 15"
#define KLOCKSVC            " .byte 16"
#define KUNLOCKSVC          " .byte 17"
#define KSEARCH             " .byte 18"
#define KVPENTER            " .byte 19"
#define KVPREMOVE           " .byte 20"
#define KSEARCHADDR         " .byte 21"
#define KSEARCHPV           " .byte 22"
#define KCOPYPAGE           " .byte 23"

/****************************************************************
 *
 *       Extended Opcodes:
 *          Second byte of the EXOP QCode.
 *
 * Note in order to ease transition to this new order code set,
 * we keep the old encodings of LOPS/EXOPS.
 ****************************************************************/
```

```
#define INCREG0     " .byte 254" /* increment register       */
#define INCREG1     " .byte 253"
#define INCREG2     " .byte 252"
#define INCREG3     " .byte 251"
#define DECREG0     " .byte 250" /* decrement register       */
#define DECREG1     " .byte 249"
#define DECREG2     " .byte 248"
#define DECREG3     " .byte 247"
#define CVTCI       " .byte 245" /* char -> integer          */
#define CVTCL       " .byte 243" /* char -> long             */
#define SvRet2      " .byte 241" /* save long c-return value */
#define SvRet1      " .byte 239" /* save integer c-return value */
#define NGL         " .byte 3"   /* negate                   */
#define MPL         " .byte 5"   /* multiply                 */
#define DVL         " .byte 6"   /* divide                   */
#define MODL        " .byte 7"   /* modulus                  */
#define ABL         " .byte 8"   /* absolute value           */
#define LBITS       " .byte 15"  /* count asserted bits      */
#define LBNOT       " .byte 16"  /* 1's complement           */
#define LBAND       " .byte 17"  /* logical and              */
#define LBOR        " .byte 18"  /* inclusive or             */
#define LBXOR       " .byte 19"  /* exclusive or             */
#define LOPSHI      " .byte 246" /* shift                    */
#define exCHKLong   " .byte 20"  /* range check              */
#define exTNCRI     " .byte 21"  /* truncate real -> intg    */
#define exFLTIR     " .byte 22"  /* float integer -> real    */
#define exADR       " .byte 23"  /* add                      */
#define exNGR       " .byte 24"  /* negate                   */
#define exSBR       " .byte 25"  /* subtract                 */
#define exMPR       " .byte 26"  /* multiply                 */
#define exDVR       " .byte 27"  /* divide                   */
#define exRNDRI     " .byte 28"  /* round real -> int        */
#define exABR       " .byte 29"  /* absolute value           */
#define exEQUReal   " .byte 30"  /* equality                 */
#define exNEQReal   " .byte 31"  /* inequality               */
#define exLEQReal   " .byte 32"  /* lesser or equal          */
#define exLESReal   " .byte 33"  /* lesser                   */
#define exGEQReal   " .byte 34"  /* greater or equal         */
```

```
#define exGTRReal   " .byte 35"  /* greater                    */
#define exTNCQL     " .byte 36"  /* truncate quad -> long      */
#define exFLTLQ     " .byte 37"  /* float long -> quad         */
#define exADDQ      " .byte 38"  /* add                         */
#define exNEGQ      " .byte 39"  /* negate                      */
#define exSUBQ      " .byte 40"  /* subtract                    */
#define exMULQ      " .byte 41"  /* multiply                    */
#define exDIVQ      " .byte 42"  /* divide                      */
#define exRNDQL     " .byte 43"  /* round quad -> long         */
#define exABSQ      " .byte 44"  /* absolute value              */
#define exEQUQ      " .byte 45"  /* quad equality               */
#define exNEQQ      " .byte 46"  /* quad inequality             */
#define exLEQQ      " .byte 47"  /* quad lesser or equal       */
#define exLESQ      " .byte 48"  /* quad lesser                 */
#define exGEQQ      " .byte 49"  /* quad greater or equal      */
#define exGTRQ      " .byte 50"  /* quad greater                */
#define exTNCRL     " .byte 51"  /* truncate real -> long      */
#define exFLTLR     " .byte 52"  /* float long -> real         */
#define exCVQR      " .byte 53"  /* quad -> real                */
#define exCVRQ      " .byte 54"  /* real -> quad                */
#define exRNDRL     " .byte 55"  /* round real -> long         */
#define exLDQ       " .byte 56"  /* load quad                   */
#define exSTQ       " .byte 57"  /* store quad                  */
#define exEXCHQ     " .byte 58"  /* exchange quad               */
#define exPERMD     " .byte 59"  /* permute long and quad      */
#define exJLK       " .byte 60"  /* jump and link               */
#define exJMS       " .byte 61"  /* jump memory stack           */
#define exEQUStr    " .byte 62"  /* string equality             */
#define exNEQStr    " .byte 63"  /* string inequality           */
#define exLEQStr    " .byte 64"  /* string lesser or equal     */
#define exLESStr    " .byte 65"  /* string lesser               */
#define exGEQStr    " .byte 66"  /* string greater or equal    */
#define exGTRStr    " .byte 67"  /* string greater              */
#define exEQUByt    " .byte 68"  /* byte array equality        */
#define exNEQByt    " .byte 69"  /* byte array inequality      */
#define exLEQByt    " .byte 70"  /* byte array lesser/equal    */
#define exLESByt    " .byte 71"  /* byte array lesser          */
#define exGEQByt    " .byte 72"  /* byte array great/equal     */
#define exGTRByt    " .byte 73"  /* byte array greater         */
```

```
#define exEQUPowr   " .byte 74"  /* set equality              */
#define exNEQPowr   " .byte 75"  /* set inequality            */
#define exLEQPowr   " .byte 76"  /* set lesser or equal       */
#define exSGS       " .byte 77"  /* singleton set construct */
#define exGEQPowr   " .byte 78"  /* set greater or equal      */
#define exSRS       " .byte 79"  /* subrane set constructor */
#define exINN       " .byte 80"  /* set membership            */
#define exUNI       " .byte 81"  /* set union                 */
#define exQINT      " .byte 82"  /* set intersection          */
#define exDIF       " .byte 83"  /* set difference            */
#define exADJ       " .byte 84"  /* set size adjust           */
#define exEQUWord   " .byte 85"  /* multi-word equality       */
#define exNEQWord   " .byte 86"  /* multi-word inequality     */
#define exRASTOP    " .byte 87"  /* raster-op                 */
#define exSTRROP    " .byte 88"  /* string raster-op          */
#define exLINE      " .byte 89"  /* draw a line               */
#define exMVBB      " .byte 90"  /* move bytes - byte count */
#define exMVBW      " .byte 91"  /* move bytes - word count */
#define exMOVB      " .byte 92"  /* move words - byte count */
#define exMOVW      " .byte 93"  /* move words - word count */
#define exSTMW      " .byte 94"  /* store multi-word (sets) */
#define exLDMC      " .byte 95"  /* load multi-const (sets) */
#define exLDMW      " .byte 96"  /* load multi-word  (sets) */
#define exSETEXC    " .byte 97"  /* set-up exceptions         */
#define exENABLE    " .byte 98"  /* enable an exception       */
#define exQRAISE    " .byte 99"  /* raise an exception        */
#define exZEROMEM   " .byte 100" /* zero memory (kernel)     */
#define exINCDDS    " .byte 101" /* increment dds             */
#define exSTRTIO    " .byte 102" /* start i/o                 */
#define exLVRD      " .byte 103" /* Load var. rout. desc.    */
#define exLBIR0W    " .byte 104" /* load based intg    word */
#define exLBIR1W    " .byte 105" /*    "     "     "     word */
#define exLBIR2W    " .byte 106" /*    "     "     "     word */
#define exLBIR3W    " .byte 107" /*    "     "     "     word */
#define exLBQR0B    " .byte 108" /* load based quad    byte */
#define exLBQR0W    " .byte 109" /*    "     "     "     word */
#define exLBQR1B    " .byte 110" /*    "     "     "     byte */
#define exLBQR1W    " .byte 111" /*    "     "     "     word */
#define exLBQR2B    " .byte 112" /*    "     "     "     byte */
```

```
#define exLBQR2W    " .byte 113" /*   "    "    "       word */
#define exLBQR3B    " .byte 114" /*   "    "    "       byte */
#define exLBQR3W    " .byte 115" /*   "    "    "       word */
#define exIXAR01    " .byte 116" /* intg index reg 0 size 1 */
#define exIXAR02    " .byte 117" /*   "    "    "  0  "   2 */
#define exIXAR03    " .byte 118" /*   "    "    "  0  "   3 */
#define exIXAR04    " .byte 119" /*   "    "    "  0  "   4 */
#define exIXAR11    " .byte 120" /* intg index reg 1 size 1 */
#define exIXAR12    " .byte 121" /*   "    "    "  1  "   2 */
#define exIXAR13    " .byte 122" /*   "    "    "  1  "   3 */
#define exIXAR14    " .byte 123" /*   "    "    "  1  "   4 */
#define exIXAR21    " .byte 124" /* intg index reg 2 size 1 */
#define exIXAR22    " .byte 125" /*   "    "    "  2  "   2 */
#define exIXAR23    " .byte 126" /*   "    "    "  2  "   3 */
#define exIXAR24    " .byte 127" /*   "    "    "  2  "   4 */
#define exIXAR31    " .byte 128" /* intg index reg 3 size 1 */
#define exIXAR32    " .byte 129" /*   "    "    "  3  "   2 */
#define exIXAR33    " .byte 130" /*   "    "    "  3  "   3 */
#define exIXAR34    " .byte 131" /*   "    "    "  3  "   4 */
#define exLXAR01    " .byte 132" /* long index reg 0 size 1 */
#define exLXAR02    " .byte 133" /*   "    "    "  0  "   2 */
#define exLXAR03    " .byte 134" /*   "    "    "  0  "   3 */
#define exLXAR04    " .byte 135" /*   "    "    "  0  "   4 */
#define exLXAR11    " .byte 136" /* long index reg 1 size 1 */
#define exLXAR12    " .byte 137" /*   "    "    "  1  "   2 */
#define exLXAR13    " .byte 138" /*   "    "    "  1  "   3 */
#define exLXAR14    " .byte 139" /*   "    "    "  1  "   4 */
#define exLXAR21    " .byte 140" /* long index reg 2 size 1 */
#define exLXAR22    " .byte 141" /*   "    "    "  2  "   2 */
#define exLXAR23    " .byte 142" /*   "    "    "  2  "   3 */
#define exLXAR24    " .byte 143" /*   "    "    "  2  "   4 */
#define exLXAR31    " .byte 144" /* long index reg 3 size 1 */
#define exLXAR32    " .byte 145" /*   "    "    "  3  "   2 */
#define exLXAR33    " .byte 146" /*   "    "    "  3  "   3 */
#define exLXAR34    " .byte 147" /*   "    "    "  3  "   4 */
#define exPop1      " .byte 148" /* Pop EStack once          */
#define exPop2      " .byte 149" /* Pop EStack twice         */
#define exNDSTLB    " .byte 150" /* store local intg   byte */
#define exNDSTLW    " .byte 151" /*   "     "    "     word */
```

```
#define exNDSLLB    " .byte 152" /*    "      "    long    byte */
#define exNDSLLW    " .byte 153" /*    "      "     "      word */
#define exNDSIR0    " .byte 158" /* store intg register 0    */
#define exNDSIR1    " .byte 159" /*    "      "       "      1    */
#define exNDSIR2    " .byte 160" /*    "      "       "      2    */
#define exNDSIR3    " .byte 161" /*    "      "       "      3    */
#define exNDSLR0    " .byte 162" /*    "    long      "      0    */
#define exNDSLR1    " .byte 163" /*    "      "       "      1    */
#define exNDSLR2    " .byte 164" /*    "      "       "      2    */
#define exNDSLR3    " .byte 165" /*    "      "       "      3    */
#define exMathMODI  " .byte 166" /* Mathematical integer MOD*/
#define exMathMODL  " .byte 167" /* Mathematical long MOD    */
```

6.38. PMatch

File PMatch.h does pattern matching on strings. Patterns accepted are as follows:

```
"*"   matches 0 or more characters.
 "?"  matches exactly 1 character.
 "\*" matches '*', other pattern chars
         can be quoted also.
```

Note: The type pms255 (exported from the Pascal version of this module) is not available in the C version. All string parameters in the C version are of type char * and therefore the type pms255 is not needed. The user may, of course define a type "pms255" as a 255 character string if so desired.

Include Files:

```
#include <c_types.h>
```

6.38.1. Function PattDebug

Code:

```
extern void PattDebug();
  /*    Boolean v;    */
```

Abstract:

Sets the global debug flag. This causes debug statements to be printed to the standard output.

Parameters:

v	Value to set debug to

Side Effects:

Changes debug value

6.38.2. Function SetCaseFold

Code:

```
extern void SetCaseFold();
  /*    Boolean v;    */
```

Abstract:

Sets static global CaseFold marker to value specified by V. The casefold marker is used by NextCh and UpCH. If casefold then characters are forced to uppercase before being returned by these functions. Otherwise they are not.

Parameters:

v	Boolean value to set CaseFold

6.38.3. Function NextCh

Code:

```
extern char NextCh();
  /*    char *s;
        short *i;
        Boolean *sp;
   */
```

Abstract:

Returns the next character from s starting at i; decodes special characters, quoted characters and case folds if necessary

NOTE: User must set casefold (static global to this module) via SetCaseFold function before calling this routine.

Parameters:

s	String to look at
i	Index of where to start looking

sp Returns true if got a pattern matching character, else false

Returns:

The character found

Side Effects:

If s[i] is Quote_Char, then s[i+1] is returned and i is incremented by 1.

Errors:

If i > strlen(s) then reteturns ' ' with sp=TRUE.

If next character is Quote_Char and also is the last character in the string, then returns ' ' with sp=FALSE.

6.38.4. Function UpCh

Code:

```
extern char UpCh();
  /*     char *s;
         short i;
   */
```

Abstract:

Returns uppercase of s[i], if casefold is TRUE.

Parameters:

s String to look at; i is index of char wanted;

Returns:

If (i > strlen(s)) then space; else if casefold then UpperCase of the character; else the character itself

6.38.5. Function IsPattern

Code:

```
extern Boolean IsPattern();
  /*     char *str;     */
```

Abstract:

Tests to see whether str contains any pattern matching characters.

Parameters:

str String to test. Not changed.

Returns:

True if str contains any pattern matching characters; else false

6.38.6. Function PattCheck

Code:

```
extern Boolean PattCheck();
  /*     char * p1;
         char * p2;
   */
```

Abstract:

Checks to see that two strings have the same pattern match characters in the same order

Parameters:

p1 and p2 The patterns to compare

Returns:

True if have the same patterns in the same places; false otherwise

6.38.7. Function PattMatch

Code:

```
extern Boolean PattMatch();
  /*     char * str;
         char * pattern;
         Boolean fold;
   */
```

Abstract:

Compares str against pattern.

Parameters:

str	Full string to compare against pattern
pattern	Pattern to compare against. It can have special characters in it
fold	If TRUE, then "str" is forced to uppercase before comparison with patterns.

Returns:

True if string matches pattern; false otherwise

6.38.8. Function PattMap

Code:

```
extern Boolean PattMap();
  /*     char * str;
         char * inpatt;
         char * outpatt;
         char * outstr;
         Boolean fold;
   */
```

Abstract:

Compares Str against InPatt, putting the parts of Str that match Inpatt into the corresponding places in Outpatt and returning the result (in Outstr).

EXAMPLE:

PattMap("test9.pas", "*.pas", OutStr, "*.ada") => TRUE,
=> OutStr="test9.ada"

Parameters:

str	Full string to compare against pattern
inpatt	Pattern to compare against. It can have special characters in it
outpatt	Pattern to put the parts of str into; it must have the same special characters in the same order as in inpatt
outStr	The resulting string if PattMap returns true
fold	If TRUE, then "str" is forced to uppercase before comparison with patterns.

Returns:

True if string matches pattern; false otherwise

Errors:

Returns false if outPatt and inPatt do not have the same patterns in the same order (outstring will be empty).

6.39. ProcMgr/ProcMgrDefs

Files ProcMgr.h and ProcMgrDefs.h contain the routines and definitions for the process manager. See the document "The Process Manager" in the *Accent Programming Manual.*

6.39.1. ProcMgr

Code:

```
#include <AccentType.h>
#include <TSDefs.h>
#include <SapphDefs.h>
#include <ProcMgrDefs.h>

 extern void InitProcMgr ();
  /*     Port RPort */

 extern String ProcMgr_Version ( );
  /*     Port ServPort */

 extern GeneralReturn PMRegisterProcess ( );
  /*     Port ServPort;
         Port HisKPort;
         Port HisDPort;
         String ProgName;
         Window HisWindow;
         Typescript HisTypescript;
         Port EMConn;
         Port Parent;
  */

 extern GeneralReturn PMSetSignal ( );
  /*     Port ServPort;
         Port ProcPort;
         SignalName Signal;
         SignalAction Action;
   */
```

```
extern GeneralReturn PMSetSignalPort ( );
 /*     Port ServPort;
        Port ProcPort;
        Port SignalPort;
  */

extern GeneralReturn PMSetDebugPort ( );
 /*     Port ServPort;
        Port ProcPort;
        Port DebugPort;
        Boolean DebugSignalOnly;
  */

extern GeneralReturn PMSaveLoadTime ( );
 /*     Port ServPort;
        Port ProcPort;
        long LoadTime;
  */

extern GeneralReturn PMGetWaitID ( );
 /*     Port ServPort;
        Port ProcPort;
        long  * WaitID;
  */

extern GeneralReturn PMGetTimes ( );
 /*     Port ServPort;
        Port ProcPort;
        long  * LoadTime;
        long  * RunTime;
        long  * ElapsedTime;
  */

extern GeneralReturn PMGetProcPorts ( );
 /*     Port ServPort;
        Port ProcPort;
        Window  * hisWindow;
        Typescript  * hisTypescript;
        Port  * hisEMConn;
```

```
  */

extern GeneralReturn PMTerminate ( );
 /*     Port ServPort;
        Port ProcPort;
        long Reason;
  */

extern GeneralReturn PMDebugProcess ( );
 /*     Port ServPort;
        Port ProcPort;
        long Reason;
  */

extern GeneralReturn PMAddCtlWindow ( );
 /*     Port ServPort;
        Window CtlWindow;
        Window NewCtlWindow;
  */

extern GeneralReturn PMRemoveCtlWindow ( );
 /*     Port ServPort;
        Window CtlWindow;
  */

extern GeneralReturn PMChangeGroup ( );
 /*     Port ServPort;
        Port ProcPort;
        Window NewWindow;
  */

extern GeneralReturn PMSuspend ( );
 /*     Port ServPort;
        String ProcID;
  */

extern GeneralReturn PMResume ( );
 /*     Port ServPort;
        String ProcID;
```

```
  */

extern GeneralReturn PMDebug ( );
 /*     Port ServPort;
        String ProcID;
  */

extern GeneralReturn PMKill ( );
 /*     Port ServPort;
        String ProcID;
  */
extern GeneralReturn PMSetPriority ( );
 /*     Port ServPort;
        String ProcID;
        short priority;
  */

extern void PMGroupSignal ( );
 /*     Port ServPort;
        Window CtlWindow;
        SignalName Signal;
  */

extern void PMProcessSignal ( );
 /*     Port ServPort;
        Port ProcPort;
        SignalName Signal;
  *

extern GeneralReturn PMBroadcast ( );
 /*     Port ServPort;
        String s;
  */

extern GeneralReturn PMGetStatus ( );
 /*     Port ServPort;
        String ProcID;
        StatList  * Stats;
        Long  * Stats_Cnt;
```

```
   */

 extern GeneralReturn PMSignalByName ();
  /*     Port ServPort;
         String ProcID;
         Boolean ProcessOrGroup;
         SignalName Signal;
   */
```

6.39.2. ProcMgrDefs

Code:

```
#include <AccentType.h>
#include <SapphDefs.h>   /* for ProgStr, Window */

/* Const */
#define ProcMgrBase 3600 /* Process Manager messages and errors */
#define SignalBase 3800  /* Signals and asynchronous returns */

        /* Error returns from ProcMgr */

#define UnknownProcess ProcMgrBase+1
#define UnknownSignal ProcMgrBase+2
#define UnknownAction ProcMgrBase+3
#define UnknownWindow ProcMgrBase+4
#define WindowInUse ProcMgrBase+5
#define NoChildren ProcMgrBase+6
#define UnknownPort ProcMgrBase+7  /* NullPort given to ProcMgr */
#define NameAmbiguous ProcMgrBase+8
#define ProcessDisowned ProcMgrBase+9

        /* Signals */

#define MinSignal SignalBase
#define MaxSignal SignalBase+63

       /* signal numbers 1..32 are reserved for Qnix! */
```

```
#define SigSuspend MinSignal+33
#define SigResume MinSignal+34
#define SigStatus MinSignal+35
#define SigDebug MinSignal+36
#define SigLevel1Abort MinSignal+37
#define SigLevel2Abort MinSignal+38
#define SigLevel3Abort MinSignal+39


/* type */
typedef short SignalName;


        /* Actions */
#define SigDefault 0
#define SigIgnore  1
#define SigSend    2
typedef short SignalAction;


        /* Statistics returned to caller */

typedef struct statrecord
                {
                long   RunTime;      /* microseconds */
                long   LoadTime;     /* microseconds */
                long   ElapsedTime; /* ticks */
                long   Kernelport;  /* number, not port */
                short     Priority;
                short     QueueID;
                String   ProcName;  /* Process name */
                ProgStr   IconName; /* name in Icon -
                                       group name */
                ProcState State;
                } StatRecord;

typedef struct statrecord *StatArray;
typedef struct statrecord *StatList;
```

```
/*
/* Asynchronous (emergency) messages to a process
/**/

/*-------------------------------------------------------------*/
/*
/* Message sent to DebugPort to report an action on another process
/* (this is the same as the kernel's M_DebugMsg)
/*
/* Parameters:
/*   KPort      Kernel port of process affected
/*   Arg1       Always 0 (present for historical reasons)
/*   Arg2       Reason for DebugMessage:
/*                 if the process was halted by an error, this is
/*                 the GeneralReturn value describing the error
/*                 (MemFault, UncaughtException, ...)
/*                 if the process has a Process Manager Debug
/*                 Port set (by PMSetDebugPort), and a signal
/*                 was raised on the process and not sent or
/*                 ignored, this is the name of the Signal that
/*                 was raised.
/*-------------------------------------------------------------*/

typedef struct
        {
        Msg     Head;           /* Accent message header */

        TypeType    tKPort; /* (TypePT, 32) */
        Port     KPort;      /* Kernel port of process affected */

        TypeType    tArg1;  /* (TypeInt32, 32) */
        Long     Arg1;        /* What happened, field 1:
                                     always 0! */
        TypeType    tArg2;       /* (TypeInt32, 32) */
        Long     Arg2;        /* What happened, field 2:
                                     GeneralReturn value for error
                                     or signal */
        } DebugMessage;
```

```
/*-------------------------------------------------------------*/
/*
/* Message sent on process termination
/*
/* Parameters:
/*   WaitID       Wait ID of process that died, as returned
/*                    from PMGetWaitID.
/*   Reason       Reason for process death:
/*                   if killed by Terminate (in AccInt),
/*                   PROCESSDEATH (should be the REASON given to
/*                   Terminate, but the kernel must be fixed to
/*                   do this)
/*
/*                   if killed by PMTerminate(in ProcMgr),
/*                   the Reason given to PMTerminate.
/*
/*                   if the process re-registered with a
/*                   different parent, ProcessDisowned.
/*
/*                   if the process was killed by a
/*                   SigLevel[1,2,3]Abort, the Signal value.
/*
/*   LoadTime     CPU time to load process (microseconds)
/*   RunTime      CPU time to run process  (microseconds)
/*   ElapsedTime Total time elapsed from PMRegisterProcess
/*                    until termination (60Hz clock ticks).
/*
/*-------------------------------------------------------------*/

#define ProcessDeathMsgID  3800

typedef struct
        {
        Msg     Head;           /* Accent Message Header */

        TypeType    tWaitID;  /* (TypeInt32, 32) */
        long     WaitID;       /* Wait ID of process, returned
                                      from PMGetWaitID */
```

```
        TypeType    tReason;  /* (TypeInt32, 32) */
        long     Reason;       /* reason for process death */

        TypeType    tLoadTime;  /* (TypeInt32, 32) */
        long     LoadTime;  /* process Load time (microseconds) */

        TypeType    tRunTime; /* (TypeInt32, 32) */
        long     RunTime;   /* process Run time (microseconds) */

        TypeType    tElapsedTime;       /* (TypeInt32, 32) */
        long     ElapsedTime;           /* Elapsed time (ticks) */
        } ProcessDeathMsg;


/*-----------------------------------------------------------------*/
/*
/* Message sent for Signal to a process:
/* this is sent to a process's SignalPort, or to its DATAPORT if
/* the SignalPort has not been set or is NullPort.
/*
/* Parameters:
/*   CtlWindow            Window that the signal was sent from.
/*                         If the signal was sent only to one
/*                         process, it is the window that
/*         —               heads the control group for the process.
/*
/*   Signal               Signal sent to process.
/*
/*-----------------------------------------------------------------*/

#define SignalMsgID  3801

typedef struct
        {
        Msg     Head;           /* Accent message header */

        TypeType    tCtlWindow;     /* (TypePt, 32) */
        Window     CtlWindow;      /* window that signal was
```

```
                                        received on */

TypeType    tSignal;        /* (TypeInt16, 16) */
SignalName  Signal;         /* signal received */
} SignalMsg;
```

6.40. QMapDefs

File QMapDefs.h contains the constants and types used by the Q-code to source mapping facility.

Code:

```
#include <stdio.h>
#include <c_types.h>


/*-------------------------------------------------------------
* Design:
*   The QMap file is used to associate QCode numbers with source
*   file numbers.  There is one QMap entry for each statement
*   in the source file.  These are organized by procedures
*   These are organized by procedures for easier access.
*
*   The format for the QMap file is:
*
*     Block  0, 1                      : are of Type QMapDict.
*     Block  2 .. SourceFileBlock-1 : are of type QMap.
*     Block  SourceFileBlock .. EOF : are of type SourceMap.
*
*   SourceFileBlock is the field by that name in the QMapDict
*   in block 0.
*
*    Block 0 contains a CompID (which is a TimeStamp).
*    The CompID will be the time the compilation was done.
*    This will be put into Seg File also and will allow the
*    QMap file, the Sym file, and the Seg File to be
*    associated.
*
*    Also in Block 0 is the offset of the start of each
*    procedure's qmap entries.
*
*    In Block 1, the CompID and SourceFileBlock fields will be
*    zero. The number for each routine will be the offset of the
```

```
*    last entry for that routine's statements.
*
*   The SourceMap is Block Aligned.
*   The QMap entries start in block 2.  For a particular
    routine j,
*       Read Block 0 into p.  i := p.dict^.routines[j]
*       Read block (i div QMap_entries_per_block)+2 into p.
*         p.pMap^[i mod QMap_entries_per_block] is the first
*           QMap entry for that procedure.
*
*   In each QMap entry is information about the start of the text
*   in the source file for the corresponding statement.  There is
*   no way to find the end of the statement in the text file.
*   The block and offset information actually points in the source
*   file to the last character of the first identifier of the
*   statement.  Search backwards in the text file for the first
*   separator character (or the beginning of the file) to get the
*   start of the statement.
*
*   The QCode number in the QMap entry is the offset in the
*   seg file of the first qcode for this statement.  The numbers
*   are procedure relative. Some entries may not start at zero,
*   however, since there may be code in the seg file which does
*   not correspond to any user statements.  An example of this
*   is automatic initialization code for stream input and output
*   in the main body of a program.
*
*   To get the source file name for qcode entry k,
*   m := p.pMap^[k].SourceFileNumber;
*   blk := SourceFileBlock + (m div Source_entries_per_block);
*   offset := (m mod Source_entries_per_block);
*       read block blk into p.
*       file name is p.pSource^[offset].fileName
*
*   There may be more than one fileName if there are any INCLUDED
*   files since these may contain code.  The source file names
*   may take more than one block to hold.
*
*   As a check to insure that the source file found is the same
```

```
*   one used when the QMap file was created, we put the creation
*   time of the file when compiled into the QMap file entry for
*   that file.
*
*-----------------------------------------------------------------*/

#include <SesameDefs.h>

#define Max_QmapRoutines 251

typedef struct
     {
        Internal_Time CompID;
        short SourceFileBlock;
        short Routines[Max_QmapRoutines+1];
     } QMapDict;

typedef struct
     {
        short QCodeNumber;        /*a procedure relative offset
                                      of the first qcode for
                                      this statement*/
        short SourceLineNumber;  /*count of the number of
                                      carriage returns in text
                                      file before this statement*/
        short BlockNumber;        /*block offset in text file
                                      of start of statement*/
        short Offset:9;           /*byte offset in that block
                                      of the last character of
                                      the first id of the
                                      statement*/
        short SourceFileNumber:7;
     } QMapInfoRecord;


typedef struct
     {
        APath_Name FileName;
        Internal_Time CompID;
```

```
        short Filler[125];
    } SourceMapRecord;

#define WSQMapInfoRecord (sizeof(QMapInfoRecord)>>1)
#define QMap_entries_per_block 256 / WSQMapInfoRecord

#define WSSourceMapRecord (sizeof(SourceMapRecord)>>1)
#define Source_entries_per_block 256 / WSSourceMapRecord

typedef QMapInfoRecord QCodeMap[QMap_entries_per_block];
typedef SourceMapRecord SourceMap[Source_entries_per_block];
typedef short QMap_Buffer[PageWordSize];
typedef Bit8 QMap_ByteBuffer[PageByteSize];

typedef QCodeMap *pQMap;
typedef SourceMap *pSourceMap;
typedef QMap_Buffer *pQMap_Buffer;
typedef QMap_ByteBuffer *pQMap_ByteBuffer;
typedef QMapDict *pQMapDict;

typedef union
    {
        caddr_t P;
        pQMap_Buffer Buffer;
        pQMap_ByteBuffer ByteBuffer;
        pQMapDict pDict;
        /*blocks 0 .. 1 of file*/
        pQMap pMap;
        /*blocks 2 .. N-1*/
        pSourceMap pSource; /*blocks N .. EOF*/
    } QMap_ptr_type;
```

6.41. RD

File RD.h defines offsets within routine dictionary entries.

Code:

```
/* Parameter Size - number of words of parameters */
#define RDPS 0
/* Result + Parameter Size - RDPS + words of result */
#define RDRPS 1
/* Local + Temporary Size - total words of locals */
#define RDLTS 2
/* Entry point address - byte offset from code base */
#define RDEntry 3
/* Exit point address - byte offset from code base */
#define RDExit 4
/* Lexical Level */
#define RDLL 5
/* unused */
#define RDFree6 6
/* unused */
#define RDFree7 7
```

6.42. RunDefs

File RunDefs.h contains the run file definitions.

Code:

```
#include <accenttype.h>          /* PageWordSize  */
#include <sesamedefs.h>          /* APath_Name    */
#include <timedefs.h>            /* Internal_Time */

#define ModNameLength   19
#define RMaxFile         255
#define RMaxImport      4095
#define MainKludge       255
#define RFDelayedMain   255
#define RMaxSeg          (RFDelayedMain - 1)
#define RFFormat         5
                              /* Run File Format Version Number */
#define RunHeadLoc      PageWordSize    /* MinUser */

DefineString(ModString,ModNameLength);
DefineString(String20,20);

typedef unsigned char   PkSegIndex; /* Packed Form */
typedef short            SegIndex;
typedef unsigned char   PkFileIndex; /* Packed Form */
typedef short            FileIndex;
typedef short            ImpIndex;
/* Array and pointer same in C */
typedef PkSegIndex      ImpArray[4097], *pImpArray;

#define SegFileTyp 0 /* name conflict => change name -wdh */
#define RunFile 1
#define DataFile 2
#define SymsFileType 3
#define QMapFileType 4
typedef short LinkFileType;

typedef unsigned int8 PkLinkFileType; /* Packed Form */
```

```
struct FileEntry
        {
           LinkFileType   FileType;
           Long            FileLocation;
           Long            FileBlocks;
           APath_Name      FileName;
           Internal_Time   WriteDate;
           Long            Version;
        };
/* Array and pointer same in C */
typedef struct FileEntry FEArray[RMaxFile+1],*pFEArray;
struct SegEntry
        {
           ModString ModuleName;
           Long      GDBSize;          /* Size of Global Data
                                          Block */
           Long      GDBLocation;      /* can be assigned by
                                          the loader */

           FileIndex SegFile;          /* index of file
                                          containing code */
           Long      SegHdrOffset;     /* offset of .SEG hdr
                                          in that file */
           Long      SegHdrSize;       /* size of .SEG header
                                          info */

           Long      CodeOffset;       /* offset of code in
                                          the file */
           Long      CodeSize;         /* size of code
                                         (incl. RDict) */

           Long      ImportsOffset;    /* offset of imports
                                               list */
           Long      ImportsSize;      /* length of imports
                                               info */

           Long      ProcNamesOffset; /* offset of procedure
                                          names */
           Long      ProcNamesSize;    /* length of proc name
```

```
                                        entries */

        FileIndex SymsFile;         /* index of file
                                        containing SYMS info */
        Long      SymsOffset;       /* offset of .SYMS info */
        Long      SymsSize;         /* length of .SYMS info */

        FileIndex QMapFile;         /* index of file
                                      containing QMAP info */
        Long      QMapOffset;       /* offset of .QMAP info */
        Long      QMapSize;         /* length of .QMAP info */

        ImpIndex  FirstImport;      /* place in ImpArray where
                                        imports start */
        short     NumImports;       /* number of files
                                        imported */
    };
/* Array and pointer same in C */
typedef struct SegEntry SEArray[RFDelayedMain+1], *pSEArray;
struct RunHead
    {
        short          RFileFormat;

        SegIndex       InitSeg;         /* index of Pascal
                                           initializer */
        SegIndex       MainSeg;         /* index of user Main
                                           program */

        String20       LinkVersion;
        String20       LinkLocation;
        Internal_Time  LinkDate;        /* when created */
        String80       LinkCommand;

        Long           Version;         /* version of this
                                           file */

        Long           SegEntryOffset;  /* offset of SEarray
                                           in .RUN file */
        SegIndex       NumSegs;         /* number of SegEntries
```

```
                                        0..NS-1 */
        Long            ImpIndexOffset; /* offset of ImpArray
                                           in .RUN file */
        ImpIndex        NumImports;     /* number of entries
                                           in ImpArray
                                           0..NS-1 */
        Long            FileEntryOffset; /* offset of FEarray
                                            in .RUN file */
        FileIndex       NumFiles;       /* number of
                                           FileEntries
                                           0..NS-1 */
    };
typedef struct RunHead *pRunHead;
```

6.43. SaltError/SaltErrorDefs

6.43.1. SaltError

File SaltError.h exports a series of GR Error Message routines which will translate a GeneralReturn into a string error message. It handles all errors produced by the system level code: Accent, SesameFile system, CommandParse, EnvironmentMgr, ProcessMgr, and Msgserver.

By giving ErAnyError imported by commanddefs or Other imported by AccentType as the GeneralReturn value, InMsg is the message returned and only *'s and module names are added if desired.

CAUTION ** The InMsg parameter is incorporated into the GeneralReturn error message. For example, if InMsg is a filename and GR is NameNotFound, then the OutMsg returned or printed is

' * Name foo.bar not found'

Include Files:

#include <AccentType.h> #include <Spice_String.h> #include <SaltErrorDefs.h>

6.43.1.1. Function GRStdError

Code:

```
extern void GRStdError ();
  /*    GeneralReturn GR;
        GR_Error_Type ER_Type; (The return code to translate )
        char * InMsg;          (Warning, Error or Fatal Error)
        char * OutMsg;         (A filename, etc. to display  )
   */
```

Abstract:

Takes the GR value, finds the message and returns it in OutMsg. If warning, precedes message with one *. If Error or FatalError, precedes it with two **. If FatalError, exits program.

6.43.1.2. Function GRWriteStdError

Code:

```
extern void GRWriteStdError ();
  /*    GeneralReturn GR;
        GR_Error_Type ER_Type; ( The return code to translate  )
        char * InMsg;          ( Warning, Error or Fatal Error )
   */
```

Abstract:

Takes the GR value, finds the message and writes it. If warning, precedes message with one *. If Error or FatalError, precedes it with two **. If FatalError, exits program.

6.43.1.3. Function GRStdErr

Code:

```
extern Boolean GRStdErr ();
  /*    GeneralReturn GR;
        GR_Error_Type ER_Type; ( The return code to translate   )
        char * InMsg;          ( Warning, Error or Fatal Error  )
        char * OutMsg;         ( A filename, etc. to display    )
   */
```

Abstract:

Takes the GR value, finds the message and returns it in OutMsg Returns TRUE if a message has been found; otherwise FALSE. If warning, precedes message with one *. If Error or FatalError, precedes it with two **. If FatalError, exits program.

6.43.1.4. Function GRErrorMsg

Code:

```
extern void GRErrorMsg ();
  /*    GeneralReturn GR;
        GR_Error_Type ER_Type; ( The return code to translate  )
        char ProgName[];       ( Warning, Error or Fatal Error )
        char * InMsg;          ( Program name to display       )
        char * OutMsg;         ( A filename, etc. to display   )
   */
```

Abstract:

Takes the GR value, finds the message and returns it in OutMsg as <stars> Progname: Module: GrMessage If warning, precedes message with one *. If Error or FatalError, precedes it with two **. If FatalError, exits program.

6.43.1.5. Function GRWriteErrorMsg

Code:

```
extern void GRWriteErrorMsg ();
  /*    GeneralReturn GR;       ( The return code to translate  )
        GR_Error_Type ER_Type; ( Warning, Error or Fatal Error )
        char ProgName[];       ( Program name to display       )
        char * InMsg;          ( A filename, etc. to display   )
   */
```

Abstract:

Takes the GR value, finds the message and writes it as <stars> Progname: Module: GrMessage. If warning, precedes message with one *. If Error or FatalError, precedes it with two **. If FatalError, exits program.

6.43.1.6. Function ErrorMsgPMBroadcast

Code:

```
extern void ErrorMsgPMBroadcast ();
  /*     GeneralReturn GR;
         GR_Error_Type ER_Type; ( The return code to translate  )
         char ProgName[80];     ( Warning, Error or Fatal Error )
         char * InMsg;
   */
```

Abstract:

Takes the GR value, finds the message, broadcasts the message to the process manager window, and puts an ! in the icon. If warning, precedes message with one *. If Error or FatalError, precedes it with two **. If FatalError, exits program.

6.43.2. SaltErrorDefs

Code:

```
/*space used to add stars to beginning of message*/
#define BlankStarString "     "


#define GR_Warning 0
#define GR_Error 1
#define GR_FatalError 2

typedef short GR_Error_Type;


#define AccentGeneralBase       0
#define AccentSystemBase        1
#define SesameDefsBase          2
#define SesDiskDefsBase         3
#define EnvMgrDefsBase          4
#define ProcMgrDefsBase         5
```

```
#define ProcMgrDefsSignalsBase 6
#define NameErrorsBase          7
#define CmdParseBase            8
#define AuthServerBase          9
#define IOSubSystemBase        10
#define KeyTranBase            11

typedef short Base;
```

6.44. Sapphire Files (Window Manager)

Files Sapph.h, SapphDefs.h, SapphFiledefs.h, and Sapphloadfile.h contain routines and definitions for Sapphire, the window manager. Also see Viewkern, Viewpt, and WindowUtils.

The routines are described in the document "The Window Manager" in the *Accent Programming Manual.*

6.44.1. Sapph

File Sapph.h contains the window manager routines used by applications.

Code:

```
#include <AccentType.h>
#include <SapphDefs.h>
#include <ViewPt.h>

 extern void InitSapph ();
  /*     No parameters */

 extern void ModifyWindow ( );
  /*     Window ServPort;
         short newleftx;
         short newtopy;
         short newouterwidth;
         short newouterheight;
         short newRank;
   */

 extern Window CreateWindow ( );
  /*     Window ServPort;
         Boolean fixedPosition;
         short  * leftx;
         short  * topy;
         Boolean fixedSize;
```

```
        short  * width;
        short  * height;
        Boolean hasTitle;
        Boolean hasborder;
        TitStr title;
        ProgStr  * ProgName;
        Boolean hasIcon;
        Viewport  * vp;
  */

extern void SetWindowTitle ( );
 /*     Window ServPort;
        TitStr title;
  */

extern void DeleteWindow ( );
 /*     Window ServPort;     */

extern void WindowViewport ( );
 /*     Window ServPort;
        Viewport  * vp;
        short  * vpWidth;
        short  * vpHeight;
  */

extern void FullWindowState ( );
 /*     Window ServPort;
        short  * leftx;
        short  * topy;
        short  * outerwidth;
        short  * outerHeigh;
        short  * rank;
        Boolean  * hasBorder;
        Boolean  * hasTitle;
        Boolean  * isListener;
        ProgStr  * name;
        TitStr  * title;
  */
```

```
extern void GetScreenParameters ( );
 /*     Window ServPort;
        short  * width;
        short  * height;
  */

extern void EnableWinListener ( );
 /*     Window ServPort;
        Port EmergPort;
  */

extern void MakeWinListener ( );
 /*     Window ServPort;     */

extern void SetWindowAttention ( );
 /*     Window ServPort;
        Boolean attn;
  */

extern void SetWindowRequest ( );
 /*     Window ServPort;
        Boolean requesting;
  */

extern void SetWindowError ( );
 /*     Window ServPort;
        Boolean error;
  */

extern void SetWindowName ( );
 /*     Window ServPort;
        ProgStr  * ProgName;
  */

extern void SetWindowProgress ( );
 /*     Window ServPort;
        short nestLevel;
        Long value;
        Long max;
```

```
   */

extern void IdentifyWindow ( );
 /*     Window ServPort;     */

extern void CompactIcons ( );
 /*     Window ServPort;     */

extern void IconAutoUpdate ( );
 /*     Window ServPort;
        Boolean allowed;
  */

extern void GetIconViewport ( );
 /*     Window ServPort;
        Viewport  * iconvp;
        short  * width;
        short  * height;
  */

extern void DeAllocIconVP ( );
 /*     Window ServPort;     */

extern void DefineFullSize ( );
 /*     Window ServPort;
        Window exceptW;
  */

extern void ExpandWindow ( );
 /*     Window ServPort;     */

extern void ShrinkWindow ( );
 /*     Window ServPort;     */

extern void RemoveWindow ( );
 /*     Window ServPort;     */

extern void RestoreWindow ( );
 /*     Window ServPort;     */
```

```
extern void GetWinNames ( );
 /*     Window ServPort;
        pWinNameArray  * names;
        Long  * names_Cnt;
        short  * curListenIndex;
  */

extern Window WinForName ( );
 /*     Window ServPort;
        ProgStr name;
  */

extern Window GetFullWindow ( );
 /*     Window ServPort;     */

extern Window GetIconWindow ( );
 /*     Window ServPort;     */

extern Port GetWinProcess ( );
 /*     Window ServPort;     */

extern Window GetListenerWindow ( );
 /*     Window ServPort;     */

extern Window WinForViewPort ( );
 /*     Window ServPort;
        Viewport vp;
        Boolean  * isouter;
  */

extern void SetPMPort ( );
 /*     Window ServPort;
        Port ProcCtlPort;
  */

extern String Sapph_Version ( );
 /*     Window ServPort;     */
```

```
extern EventRec GetEvent ( );
 /*     Window ServPort;
        KeyHowWait howWait;
  */

extern Boolean FlushEvents ( );
 /*     Window ServPort;     */

extern void ReserveCursor ( );
 /*     Window ServPort;
        Boolean reserve;
  */

 extern void SapphException ();
 /*     No parameters     */
```

6.44.2. SapphDefs

File SapphDefs.h contains exported type definitions for the window manager (Sapphire). This includes the exports for both the window manager layer and the viewport layer. This file is included by both the server and the user side of the interface.

Code:

```
#include <stdio.h>
#include <c_types.h>
#include <AccentType.h>

/************************  VIEWPORTS ************************/

/*------ Exported Constants ------*/

#define VPRegion 1
#define OutRegion 0

#define Unchanged -32001
#define Offscreen -32002
```

```
#define DontCare -32004
#define Bottom 32000

#define NullViewPort NullPort

#define MaxNumRectangles (256/4) /*number that can fit in a page*/

#define SysFontName "Fix13.Kst"
#define SysFontHeight 13
#define SysFontWidth 9

/*------ EXPORTED types ------*/

typedef String255  VPStr255;

typedef Port Viewport;
typedef Port ViewPort;

typedef Port CursorSet;

typedef struct
     {
        short h;
        short w;
        short ty;
        short lx;
     } Rectangle;

/*used to control the cursor*/

#define cfScreenOff 0
#define cfBroken 1
#define cfOR 2
#define cfXOR 3
#define cfCursorOff 4
typedef Integer CursorFunction;

/*used in calls to GetKeyEvent to control waiting*/
```

```
#define KeyWaitDiffPos 0
#define KeyDontWait 1
#define KeyWaitEvent 2
typedef Integer KeyHowWait;

/*used in ViewLine*/

#define DrawLine 0
#define EraseLine 1
#define XORLine 2
typedef Integer LineFunct; /*used in ViewLine*/

/*used in ViewColorRect*/

#define RectBlack 0
#define RectWhite 1
#define RectInvert 2
typedef Integer RectColorFunct; /*used in ViewColorRect*/

/*used in ViewRop, ViewStrArray, etc.*/

#define RRpl 0       /* Destination gets source      */
#define RNot 1       /* Destination gets NOT source */
#define RAnd 2       /* Destination gets Destination AND
                        source        */
#define RAndNot 3    /* Destination gets Destination AND
                        (NOT source) */
#define ROr 4        /* Destination gets Destination
                        OR source */
#define ROrNot 5     /* Destination gets Destination OR
                        (NOT source)  */
#define RXor 6       /* Destination gets Destination
                        XOR source  */
#define RXNor 7
typedef Integer RopFunct; /* Destination gets Destination XOR
                             (NOT source) */

typedef short VPIntegerArray[1];
typedef short *pVPIntegerArray;
```

```
/*used in ViewCharArray*/
typedef Char VPCharArray[2];
typedef caddr_t pVPCharArray; /* pascal-ic address */

/*used in exception for viewport becoming exposed*/
typedef Rectangle RectArray[MaxNumRectangles + 1];
typedef Rectangle *pRectArray; /* array of
                          64 rectangles (one page's worth) */

typedef struct
     {
        short num;
        Port ar[1];
     } VPPortArray;
typedef VPPortArray *pVPPortArray;



/********************  KEY DEFINITIONS ************************/

/* User Input events for the 3RCC Perq */
  /* 0 -- 127     -- down key transition w/o CNTRL,
     mapped to ASCII */
  /* 128 -- 255   -- down key transition with CNTRL,
     ASCII for non-control */

/* special key tran codes.  Will set special bit.
  * used for mouse button transitions and other special ones */

#define cdYellowDown 256 /* or pen button */
#define cdWhiteDown 257
#define cdBlueDown 258
#define cdGreenDown 259
#define cdYellowUp 260
#define cdWhiteUp 261
#define cdBlueUp 262
```

```
#define cdGreenUp 263

#define cdRegionExit 264        /* generated when cursor
                                     leaves a region */
#define cdTimeout 265           /* event generated for timeouts */
#define cdPosResponse 266       /* response to Position request
                                     if same position*/
#define cdDiffPosResponse 267   /* response to Position request
                                     if diff position*/
#define cdNoEvent 268           /* response to Position request
                                    if not listener*/
#define cdListener 269          /* generated when new viewport
                                     is listener*/

#define MaxCode (127 + 16)      /* up to 16 specials, currently
                                     have 14*/

/*     typedef short KeyState    */

/* transition as stored in untranslated Event */
  /* ** WARNING ** the fields must overlay correctly!! */

typedef  union
    {
       struct {
                unsigned int Code:7;
                       /* key character coding */

                unsigned int Cntrl0n:1;
                      /* true for keyboard chars, if
                         cntrl key was down*/

                unsigned int Special:1;
                     /* true for transitions not generated
                        by keyboard, i.e., buttons or
                        regionExit, etc. */

                unsigned int  EscCl:3;
                    /* current escape class -- added
```

```
                    in KeyTran. Always zero from Sapphire. */

                unsigned int MButtons:4;
                    /* state of mouse buttons after event */

            } KeyCode;


        struct {
                unsigned int FullCode:9;
                    /* code and special and control */

                unsigned int  EscClAndMButtons:7;

          } FullKey;

        short ks;
    } KeyState;



/*      EventRec:
        an untranslated event generated directly by the user
        by typing on the keyboard or pressing a puck button,
        consisting of a keystate code, a region, a viewport,
        and a pointer position.*/

typedef struct
     { /*one untranslated event-not packed since goes into
         a message*/
       Viewport vp;
       short y, x; /* relative to VP */
       short region;
       KeyState code;
     } EventRec;


/*************************  WINDOWS ************************/
```

```
#define BorderOverhead 5
#define TitleOverhead (SysFontHeight + 6)

#define LandScapeBitWidth 1280
#define PortraitBitWidth 768
#define LandScapeBitHeight 1024
#define PortraitBitHeight 1024

#define TitStrLength (LandScapeBitWidth / SysFontWidth)

#define NullWindow NullPort
#define AskUser -32005
#define MaxCoord 16000 /*largest legal window coord*/
#define MinCoord -16000 /*smallest legal window coord*/
#define MaxSize 16000 /*largest legal window size*/

/*--- Icons ---*/
#define NumProgressBars 2
#define ProgressInTitle 1 /*the UtilProgress nest level
                            that is in the title line*/
#define IconWidth 64
#define IconHeight 64
#define ProgStrLength ((IconWidth - 2 * BorderOverhead) /
   SysFontWidth)

/*--- Exported Types ---*/

typedef Port Window;

DefineString(TitStr, TitStrLength);
DefineString(ProgStr, ProgStrLength);

typedef ProgStr *pWinNameArray;
typedef ProgStr WinNameArray[1]; /*vbl length array*/
```

6.44.3. SapphFileDefs

File SapphFileDefs.h contains type definitions for viewports available to applications, extracted from VPDefs.

Code:

```
#include <stdio.h>
#include <AccentType.h>
#include <SapphDefs.h>

/************** Font Definitions ********************/


#define FontWordWidth 48       /*word width for all fonts*/
#define FontHeadOverhead 0404 /*number of words in the
                                   header for fonts*/


typedef struct
     { /*header portion of a font*/
        short Height;              -/*Height of the KSet */
        short Base;                 /*distance from top of
                                       characters to base-line*/

        /* array indexed by ASCII of character descriptors*/
        struct CharInfo
             {
                unsigned int Offset:10;
                     /* position of character in patterns */

                unsigned int Line:6;
                     /* Line of patterns containing char */

                short Width;
                    /* Width of the character */
```

```
            } Index[128];

        short Filler[2];

        int * Pat;                  /* patterns go here */
    }     FontMapRec ;


typedef FontMapRec * FontMap;


/*****************  CURSOR DEFINITIONS ******************/

typedef short PatternMap[64][4];

typedef short * Pattern;

typedef short * pPatternMapArray;


typedef struct
    { /* Multiple cursors as a group read from a file */
       struct Type489
            {
               short NumCursors;        /* 1..127 */
               short filler;
               struct Type494
                    {
                       short x;
                       short y;
                    } Offsets[127];
            } FirstBlock;

       PatternMap cursors;   /* This is a variable length
                                array - User may have more
                                than one PatternMap.  */

    }        CursorArrayRec;
```

```
typedef CursorArrayRec * pCursorArrayRec;


typedef struct OffsetRec
     {
        short x;
        short y;
     }    OffsetPairArray[127];
```

6.44.4. Sapphloadfile

File Sapphloadfile.h contains routines for obtaining information to pass on to the window manager, such as cursor data files, font data files and Picture files.

Include Files:

#include <stdio.h> #include <SapphFileDefs.h>

6.44.4.1. Function LoadVPCursors

Code:

```
extern CursorSet LoadVPCursors ();
 /*     Viewport vp;
        char * FileName;
        short *NumCursors;
  */
```

Abstract:

Load a cursor set into memory so that the cursor can be set to it.

Parameters:

vp The user viewport

fileName The name of the file to read the cursors from

numCursors
Set to the number of cursors found in the file

Returns:

A cursor set or nullport if file is not found

Errors:

If file does not seem to be a valid cursor file (Calls GRWriteStdError)

6.44.4.2. Function LoadFontFile

Code:

```
extern Viewport LoadFontFile ();
 /*     Viewport vp;
        char * FileName;
  */
```

Abstract:

Read in a font from the disk and creates a viewport for it.

Parameters:

vp The user viewport

FileName The string name of the font (including any extensions). A normal file lookup is used to find the file.

Returns:

The font viewport for the file or NullViewPort if not found

Errors:

If cannot allocate memory for font (Calls GRWriteStdError)

6.44.4.3. Function LoadVPPicture

Code:

```
extern Viewport LoadVPPicture ();
 /*     Viewport vp;
        char * FileName;
        short width, height;
  */
```

Abstract:

Read in a picture from the disk and create a viewport for it. This is only useful when the size of the picture is known.

Parameters:

vp The parent viewport of the picture viewport

FileName The string name of the picture file (including any extensions). A normal file lookup is used to find the file. The picture is read starting at block 0 of the file.

width, height
The desired dimensions of the picture. This must be consistent with the actual picture or it will not be read in correctly.

Returns:

The viewport for the picture or NullViewport if not found

6.45. SegDefs

File SegDefs.h defines the constants and types used in the .SEG file output of the C compiler.

Code:

```
#include <oldtime.h>
#include <stdio.h>

#define QCodeVersion     4
#define SegLength        8
#define CommentLen       80
#define FileLength       100

typedef char             SNArray[SegLength];

DefineString(FNString,FileLength);
DefineString(CmString,CommentLen);

typedef int8             QVerRange;

#define Pascal 0
#define Fortran 1
#define Imp 2
typedef short Language;

typedef unsigned int PkBoolean;

typedef union
        {
         struct
                {
                 union {
                        struct {
                                Boolean ProgramSegment : 1;
                                Boolean LongIds        : 1;
                                Boolean DbgInfoExists  : 1;
```

```
                                Boolean OptimizedCode   : 1;
                                Boolean ThirtyChIds     : 1;
                                short   SegBlkFiller    : 3;
                                QVerRange QVersion      : 8;
                               } RealFields;
                        unsigned char QVerHack[2];
                       } BitHack;
                SNArray         ModuleName;
                FNString        FileName;
                short           NumSeg;
                short           ImportBlock;
                short           GDBSize;
                CmString        Version;
                CmString        Comment;
                Language        Source;
                short           PreLinkBlock;
                short           RoutDescBlock;
                short           DiagBlock;
                FNString        QMapFileName;
                FNString        SymFileName;
                TimeStamp       CompID;
               } Info;
           struct
           {
                short   OffsetRD;
                short   RoutsThisSeg;
           } Rout;
           short   Block[256];
       } SegBlock;
typedef SegBlock *pSegBlock;
typedef union
       {
           struct
           {
               SNArray ModuleName;
               FNString FileName;
           } Names;
           short   Ary[1];
       } CImpInfo;
```

```
typedef struct CImpInfo *pImpInfo;

typedef FILE *SegFileType;
```

6.46. Sesame and SesDisk Files (File System)

Files Sesame.h and SesameDefs.h contain routines and definitions used by the file system. Files SesDisk.h and SesDiskDefs.h contain the routines and definitions used in disk mounting and certain other activities.

See the document "The File System" in the *Accent Programming Manual.*

6.46.1. Sesame

Code:

```
#include <AccentType.h>
#include <SesameDefs.h>

 extern void InitSesame ();
  /*     Port RPort;      */

 extern GeneralReturn SubReadFile ( );
  /*     Port ServPort;
         APath_Name APathName;
         File_Data  * Data;
         Long  * Data_Cnt;
   */

 extern GeneralReturn SesReadFile ( );
  /*     Port ServPort;
         APath_Name  * APathName;
         File_Data  * Data;
         Long  * Data_Cnt;
         Data_Format  * DataFormat;
         Internal_Time  * CreationDate;
         Name_Status  * NameStatus;
   */

 extern GeneralReturn SubWriteFile ( );
```

```
 /*     Port ServPort;
        APath_Name  * APathName;
        File_Data Data;
        Long_Data_Cnt;
        Data_Format DataFormat;
        Internal_Time  * CreationDate;
  */

extern GeneralReturn SesGetFileHeader ( );
 /*     Port ServPort;
        APath_Name APathName;
        File_Header  * FileHeader;
  */

extern GeneralReturn SesReadBoth ( );
 /*     Port ServPort;
        APath_Name  * APathName;
        File_Data  * Data;
        Long  * Data_Cnt;
        File_Header  * FileHeader;
        Name_Status  * NameStatus;
  */

extern GeneralReturn SubLookUpName ( );
 /*     Port ServPort;
        APath_Name  * APathName;
        Entry_Type  * EntryType;
        Entry_Data  * EntryData;
        Name_Status  * NameStatus;
  */

extern GeneralReturn SubTestName ( );
 /*     Port ServPort;
        APath_Name  * APathName;
        Entry_Type  * EntryType;
        Name_Status  * NameStatus;
  */

extern GeneralReturn SubEnterName ( );
```

```
 /*     Port ServPort;
        APath_Name  * APathName;
        Entry_Type EntryType;
        Entry_Data EntryData;
  */

extern GeneralReturn SubDeleteName ( );
 /*     Port ServPort;
        APath_Name APathName;
  */

extern GeneralReturn SubReName ( );
 /*     Port ServPort;
        APath_Name OldAPathName;
        APath_Name  * NewAPathName;
  */

extern GeneralReturn SesScanNames ( );
 /*     Port ServPort;
        Wild_APath_Name WildAPathName;
        Name_Flags NameFlags;
        Entry_Type EntryType;
        APath_Name  * DirectoryName;
        Entry_List  * EntryList;
        Long  * EntryList_Cnt;
  */

extern GeneralReturn Sesame_Version ();
 /*     Port ServPort;
        String  * VersionStr;
  */
```

6.46.2. SesameDefs

Code:

```
#include        <AccentType.h>
#include        <TimeDefs.h>

/*
 * APath_Name:       A full Name Server pathname string.
 * Wild_APath_Name: A full Name Server pathname string
 *                       allowing wildcarding.
 * Entry_Name:       A single component of a pathname string.
 */

#define         Path_Name_Size      255
                  /* Number of characters in a Path_Name */

#define         Entry_Name_Size     80
                  /* Number of characters in an Entry_Name */

DefineString (APath_Name, Path_Name_Size);
                  /* An absolute pathname */

DefineString (Wild_APath_Name, Path_Name_Size);
                  /* A wild pathname */

DefineString (Entry_Name, Entry_Name_Size);
                  /* A pathname component */

/*
 * Name_Flags:  Flags giving desired treatment of names
 *       in Name Server calls.  Note that specific flag
 *       values may be illegal for certain calls, and
 *       must be zero.
 */

#define         NFlag_Deleted   0000001
                  /* Allow deleted names */
```

```
#define         NFlag_NoNormal  0000002
                  /* Disallow normal (not deleted) names */

#define         NFlag_RESERVED  0177774
                  /* These bits reserved for expansion */

typedef         short           Name_Flags;     /* 0 .. 3 */

/*
 * Name_Status:  Flags useful for determining the disposition
 *                of a name in the name data base.
 */

#define         NStat_Deleted   0000001
                  /* Set if name is deleted */

#define         NStat_High      0000002
                  /* Set if name is highest undeleted version */

#define         NStat_Low       0000004
                  /* Set if name is lowest undeleted version */

#define         NStat_RESERVED  0177770
                  /* These bits reserved for expansion */

typedef         short           Name_Status;    /* 0..7 */

/*
 * Entry_Type:  The kinds of objects which can be in the name
                 data base.
 */

#define         Entry_All       0
                  /* Special value referencing all entry types */

#define         Entry_File      1
                  /* Entry_Data is a File_ID */

#define         Entry_Directory 2
```

```
                    /* Name refers to another level of the name
                       hierarchy.  Entry_Data is empty. */

#define             Entry_Port        3
                      /* Entry_Data is a port */

typedef             short             Entry_RESERVED;
                    /* (4..0377) These values reserved for
                       expansion */

typedef             short             Entry_UserDefined;
                    /* (0400 .. 077777) Values available to the
                       user */

typedef             short             Entry_Type;
                    /* (0..077777) */

/*
 * Entry_Data:  The variant data record dependent upon the
 * Entry_Type value.
 */
typedef    union
                    {
                    Port              EDPort;
                    Char              EDBytes[256];
                    short             EDWords[128];
                    Long              EDLongs[64];
                    String255         EDString;
                    } Entry_Data;

/* typedef       FileID      EDFileID; Not yet implemented */

/*
 * Entry_List_Record:  ScanNames returns array of
 * Entry_List_Record.
 */

typedef             struct
                              {
```

```
                        Entry_Name      EntryName;
                        Long            EntryVersion;
                        Entry_Type      EntryType;
                        Name_Status     NameStatus;
                        }
                                Entry_List_Record;

typedef         Entry_List_Record       *Entry_List;
typedef         Entry_List_Record       *Entry_List_Array;
/*
 * Valid_Name_Chars:  Those 7-bit ascii characters which can
 * occur in an entry name without being quoted.  Note that
 * to match uppercase, uppercase letters also have to be quoted.
 */

#define Valid_Name_Chars  "$-.0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
 _abcdefghijklmnopqrstuvwxyz"

/*
 * Separator characters for parts of Path_Name syntax.
 */

#define     Dir_Separator   '/'     /* Directory separator */
#define     Ver_Separator   '#'     /* Version number separator */
#define     Quote_Char      '\\'    /* Quote character for
                                        "funny" characters */

#define     Up_one_Directory    "../"
#define     This_Directory      "./"

/*
 * Group_ID:  An identifier for a User Group
 */

typedef         Long            Group_ID;


/*
 * File_Data:  A pointer to a file mapped into memory
```

```
 */

typedef          caddr_t          File_Data;

/*
 * Print_Name:  A short string name in the file header.
 */

#define __        Print_Name_Size 80
                    /* Number of characters in a file
                       Print_Name */

DefineString (Print_Name, Print_Name_Size);
                  /* A print name string */

/*
 * Data_Format:  A user-defined longword for storing data
 *        format information.  System-defined values are
 *        given below.  User-defined values must have a
 *        non-zero high word (#200000 or greater).  These
 *        values are intended to be used by programs which
 *        need to know about data formats in order to
 *        convert from one format to another (such as FTP).
 */

#define  DForm_Unspecified        0
           /* Unspecified data format */

#define  DForm_8_Bit              8
           /* 8 bit binary data */

#define  DForm_16_Bit             16
           /* 16 bit binary data */

#define  DForm_32_Bit             32
           /* 32 bit binary data */

#define  DForm_36_Bit             36
           /* 36 bit binary data */
```

```
#define  DForm_CRLF_Text          0413
          /* CRLF delimited text */

#define  DForm_LF_Text            0410
          /* LF delimited text */

#define  DForm_Press              01000
          /* Press file format data */

typedef  Long     Data_Format;

/*
 * File_Header:  How the user perceives the file header
 *       information.  It is actually generated by the
 *       GetFileHeader style calls from the real header.
 */

typedef          struct
                        {
                    Long            FileSize;
                    Data_Format     DataFormat;
                    Print_Name      PrintName;
                    Group_ID        Author;
                    Internal_Time   CreationDate;
                    Group_ID        AccessID;
                    Internal_Time   AccessDate;
                    short           FHdr_RESERVED[64-56 + 1];
                        }
                        File_Header;

/*
 * Error return values
 */

#define          Sesame_Error_Base       1200

#define          NameNotFound            (Sesame_Error_Base + 1)
#define          DirectoryNotFound       (Sesame_Error_Base + 2)
```

```
#define         DirectoryNotEmpty       (Sesame_Error_Base + 3)

#define         BadName                  (Sesame_Error_Base + 4)
#define         InvalidVersion          (Sesame_Error_Base + 5)
#define         InvalidDirectoryVersion (Sesame_Error_Base + 6)
#define         BadWildName              (Sesame_Error_Base + 7)

#define         NotAFile                 (Sesame_Error_Base + 8)
#define         NoAccess                 (Sesame_Error_Base + 9)

#define         NotSamePartition        (Sesame_Error_Base + 10)
#define         ImproperEntryType       (Sesame_Error_Base + 11)
#define         NotADirectory            (Sesame_Error_Base + 12)
```

6.46.3. SesDisk

Code:

```
#include <AccentType.h>
#include <SesDiskDefs.h>
#include <SesameDefs.h>

extern void InitSesDisk ();
 /*     Port Rport;      */

extern GeneralReturn SesDiskMount ( );
 /*     Port ServPort;
        String PartName;
  */

extern GeneralReturn SesDiskDisMount ( );
 /*     Port ServPort;
        String PartName;
  */

extern GeneralReturn SesMountDevice ( );
 /*     Port ServPort;
        DiskInterface interface;
        InterfaceInfo log_unit;
```

```
        short  * unitnum;
        ptrPartDList  * PartL;
        short  * NumElts;
   */

extern GeneralReturn SesSetPOSWriteDate ( );
 /*     Port ServPort;
        APath_Name APathName;
        Internal_Time WriteDate;
   */

extern GeneralReturn SesEnterForeignSesamoid ( );
 /*     Port ServPort;
        APath_Name APathName;
        Port ForeignPort;
        APath_Name ForeignPrefix;
   */

extern GeneralReturn SesLookupForeignSesamoid ( );
 /*     Port ServPort;
        APath_Name APathName;
        Port  * ForeignPort;
        APath_Name  * ForeignPrefix;
   */

extern GeneralReturn SesMsgServerPort ( );
 /*     Port ServPort;
        Port MsgServerPort;
   */

extern GeneralReturn SesGetNetPort ( );
 /*     Port_ServPort;
        Port  * SesNetPort;
   */

extern GeneralReturn SesGetSegID ( );
 /*     Port ServPort;
        APath_Name  * APathName;
        SegID  * SegmentID;
```

```
  */

extern GeneralReturn SesEnterSegID ( );
 /*     Port ServPort;
        APath_Name  * APathName;
        SegID SegmentID;
  */

extern GeneralReturn SesGetDiskPartitions ( );
 /*     Port ServPort;
        ptrPartDList  * PartL;
        short  * NumElts;
  */

extern GeneralReturn SesGetSegName ( );
 /*     Port ServPort;
        SegID SegmentID;
        APath_Name  * APathName;
  */

extern GeneralReturn SesGetAccessRights ( );
 /*     Port ServPort;
        APath_Name  * APathName;
        User_ID  * Owner;
        Access_Rights  * Rights;
  */

extern GeneralReturn SesSetAccessRights ( );
 /*     Port ServPort;
        APath_Name  * APathName;
        User_ID NewOwner;
        Access_Rights NewRights;
  */

extern GeneralReturn SesGetDefaultAccess ( );
 /*     Port ServPort;
        Access_Rights  * DefaultAccess;
  */
```

```
extern GeneralReturn SesSetDefaultAccess ( );
 /*     Port ServPort;
        Access_Rights NewDefaultAccess;
  */

extern GeneralReturn SesAuthServerPort ( );
 /*     Port ServPort;
        Port AuthServerPort;
  */

extern GeneralReturn SesConnect ( );
 /*     Port ServPort;
        Port RegPort;
  */

extern GeneralReturn SesDirectio ( );
 /*     Port ServPort;
        DirectIOArgs  * CmdBlk;
        Header  * DataHdr;
        DiskBuffer  * Data;
  */

extern GeneralReturn SesDisk_Version ( );
 /*     Port ServPort;
        String  * VerString;
  */
```

6.46.4. SesDiskDefs

Code:

```
#include <stdio.h>
#include <c_types.h>


#include <AccentType.h>
#include <AuthDefs.h> /* User_ID */
```

```
/**/
/* Partition list
**/


typedef struct
    {
     DevPartString PartName;   /* name of this partition       */
     DevPartString DevName;    /* which disk this part is on   */
     SegID PartRootDir;        /* SegID of Root Directory      */
     long PartNumFree;         /* HINT of how many free pages */
     Boolean PartMounted;      /* this partition is mounted    */
     DiskAddr PartStart;       /* Disk Address of 1st page     */
     DiskAddr PartEnd;         /* Disk Address of last page    */
     PartitionType PartKind;   /* Root or Leaf                 */
     Boolean PartExUse;        /* Opened exclusively           */
    } PartData;          /* entry in the PartTable */

typedef PartData PartDList[MAXPARTITIONS];
typedef PartDList *ptrPartDList;


/**/
/* Error return codes
**/

#define SesDisk_Error_Base 29400

#define BadPartitionName SesDisk_Error_Base + 1
#define BadPartitionType SesDisk_Error_Base + 2

#define AuthorizationServerGone SesDisk_Error_Base + 3

/**/
/* Temporary Entry_Type values
**/

#define Entry_Foreign 6
```

```
/**/
/* Temporary Access Control flags
**/
#define Owner_Read_Access 01
#define Owner_Write_Access 02

#define All_Read_Access 0100
#define All_Write_Access 0200

typedef short Access_Rights;

/**/
/* The temporary access control flags are returned in one
*  of the "reserved" fields in the file header.
**/

#define FHdr_Access_Rights_Offset 56
```

6.47. Spawn/SpawnDefs/SpawnInitFlags

Files Spawn.h, SpawnDefs.h, and SpawnInitFlags create a new process and notify the Process Manager of its existence.

See the document "The Pascal Library" in the *Accent Languages Manual.*

6.47.1. Spawn

Include Files:

#include <AccentType.h> #include <CommandDefs.h> #include <SesameDefs.h>

#include <SpawnDefs.h>

6.47.1.1. Function GeneralReturnSpawn

Code:

```
extern GeneralReturn Spawn();
 /*     Port *ChildKPort;
        Port *ChildDPort;
        char *ProgName;
        char *ProcName;
        CommandBlock *HisCommand;
        Boolean DebugIt;
        Boolean ProtectChild;
        ConnectionInheritance SapphConn;
        Port pWindow;
        Port pTypeScript;
        ConnectionInheritance EMConn;
        Port pEMPort;
        Port *PassedPorts;
        Long NPorts;
        Boolean LoaderDebug;
 */
```

Abstract

Spawn calls redirect_spawn with given parameters and InputFile and OutputFile set to "CONSOLE:".

Parameters:

See redirect_spawn below.

6.47.1.2. Function redirect_spawn

Code:

```
extern GeneralReturn redirect_spawn();
 /*     Port *ChildKPort;
        Port *ChildDPort;
        char *ProgName;
        char *ProcName;
        CommandBlock *HisCommand;
        Boolean DebugIt;
        Boolean ProtectChild;
        ConnectionInheritance SapphConn;
        Port pWindow;
        Port pTypeScript;
        ConnectionInheritance EMConn;
        Port pEMPort;
        Port *PassedPorts;
        Long NPorts;
        char *InputFile;
        char *OutputFile;
        Boolean LoaderDebug;
   */
```

Abstract:

This is the process creation routine.

Parameters:

ChildKPort

Will be set to the Child's KernelPort in the Parent

ChildDPort
: Will be set to the Child's DataPort in the Parent

ProgName
: The name of the .RUN file that you want loaded and executed

ProcName
: The name of the child process as it should appear in a SYSTAT, usually the same as ProgName

HisCommand
: Will be used to set the child's UserCommand

DebugIt
: If true, the new process is suspended and Mace is invoked on it before it gets control. This isn't used when just forking.

ProtectChild
: If true, the child process will not have access to physical memory or the ability to do I/O. If false, the child will inherit the parent's access capabilities.

SapphConn
: Controls access to Sapphire and to the typescript.

  If 'NewOne', then Sapphire will be asked for a window, and a new typescript will be created in that window. The window's name is set to ProcName.

  If 'Given', the Window and Typescript will be taken from the pWindow and pTypescript. They will still belong to the parent process. The window's name is UNChanged.

  'GivenReg' takes the Window and Typescript from pWindow and pTypescript, but gives their ownership to the child. The window's name is set to ProcName.

pWindow The window port to be given to the newly created process if SapphConn is "Given"

pTypeScript
The user typescript port to be given to the newly created process if SapphConn is "Given"

EMConn This controls access to the environment manager.

If 'NewOne', then a new EnvMgr connection will be made, copying the state of the parent's connection (NOT pEMPort!). The new EnvMgr connection will be given to the child's Typescript.

If 'Given', then the connection will be taken from pEMPort, but the parent will still own it.

'GivenReg' takes the connection from pEMPort, but gives its ownership to the child.

For 'Given' or 'GivenReg', if the typescript already exists, it is assumed that it has a valid EnvMgr connection. If the typescript is new (SapphConn is 'NewOne'), it is given the connection from pEMPort.

pEMPort The environment manager connection (port) to ge given to the newly created process if EMConn is "Given" or "GivenReg".

PassedPorts
An array of ports to pass to the child. This contains only the ports above and beyond any system ports that you want to pass to the child, and will be available in InPorts, with the same indexes as in this array. Everything else that you used to have to set by hand is now controlled by other parameters to spawn.

NPorts The number of ports being passed in PassedPorts

InputFile Sent to the newly created process as the DefaultInputName in the initial message

OutputFile
Sent to the newly created process as the DefaultOutputName in the initial message

LoaderDebug
Turns on Spawn and loader debugging

Returns:

In the fork case: IsParent, IsChild

In the exec case: Success, Failure

NOTES: If 'NewOne' is not specified as the SapphConn parameter, an EnableWinListener will have to be issued (if the parent hasn't already done it). When 'NewOne' is used, Spawn will make a window and typescript and set up the typescript's keytranslation table.

6.47.2. SpawnDefs

Code:

```
typedef short ConnectionInheritance;

#define NewOne   0
#define Given   1
#define GivenReg 2
```

6.47.3. SpawnInitFlags

WARNING

This module differentiates between starting processes from the boot file and starting all other processes after that. There is a version of this file used by 'system' and a version used by 'pascalinit'. This module affects 'pascalinit' and 'spawn' and 'typescript.'

Code:

```
#define FALSE 0
#define TRUE  1

#define  ForBootFile      FALSE
#define  ForLibFile       (! ForBootFile)
```

6.48. Spice_String/Spice_StringDefs

6.48.1. Spice_String

Files Spice_String.h and Spice_StringDefs.h implement string hacking routines compatible with Spice_String.pas in LibPascal. See the document "The Pascal Library" in the *Accent Languages Manual*.

String constants should not be used as parameters for any Spice_String routine that lengthens the size of the string. Instead, space should be allocated (via malloc) and strcpy used to load the variable.

Warning: Be aware of the danger involved in mixing C library string routines with C Spice_String routines. In most cases, C Spice_String malloc's space and returns pointers to the string in the malloc'ed space. In contrast, most C library string functions do not malloc space and expect the user to have provided parameters of the appropriate size.

Include Files:

```
#include <stdio.h> #include <Spice_StringDefs.h>
```

6.48.1.1. Function Adjust

Code:

```
extern char * Adjust ();
  /*     char *s1;
         int len;
   */
```

Abstract:

This function changes the length of a string.

Parameters:

s1 The string to be changed

Len The new length of the string (This doesn't include the null character.)

Result:

A pointer to the string of new length is returned.

6.48.1.2. Function AppendChar

Code:

```
extern char * AppendChar ();
  /*     char *s1;
         char c1;
   */
```

Abstract:

Puts c1 on the end of s1.

Parameters:

s1 The left string and c1 goes on the end

Side Effects:

Modifies s1.

Errors:

s1's space must be large enough to append another character.

6.48.1.3. Function AppendString

Code:

```
extern char * AppendString ();
  /*     char *s1;
         char *s2
   */
```

Abstract:

Puts s2 on the end of s1.

Parameters:

s1 The left string and s2 is the right string

Side Effects:

Modifies s1.

Errors:

s1's space must be large enough to append s2.

6.48.1.4. Functions Cat3, Cat4, Cat4, Cat6

Code:

```
extern char * Cat3 ();
  /*     char *s1, *s2, *s3;     */


extern char * Cat4 ();
  /*     char *s1, *s2, *s3, *s4;     */


extern char * Cat5 ();
  /*        char *s1, *s2, *s3, *s4, *s5;     */
```

```
extern char * Cat6 ();
  /*        char *s1, *s2, *s3, *s4, *s5, *s6;     */
```

Abstract:

These functions concatenate multiple strings together.

Parameters:

Strings to Concatenate.

Returns:

Returns a pointer to a single string composed of the parameters.

6.48.1.5. Function CheckStr

Code:

```
extern char CheckStr ();
  /*     char *s1, m1;     */
```

Abstract:

This function is called by PosString to check for equal characters in mask and source strings. PosString finds a possible match by looking at first chars while CheckStr finishes the check.

Parameters:

s1 The source string and m1 is the mask

Result:

A char is returned--'t' for a match, 'f' for no match.

6.48.1.6. Function Concat

Code:

```
extern char * Concat ();
  /*     char *s1, *s2;     */
```

Abstract:

Concatenates two strings together.

Parameters:

s1 and s2 The two strings to be concatenated

Returns:

Returns a pointer to a single string as described by the parameters.

6.48.1.7. Function ConvUpper

Code:

```
extern char * ConvUpper ();
  /*     char *s1;     */
```

Abstract:

Converts s1 to all upper case.

Parameters:

s1 To be converted

Side Effect:

Returns the pointer to a modified s1.

Design:

Uses macro calls included in ctype.h.

6.48.1.8. Function CVD

Code:

```
extern short CVD ();
  /*     char *s1;     */
```

Abstract:

Converts a decimal string to an integer. Conversion stops at the first character that is not legal in the radix used. Characters <= space which precede the value are ignored.

Parameter:

S1 The string to be converted

Returns:

A short integer is returned (16 bits).

6.48.1.9. Function CVH

Code:

```
extern short CVH ();
  /*     char *s1;     */
```

Abstract:

Converts a hexadecimal string to an integer. Conversion stops at the first character that is not legal in the radix used. Characters <= space which precede the value are ignored. Lower case 'a' .. 'f' are the same as upper case 'A'..'F'.

Parameters:

s1 The string to be converted

Result:

A short integer is returned (16 bits).

6.48.1.10. Functions CVHS, CVOS, CVS

Code:

```
extern char * CVHS();
  /*    int i;    */

extern char * CVOS ();
  /*    int i;    */


extern char * CVS ();
  /*    int i;    */
```

Abstract:

All three functions convert an integer to a string of width w, with no padding on the left. The radix will be as follows: CVHS: Hexidecimal; CVOS: Octal; CVS: Decimal.

Note: These three functions are the same as CVHSS, CVOSS, and CVSSS except that no padding will be done.

Parameter:

i The integer to be converted

Result:

A string containing the character representation. The string will be only as long as needed (no padding) and will be converted to the proper radix. CVHS is hex conversion; CVOS is octal conversion; CVS is decimal.

6.48.1.11. Functions CVHSS, CVOSS, CVSS

Code:

```
extern char * CVHSS ();
  /*    int i,w;    */

extern char * CVOSS ();
  /*    int i,w;    */

extern char * CVSS ();
  /*    int i,w;    */
```

Abstract:

All three functions convert an integer to a string of width w, padding on the left with spaces if necessary to fill out the width. The radix will be as follows: CVHSS: Hexidecimal; CVOSS: Octal; CVSS: Decimal.

Note: These functions are the same as CVHS, CVOS, and CVS except for the padding.

Parameters:

i	The integer to be converted
w	The minimum field width to be produced

Result:

A string containing the character representation; the string will be of length w, filled on the left with spaces, and the conversion done in the proper radix.

6.48.1.12. Function CvInt

Code:

```
extern short CvInt ();
  /*     char *s1;
         int radix;
   */
```

Abstract:

Converts a string to a short int according to base Radix. Conversion stops at the first character that is not legal in the radix used. Characters <= space which precede the value are ignored. A sign is permitted. Lower case 'a'..'z' are the same as upper case 'A'.. 'Z'.

Parameters:

s1 The string to be converted

radix The base to be used

Returns:

Returns an integer containing the value.

Calls: CvL.

6.48.1.13. Function CvL

Code:

```
extern long CvL ();
  /*     char *s1;
         int radix;
   */
```

Abstract:

Converts a string to a long integer according to base Radix. Conversion stops at the first character that is not legal in the radix used. Characters <=space which precede the value are ignored. A sign is permitted. Lower case 'a' .. 'z' are the same as 'A' .. 'Z'.

Parameters:

s1	The string to be converted
radix	The radix to be used

Returns:

A long integer containing the value

6.48.1.14. Function CVLS

Code:

```
extern char * CVLS ();
  /*    int w, base;
        char fill;
        int i;
   */
```

Abstract:

Converts a long to a string of width w, padding on the left with fill if necessary. The base for the conversion is radix. If Radix>10, the letters 'A'..'Z' will be used to compute the character for the representation. Using a base > 36 will produce bogus results. A negative base will force an unsigned result. (This differs only from CVN by i being a long or integer rather than a short.)

Parameters:

i	Short integer to be converted
w	Minimum field width to be produced

base Base to use(2..36)

fill A character

Returns:

Returns a pointer to a string containing the character representation; the string will be of at least width w, filled on the left with Fill, and converted according to radix Base.

Calls: Pad, Str.

6.48.1.15. Function CVN

Code:

```
extern char * CVN ();
  /*    int w, base;
        char fill;
        short i;
   */
```

Abstract:

Converts a short to a string of width w, padding on the left with fill if necessary. The base for the conversion is radix. If Radix>10, the letters 'A'..'Z' will be used to compute the character for the representation. Using a base > 36 will produce bogus results. A negative base will force an unsigned result.

Parameters:

i Short integer to be converted

w Minimum field width to be produced

base Base to use(2..36)

fill A character

Returns:

Returns a pointer to a string containing the character representation; the string will be of at least width W, filled on the left with Fill, and converted according to radix Base.

Calls: Pad, Str.

6.48.1.16. Function CVO

Code:

```
extern short CVO ();
  /*     char *s1;     */
```

Abstract:

Converts an octal string to an integer. Conversion stops at the first character that is not legal in the radix used. Characters <= space which precede the value are ignored.

Parameters:

S1 String to be converted

Result:

A short integer is returned.

6.48.1.17. Function CvUp

Code:

```
extern char * CvUp ();
  /*     char *s1;     */
```

Abstract:

Returns a pointer to a copy of s1 with lower case replaced by upper.

Parameter:

s1 String to be converted

Design:

Uses macro calls included in ctype.h

6.48.1.18. Function DeleteChars

Code:

```
extern void DeleteChars ();
  /*    char *s1;
        int index, size;
   */
```

Abstract:

Removes characters from a string.

Parameters:

S1 String to be changed. Characters will be removed from this string.

Index Starting position for the delete. (Counting numbers used for index. The digit 0 is not a valid index.)

Size Number of characters that are to be removed. Size characters will be removed from s1 starting at index.

Returns:

This function does not return anything.

Side Effects:

s1 is modified.

Errors:

Index and Size must be valid parameters.

6.48.1.19. Function GetBreak

Code:

```
extern btype GetBreak ();
  /*     No Parameters     */
```

Abstract:

Allocates and clears a break table.

Parameters:

None

Returns:

Returns a pointer of btype. This points to a breaktable containing flags, break characters, and omit characters as described in SetBreak. The user will need to do a GetBreak followed by a SetBreak.

6.48.1.20. Function Initial

Code:

```
extern int Initial ();
  /*     char *s1, *s2;     */
```

Abstract:

This function returns an integer value of 1 if s2 is an initial string of s1. The comparison is case-sensitive. A null string is an initial substring of any string.

Parameters:

s1	String to be tested

s2 Questionable initial string

Returns:

Returns the integer 1 if s2 is an initial substring of s1; otherwise returns the integer 0.

6.48.1.21. Function InsertChars

Code:

```
extern void InsertChars ();
  /*    char *srcstr, *dststr;
        int index;
   */
```

Abstract:

This function inserts a string into the middle of another string.

Parameters:

srcstr String that is to be inserted

dststr String into which the insertion is to be made

index Starting position, in dststr, for the insertion

Returns:

This function does not return anything.

Side Effects:

dststr is modified.

Errors:

Dststr's space must be large enough to insert the additional characters. Index must be valid.

6.48.1.22. Function Lop

Code:

```
extern char Lop ();
  /*     char **s1;     */
```

Abstract:

Remove the first character from s1 and returns it as the value of the function. S1 no longer contains the character.

Parameters:

s1 String from which a character will be lopped off

Returns:

Returns the first character of s1.

Side Effects:

Modifies s1.

6.48.1.23. Function Pad

Code:

```
extern char * Pad ();
  /*     char *s1;
         int totl, where;
         char padc;
   */
```

Abstract:

Produces a copy of s1 adjusted to have length totl by inserting sufficient copies of padc just after location where.

Parameters:

s1 Original string

totl Desired length for result string

padc Char to insert to achieve desired length

where Where to insert chars

Use 0 for padding on left and strlen(s1) for padding on the right. If some other value, padding will be done just after the character in that position.

Returns:

A pointer to a string of length totl consisting of a copy of s1 with sufficient copies of padc inserted just after position where. S1 is not modified.

Design:

Uses strcpyto.

6.48.1.24. Function PosC

Code:

```
extern int PosC ();
  /*     char *s1, chr;     */
```

Abstract:

Finds the position of C in s1. Returns 0 if absent.

Parameters:

s1 String to be searched

chr Character we are looking for

Returns:

If chr occurs in s1, then the index into s1 of the first character matching chr will be returned. If chr was not found (even if because s1 = '') then return 0.

6.48.1.25. Function PosString

Code:

```
extern int PosString ();
  /*     char *source, *mask;     */
```

Abstract:

Finds the position of Mask in the substring of source.

Parameters:

Source String to be searched

Mask Pattern that we are looking for

Result:

If Mask occurred in source then the index into (the original) source of the first character matching the mask will be returned. If Mask was not found then return 0. If Mask is '', return 1.

6.48.1.26. Function ReplaceChars

Code:

```
extern char * ReplaceChars ();
  /*     char *s1, *newstr;
         int index;
   */
```

Abstract:

Replaces a substring of s1 with another string.

Parameters:

s1 String into which the replacement is to be made

newstr String that is to replace the deleted segment

index Starting position in s1 for newstr. If it is greater than strlen(s1), no replacement will be made. If it is less than zero, start at one.

Returns:

The function returns a string of the same length with the appropriate replacement.

Side Effects:

Modifies s1.

6.48.1.27. Function RevPosC

Code:

```
extern int RevPosC ();
  /*    char *s1, chr;    */
```

Abstract:

Tests if chr is a member of s1.

Parameters:

s1 String to be searched

chr Character you are looking for

Returns:

Returns the index of first occurrence of chr in s1 (from the end of s1) or zero if not there.

6.48.1.28. Function RevPosString

Code:

```
extern int RevPosString ();
  /*     char *source, *mask;     */
```

Abstract:

Finds the position of Mask in the source string.

Parameters:

source String to be searched

mask Pattern we are looking for

Result:

If Mask occurred in source then the index (from the end) into (the original) source of the first character matching the mask will be returned. If Mask was not found then return 0. If mask is '', return 1.

6.48.1.29. Function Scan

Code:

```
extern char * Scan ();
  /*     char **scanstr, **brk;
         btype btp;
   */
```

Abstract:

Scans the string scanstr according to the breaktable specifications of btp.

Parameters:

Scanstr Address of the pointer to the string to be scanned

Btp — Points to the table containing the scanning info as filled by SetBreak

Brk — Address of the pointer to the string to contain the break character on which the scan stopped

Returns:

The initial substring determined by the breaktable s removed from scanstr and returned (by a pointer).

6.48.1.30. Function SetBreak

Code:

```
extern btype SetBreak ();
  /*    btype btp;
        char *breakc, *omit;
        char cc, casec, inorex;
   */
```

Abstract:

Initializes a breaktable according to the specifications of breakt, omit, cc, casec, and inorex.

Parameters:

Breakt — Set of characters (as a string) on which a scanning break will occur

Omit — Set of characters (as a string) which will be removed from the string

Cc — Signals how the break character and strings will be handled: (s=skip, a=append, r=retain)

cc = 's': Upon return, the break character will be in the break variable. The result of the scan will be all characters up to the

break character, and the input string is modified to start immediately after the break character.

cc = 'a': Upon return, the break character will be in the break variable. The result of the scan will be all characters up to and including the break character, and the input string is modified to start immediately after the break character.

cc = 'r': Upon return, the break character will be in the break variable. The result of the scan will be all characters up to the break character, and the input string is modified to start at the break character.

cc = anything else: 's' is assumed.

Casec — Will either be 'l', 'u', or ' ' for lower case, upper case, or no case respectively. Folding of the char to upper case or to lower case will occur before the break or omit check is executed. (Default for bad casec value is no case.)

Inorex — Determines the set of characters on which a break will occur. Inorex = 'i' is inclusive; Breakt is is the set of characters on which a break will occur. Inorex = 'e' is exclusive; Breakt is the set of characters on which a break will not occur.

Btp — Pointer to a break table or record. (The break table contains a flag for case, how to handle break char, whether breakt is inclusive or exclusive, and a string to identify break and omit chars.) This is returned from GetBreak.

Returns:

A pointer to a break table containing the information needed to start scanning strings.

6.48.1.31. Function ShowBreak

Code:

```
extern void ShowBreak ();
  /*     btype bpt;     */
```

Abstract:

This function will print the breaktable information.

Parameter:

Btp Pointer to a break table

Result:

Only prints out the break table.

6.48.1.32. Function Squeeze

Code:

```
extern char * Squeeze ();
  /*     char *s1;     */
```

Abstract:

Removes all spaces and tabs from s1.

Parameters:

s1 String from which spaces and tabs will be removed

Result:

A pointer to a copy of s1 which has spaces and tabs removed. This function will not modify s1.

6.48.1.33. Function Str

Code:

```
extern char * Str ();
  /*     char chr     */
```

Abstract:

This function is used to coerce a character to a string.

Parameters:

chr Character to be coerced into a string

Returns:

A pointer to the new string is returned.

6.48.1.34. Function strcmpm

Code:

```
extern int strcmpm ();
  /*     char *s1, s2;     */
```

Abstract:

Compares two strings and returns an integer result. The test is case-sensitive.

Parameters:

s1 and s2 The two strings to be compared

Returns:

Returns 0 if the two strings are equal.

Returns an integer > 0 if s1 > s2.

Returns an integer < 0 if s1 < s2.

6.48.1.35. Function strcpyfor

Code:

```
extern char * strcpyfor ();
  /*     char *s1, **s2;
         int index, size;
   */
```

Abstract:

This function will copy s1 into s2 beginning at index until s2 = size. Unlike SubStrFor, the user will have previously allocated space for s2.

Parameters:

s1	Source string
s2	Address of where the duplicate string will be stored
Index	Starting position in s1 for the copy
Size	Resulting length of s2

Returns:

This function returns a pointer to the beginning of s2. If Index+Size exceed the dynamic length of the string, return Index to DynamicLength. S2 is modified.

6.48.1.36. Function strcpyto

Code:

```
extern char * strcpyto ();
  /*     char *s1, **s2;
         int index, endindex;
   */
```

Abstract:

This function will copy s1 into s2 beginning at index and stopping at endindex. Unlike SubStrTo, the user will have previously have allocated space for s2.

Parameters:

s1	Source string
s2	Address of where the duplicate string will be placed
Index	Starting position in s1 for the copy
Endindex	Stopping position in s2 for the copy

Returns:

This function returns a pointer to s2 as described by the above parameters. If Index or EndIndex exceed the dynamic length of the string, return Index to DynamicLength. This function modifies s2.

6.48.1.37. Function Strip

Code:

```
extern char * Strip ();
  /*     char *s1;     */
```

Abstract:

Converts sequences of spaces, tabs, CR, and LF to a single space.

Parameter:

s1	String to be stripped

Returns:

Returns a pointer to a copy of s1 with spaces, tabs, CR, and LF converted to a single space. s1 is not modified.

6.48.1.38. Function SubStrFor

Code:

```
extern char * SubStrFor ();
  /*     char *source;
         int index, size;
   */
```

Abstract:

This procedure is used to return a subportion of the string passed as a parameter.

Parameters:

Sourcethat we are to take a portion of

Index	Starting position in Source of the substring
Size	Size of the substring that we are to take

Returns:

This function returns a pointer to the substring as described by the above parameters.

If Index+Size exceed the dynamic length of the string, return Index to DynamicLength; no error message is generated.

6.48.1.39. Function SubStrTo

Code:

```
extern char * SubStrTo ();
  /*     char *source;
         int index, endindex;
   */
```

Abstract:

This procedure is used to return a subportion of the string passed as a parameter.

Parameters:

Source String that we are to take a portion of

Index Starting position in source of the substring

Endindex Ending position in the source of the substring

Returns:

This function returns a pointer to a substring as described by the above parameters. Source is not modified. If Index or EndIndex exceed the dynamic length of the string, return Index to DynamicLength.

6.48.1.40. Function Trim

Code:

```
extern char * Trim ();
  /*     char *s1;     */
```

Abstract:

Deletes leading and trailing spaces, tabs, CR, and LF's from a string.

Parameters:

S1 String to be shortened

Result:

A pointer to a copy of s1 (with leading and trailing spaces, CR's, LF's, and tabs removed) is returned.

6.48.1.41. Function ULInitial

Code:

```
extern int ULInitial ();
  /*     char *s1, *s2;     */
```

Abstract:

This function returns true if s2 is an initial string of s1. The comparison is case-insensitive. A null string is an initial substring of any string.

Parameters:

s1	String to be tested
s2	String which is the initial substring to test for

Returns:

Returns the integer 1 if s2 is an initial substring of s1; otherwise returns the integer 0.

6.48.1.42. Function ULPosString

Code:

```
extern int ULPosString ();
  /*     char *source, *mask;     */
```

Abstract:

This procedure is used to find the position of a pattern in a given string without case sensitivity.

Parameters:

Source	String that is to be searched
Mask	Pattern that we are looking for

Result:

If mask occurred in source then the index into Source of the first character of mask will be returned. If mask was not found then return 0. The source string is not modified.

Calls: PosString.

6.48.1.43. Function UpChar

Code:

```
extern char UpChar ();
  /*     char chr;     */
```

Abstract:

Converts C to upper case.

Parameters:

Chr Character to be converted

6.48.1.44. Function UpEQU

Code:

```
extern int UpEQU ();
  /*     char *s1, *s2;     */
```

Abstract:

Compares two strings for case-independent equality.

Parameters:

s1 and s2 Strings to be compared

Returns:

Returns 0 if the two strings are not equal.

Returns 1 if the two strings are equal.

6.48.2. Spice_StringDefs

Code:

```
#define CR '\r'
#define LF '\n'
#define Tab '\Tab'

#define MaxPStringSize 255      /* Length of Strings */
#define Inf     -32742          /* Magic value decoded as
                                   length-of-string */

typedef struct breaktype{
        char brkch;
        char uplow;
        char inex;
        char *breakstr;
        };

typedef struct breaktype  breakt, *btype;

typedef char PString[MaxPStringSize];
```

6.49. StdIO

File StdIO.h contains standard IO definitions. See the document "The IO System" in the *Accent Programming Manual.*

Include Files

#include <C_Types.h> #include <CioDefs.h>

6.49.1. Function open

Code:

```
extern int open();
  /*  char *name;
      int mode;
  */
```

Abstract:

Opens a file for reading or writing.

Parameters:

name	Pointer to the string of chars which is the filename
mode	Integer to represent the mode. This is defined in Ciodefs.h as ReadOpen=0, WriteOpen=1, and ReadWriteOpen=2.

Returns:

-1 if not successful. A file descriptor if successful. (A file descriptor is a small positive integer which is used to identify a particular file instead of using a filename.)

6.49.2. Function _open

Code:

```
extern int _open();
  /*  char *name;
      int mode;
      int ind;
  */
```

Abstract:

Attempts to open a file on the given index.

Parameters:

name — Pointer to the string of chars which is the filename

mode — Integer to represent the mode. This is defined in Ciodefs.h as ReadOpen=0, WriteOpen=1, and ReadWriteOpen=2.

ind — Index or file descriptor. (A file descriptor is a small positive integer which is used to identify a particular file instead of using a filename.)

Returns:

-1 if not successful. The index or file descriptor if successful.

6.49.3. Function read

Code:

```
extern int read();
  /*  int find;
      register char *buffer;
      register int nbytes;
  */
```

Abstract:

Reads bytes from a file.

Parameters:

find File descriptor or index that identifies your file

buffer Buffer where the data is to come from

nbytes Number of bytes to be transferred

Returns:

-1 on error and 0 on eof.

6.49.4. Function write

Code:

```
extern int write();
  /*  int find;
      register char *buffer;
      int nbytes;
  */
```

Abstract:

Writes bytes to a file.

Parameters:

find File descriptor or index that identifies your file

buffer Bytes of data will be written to this buffer

nbytes Number of bytes to be written out to the file

Returns:

-1 on error and nbytes if successful.

6.49.5. Function close

Code:

```
extern int close();
  /*  int find;  */
```

Abstract:

Closes a file.

Parameters:

find Index or file descriptor for the file to be closed

Returns:

-1 if unsuccessful and 0 if successful

6.49.6. Function Unlink

Code:

```
extern int unlink();
  /*  char *name;  */
```

Abstract:

Deletes a file.

Parameters:

name Pointer to the string of chars which is the filename

Returns:

-1 on error and 0 normally.

6.49.7. Function lseek

Code:

```
extern int lseek();
  /*  int find;
      long offset;
      int whence;
  */
```

Abstract:

Allows the user to move around in a file. This moves the current position in the file to 'offset positions from whence'.

Parameters:

find	File descriptor
offset	New position desired
whence	Origin from which the offset will be taken:
	0 = the beginning of the file
	1 = the current position
	2 = the end of the file

Returns:

-1 if unsuccessful. 0 if the file descriptor is type _NullDevice or _TypeScript. The new position if successful.

6.49.8. Function tell

Code:

```
extern long tell();
  /* int find; */
```

Abstract:

Returns the current position in the file relative to the begining.

Parameters:

find File descriptor

Returns:

Returns -1 on error or the current position if successful.

6.49.9. Function creat

Code:

```
extern int creat();
  /*  char *name;
      int protection;
  */
```

Abstract:

Creates a file. Protection is ignored.

Parameters:

name Name of the file to be created.

protection Currently an int not used in the code

Returns:

-1 if not successful. The file descriptor if successful.

6.49.10. Function setbuf

Code:

```
extern int setbuf();
  /*  FILE *fp;
      char *buf;
  */
```

Abstract:

This is not for user programs.

6.49.11. Function _cleanup

Code:

```
extern void _cleanup();
  /*  no parameters  */
```

Abstract:

Closes all files open for writing.

Parameters:

None

Returns:

There is no return value.

6.49.12. Function ungetc

Code:

```
extern int ungetc();
  /*  char ch;
      FILE *fp;
  */
```

Abstract:

Provides one character of push back for a text stream.

Parameters:

ch Character to be pushed back onto the file

fp File pointer

Returns:

-1 if unsuccessful or 0 if successful

6.49.13. Function printf

Code:

```
extern int printf();
  /*  char *format;
      int arguments;
      int a1,a2,a3,a4,a5,a6,a7,a8,a9;
      int b1,b2,b3,b4,b5,b6,b7,b8,b9;
  */
```

Abstract:

Produces formatted output - where the output is stdout.

Parameters:

format Format control string

arguments Number of arguments depends on the number of conversion specifications in the format string.

Returns:

Indeterminate

6.49.14. Function fprintf

Code:

```
extern int fprintf();
  /*  FILE *fp;
      char *format;
      int arguments;
      int a1,a2,a3,a4,a5,a6,a7,a8,a9;
      int b1,b2,b3,b4,b5,b6,b7,b8,b9;
  */
```

Abstract:

Produces output formatting with the output sent to the stream specified as the first argument.

Parameters:

fp File pointer

format Control string

arguments As required by the control string

Returns:

Indeterminate

6.49.15. Function sprintf

Code:

```
extern int sprintf();
  /*  char *resultstr;
      char *format;
      int arguments;
      int a1,a2,a3,a4,a5,a6,a7,a8,a9;
      int b1,b2,b3,b4,b5,b6,b7,b8,b9;
  */
```

Abstract:

Produces formatted output to a string.

Parameters:

resultstr String to contain the output

format Control string

arguments Argments as needed from the control string

Results:

Indeterminate

6.49.16. Function scanf

Code:

```
extern int scanf();
  /*  char *format;
      int arguments;
      int a1,a2,a3,a4,a5,a6,a7,a8,a9;
      int b1,b2,b3,b4,b5,b6,b7,b8,b9;
  */
```

Abstract:

Reads formatted input text.

Parameters:

format Control string

arguments As specified by the control string

Returns:

The number of successfully matched and assigned input items. If EOF is

reached, EOF (-1) is returned.

6.49.17. Function fscanf

Code:

```
extern int fscanf();
  /*  FILE *fp;
      char *format;
      int arguments;
      int a1,a2,a3,a4,a5,a6,a7,a8,a9;
      int b1,b2,b3,b4,b5,b6,b7,b8,b9;
  */
```

Abstract:

Reads formatted input text from fp.

Parameters:

fp File pointer

format Control string

arguments Arguments as needed in the control string.

Returns:

The number of successfully matched and assigned input items. If EOF is reached, EOF (-1) is returned.

6.49.18. Function sscanf

Code:

```
extern int sscanf();
/*  char *str;
    char *format;
    int arguments;
    int a1,a2,a3,a4,a5,a6,a7,a8,a9;
```

```
    int b1,b2,b3,b4,b5,b6,b7,b8,b9;
*/
```

Abstract:

Reads formatted input text from str.

Parameters:

str — String from which to read text

format — Control string

arguments — As specified by the control string

Returns:

The number of successfully matched and assigned input items. If EOF is reached, EOF (-1) is returned.

6.49.19. Function _puts

Code:

```
extern int _puts();
  /*  register char *str;
      register FILE *fptr;
      int addnl;
  */
```

Abstract:

Writes a string to a file. This routine handles calls to puts and fputs (see the macros in stdio.h)

Parameters:

str — String to be written to the file

fptr — File pointer

addnl 1 or 0. If 1, an extra newline character is written to file.

Returns:

Indeterminate

6.49.20. Function gets

Code:

```
extern char *gets();
  /*  register char *str;  */
```

Abstract:

Gets a string from stdin. Do not put the \n in the output string.

Parameters:

str Array where the input string is to be stored

Returns:

str if successful. NULL if unsuccessful.

6.49.21. Function fgets

Code:

```
extern char *fgets();
  /*  register char *str;
      register int max;
      register FILE *fptr;
  */
```

Abstract:

Reads a string from the given file. The returned string will be at most max chars long and include the \n if found.

Parameters:

str Array to contain the new string

max Maximum number of chars to be read

fptr File pointer

Returns:

The pointer to the new string if successful. NULL is returned if eof occurs.

6.49.22. Function fgetc

Code:

```
extern char fgetc();
  /*  register FILE *fptr;  */
```

Abstract:

Reads a character from any file.

Parameters:

fptr File pointer

Returns:

The character read

6.49.23. Function Fillbuf

Code:

```
extern char fillbuf();
  /*  register FILE *fptr;  */
```

Abstract:

This function is not for user programs.

6.49.24. Function fputc

Code:

```
extern int fputc();
  /*  char ch;
      register FILE *fptr;
  */
```

Abstract:

Writes a character to a file.

Parameters:

ch Character to be written to the file

fptr File pointer

Returns:

Indeterminate

6.49.25. Function putbuf

Code:

```
extern int putbuf();
  /*  char ch;
      register FILE *fptr;
  */
```

Abstract:

This function is not for user programs.

6.49.26. Function fopen

Code:

```
extern FILE *fopen();
  /*  char *name;
      char *mode;
  */
```

Abstract:

Opens a file for read, write, or append.

Parameters:

name — Filename to be opened

mode — "r" for read, "w" for write, "a" for append

Returns:

NULL if unsuccessful. Otherwise the file pointer for the opened file.

6.49.27. Function freopen

Code:

```
extern FILE *freopen();
  /*  char *name;
      char *mode;
      FILE *fp;
  */
```

Abstract:

Recycles the stream (ie. fclose followed by an fopen).

Parameters:

name — Filename to be associated with the stream

mode "r" for read, "w" for write, "a" for append

fp the stream or file pointer

Returns:

NULL if unsuccessful or the file pointer if successful.

6.49.28. Function fread

Code:

```
extern int fread();
  /*  char *ptr;
      int size;
      int nitems;
      FILE *fp;
  */
```

Abstract:

Reads a block of data.

Parameters:

ptr Points to the buffer where the bytes of data are to be stored

size Size of the items to be read

nitems Number of items to be read

fp File pointer to the opened stream

Returns:

0 if unsuccessful or the number of items read if successful

6.49.29. Function fwrite

Code:

```
extern int fwrite();
  /*  char *ptr;
      int size;
      int nitems;
      FILE *fp;
  */
```

Abstract:

Writes a block of data.

Parameters:

ptr	Points to the first item in the buffer to be written
size	Size of the items
nitems	Number of items to be written
fp	File pointer or stream

Returns:

0 if unsuccessful or the number of items written if successful

6.49.30. Function fclose

Code:

```
extern int fclose();
  /*  FILE *fp;  */
```

Abstract:

Closes a file.

Parameters:

fp File pointer

Returns:

-1 if unsuccessful and 0 if successful.

6.49.31. Function fflush

Code:

```
extern int fflush();
  /*  FILE *fp  */
```

Abstract:

Empties the buffer to the destination device.

Parameters:

fp Stream or file pointer

Returns:

-1 if unsuccessful or 0 if successful.

6.49.32. Function malloc

Code:

```
extern char *malloc();
  /*  unsigned size;  */
```

Abstract:

Allocates memory.

Parameters:

size Number of bytes to be allocated

Returns:

A char pointer to the allocated memory

6.50. StrHack

Abstract:

File StrHack.h contains macros to convert a Pascal string to a C string and vice versa.

Include Files

#include <C_Types.h>

6.50.1. Macro Pascal_To_C

Code:

```
#define Pascal_To_C(cstr,passtr) \
 { \
 int _i; \
 for(_i=1; _i<=(passtr)._StrSize; _i++) \
  (cstr)[_i-1] = (passtr)._StringChars[_i]; \
 (cstr)[_i-1] = '\0'; \
 }
```

Abstract:

Copies Pascal-type string (see DefineString in C_Types.h) into c string.

Parameters:

cstr — Something of type "char *" for which enough space has been alocated to hold all of passtr

passtr — Something of a "DefineString" type (a pascal-type string)

6.50.2. Macro C_To_Pascal

Code:

```
#define C_To_Pascal(passtr,cstr) \
 { \
 int _i; \
 for(_i=0;(cstr)[_i]!='\0';_i++) \
  (passtr)._StringChars[_i+1] = (cstr)[_i]; \
 (passtr)._StrSize = _i; \
 }
```

Abstract:

Copies c string into a Pascal-type string (see DefineString in C_Types.h).

Parameters:

passtr — Something of a "DefineString" type (a pascal-type string)

cstr — Something of type "char *" for which enough space has been alocated to hold all of passtr

6.51. String.h

6.51.1. Function strcat

Code:

```
extern char * strcat();
  /*     char *s1, s2;     */
```

Abstract:

Concatenates s2 at the end of s1. s1 must be large enough to hold the concatenated string.

Result:

Returns s1.

6.51.2. Function strcmp

Code:

```
extern int strcmp();
  /*     char *s1, *s2;     */
```

Abstract:

Compares strings s1 and s2.

Result:

Returns an integer:

s1>s2 >0 s1 == s2 0 s1<s2 <0

6.51.3. Function strlen

Code:

```
extern int strlen();
  /*     char *s1;     */
```

Abstract:

Returns the number of non-NULL bytes of s1.

6.51.4. Function strcpy

Code:

```
extern char * strcpy();
  /*     char *s1, *s2;     */
```

Abstract:

This function copies string s2 to s1. S1 must be large enough to hold s2.

Returns:

Returns s1.

6.51.5. Function strncat

Code:

```
extern char * strncat();
  /*     char *s1, s2;     */
```

Abstract:

Concatenates at most n bytes of s2 on the end of s1. Be sure that s1 is big enough.

Returns:

Returns s1 with concatenated characters.

6.51.6. Function strncmp

Code:

```
extern int strncmp();
  /*     char *s1, *s2;     */
```

Abstract:

Compare a maximum of n bytes of s1 and s2.

Result:

Return an integer according to the following:

s1>s2 : >0

s1==s2 : 0

s1<s2 : <0

6.51.7. Function strncpy

Code:

```
extern char * strncpy();
  /*     char *s1, *s2;
         int n;
   */
```

Abstract:

Copies string s2 to s1, truncating or null-padding to copy n bytes.

Result:

Returns a pointer to s1.

6.51.8. Function index

Code:

```
extern char * index();
  /*    char *s1;
        char c;
   */
```

Abstract:

Returns the pointer in s1 at which the character c appears. (Or NULL if not found).

6.51.9. Function rindex

Code:

```
extern char * rindex();
  /*    char *s1;
        char c;
   */
```

Abstract:

Returns the pointer in s1 at which the character c last appears (or NULL if not found).

6.52. Symdefs

File Symdefs.h is the definitions file for the Symbol table file produced by the Pascal compiler. This file is used by the debugger and the compiler so it cannot include anything from either.

Design:

The Sym file is used to get the offset and types for all variables defined in each module. This information is organized by procedure. The Sym file format is very simple and does not have references to other Sym files.

The format is:

Block 0 is of type VarDictEntry. Blocks 1 to EOF have the variable information in them.

In Block 0, the CompID is used to check that this file is associated with the Seg file. The globals word points to the entry that describes the globals of the module or program. If this is a program, the number in the globals word will be the same number as in the entry for routine zero in the routine section.

The numbers in the "routine" section (and in the globals word) are UNSIGNED word offsets from the start of block 1 of the word that begins the entry for that particular routine.

For each routine, there are two sets of information. First is a packed array containing all the identifiers. Second is two words for each identifier describing its offset and type.

The character section starts with an integer containing the byte count of the total number of bytes used in the name section. It is called TotNumChars. This number can be used to find the word start of the type information for this procedure by the formula:

procStart + (TotNumChars+1) div 2.

The format for the variable names is:

length of this name: 1 byte;

name: "length" bytes

Thus all the names of variables defined in the routine will be packed together at the top of the description for that routine. The name will be in uppercase and less than or equal to 14 characters. Thus, adding length+1 to the index for the start of one name will get to the start of the next name.

The information for each variable is two words. The first is the offset from the start of this procedure's local area on the stack of the start of this variable. If the variable is in the global area, then the offset will be from the start of the global area. The second word is a VarDescriptor and contains some information about the type of variable.

Thus to find the entry for variable FOO in routine X:

```
Read Block 0 into p.
L.low := p.dict^.routines[X];  (* so can use all 16 bits *)
L.high := 0;

Read Block (shrink(L.lng div 256) + 1) into p.
```

wordOffset := shrink(L.lng mod 256);
TotNumChars := p.int↑[wordOffset];

byteOffset := 2*WordOffset + 2;
(* 2 bytes for TotNumChars *)

p.chars↑[byteOffset] is the length of the first identifier.
p.chars↑[byteOffset+1] is the first character of the first identifier.

Iterate through the identifiers looking for FOO, keeping count of the number of identifiers searched in cnt (starting from 0).

wordOffset := (byteOffset + TotNumChars+1) div 2;
wordOffset := wordOffset + 2*cnt;

p.int↑[wordOffset] is now the offset of the variable on the stack.
p.int↑[wordOffset+1] is now the VarDescriptor for the variable.

NOTE that in the above, if the index into p.p got to be bigger than 255 (or 511 for character indexing), the next block would be read into p and the index would be decremented by the appropriate amount.

Code:

```
/*
#if !UnderAccent
#include <PFileSys.h>
```

```
#endif
*/
#include <TimeDefs.h>


#define T_Unknown 0
#define T_Char 1
#define T_Boolean 2
#define T_Integer 3
#define T_Enumerated 4
#define T_Real 5
#define T_Long 6
#define T_Pointer 7
#define T_Var 8
#define T_Routine 9
#define T_File 10
#define T_String 11
#define T_Set 12
#define T_Record 13
#define T_Array 14
#define T_Misc 15

typedef short BaseType;

/*Note: Subranges use the base type.  T_Var is for
  Var parameters to routines.  T_Routine is used for
  Routine parameters to routines.  T_Unused is a
  place holder and T_Unknown is used when the compiler
  is unable to provide the information for some reason
  or if there is no information to supply.*/

/* Note: T_Misc in conjunction with the Etc field specifies
   symbols which are either in the registers or are compiler
   generated temporaries.    */

/* special values for Etc when mainType is T_Misc  */

#define MiscEtc_CompilerTemp 0
         /* compiler generated temporary */
```

```
#define MiscEtc_Register_16Bit 1
        /* 16-bit, integer, which resides in a register */

#define MiscEtc_Register_32Bit 2
        /* 32-bit, long, which resides in a register */


#define Max_SymRoutines 251

typedef struct
 { /*this is block zero of the file*/
    Internal_Time CompID;
    short Globals; /*the globals of the module or program*/
    short Routines[Max_SymRoutines+1];
 } VarDictEntry;

typedef VarDictEntry *pVarDictEntry;

typedef struct
 {
    short TotNumChars;
    char *VarNames;
 } VarNameArray;

/*The VarNameArray type is actually never used to access
  the variable character information.  It is included
  here for documentation purposes only*/

typedef struct
 {
    unsigned int mainType:8; /*the type of the variable*/

    unsigned int subType:8; /*if Array, Pointer, File
                              or Set, this field specifies
                              the type of thing aggregated
                              over*/

    unsigned int etc:8; /*if mainType = record or set
```

```
                                then is number of words
                                necessary to store mainType,
                                else if mainType = string,
                                then is number of characters
                                in the string, else if mainType
                                = array then is size of SubType
                                (the component of the array)
                                else 1.  Except:  If mainType
                                is a packed array then 0.  If
                                the etc field would be filled
                                with a number > 255 then 0 */
 } VarDescriptor;

typedef char Var_CharAr[512];
typedef Var_CharAr *pVar_CharAr;

typedef short Var_IntAr[256];
typedef Var_IntAr *pVar_IntAr;

typedef union
    {
/*#if UnderAccent*/
        caddr_t P;
/*#else
        pDirBlk p;
#endif*/
        pVarDictEntry dict;
        pVar_CharAr chars;
        pVar_IntAr Int;
    } var_ptr;
```

6.53. Time Files

Files Time.h and TimeDefs.h contain the routines and definitions for the time server. File TimeBits.h contains definitions for bit fields in TimeDefs.h and Oldtime.h and a conversion hack using union.

Also see OldTime.

The routines are explained in the document "The Time Server" in the *Accent Programming Manual.*

6.53.1. Time

Code:

```
#include <AccentType.h>
#include <TimeDefs.h>

 extern void InitTime ();
  /*     Port RPort;     */

 extern Internal_Time GetDateTime ( );
  /*     Port ServPort;     */

 extern void SetDateTime ( );
  /*     Port ServPort;
         Internal_Time ITime;
   */

 extern void SetSystemZone ( );
  /*     Port ServPort;
         Integer TimeZone;
         Boolean DSTWhenTimely;
   */

 extern User_Time T_IntToUser ( );
  /*     Port ServPort;
```

```
        Internal_Time ITime;
  */

extern Internal_Time T_UserToInt ( );
 /*     Port ServPort;
        User_Time UTime;
  */

extern String T_UserToString ( );
 /*     Port ServPort;
        User_Time UTime;
  */

extern User_Time T_StringToUser ( );
 /*     Port ServPort;
        String_255 STime;
        short  * Index;
        short  * WhatIFound;
  */

extern String T_IntToString ( );
 /*     Port ServPort;
        Internal_Time ITime;
        short TimeFormat;
  */

extern Internal_Time T_StringToInt ( );
 /*     Port ServPort;
        String_255 STime;
        short  * Index;
        short  * WhatIFound;
  */


extern User_Time T_IntToZone ( );
 /*     Port ServPort;
        Internal_Time ITime;
        Zone_Info TZone;
  */
```

```
extern Internal_Time T_Never ( );
 /*     Port ServPort;     */

extern User_Time GetUserTime ( );
 /*     Port ServPort;     */

extern String GetStringTime ( );
 /*     Port ServPort;
        short TimeFormat;
  */
```

6.53.2. TimeBits

File TimeBits.h contains definitions for bit fields in TimeDefs.h and Oldtime.h and a conversion hack using union.

Code:

```
/* Fields in Date_Fields (Year : int, Month : 4,
                          Day : 5, WeekDay : 3) */

#define DtYrLoc     1
#define DtYrCnt    16

#define DtMonLoc   17
#define DtMonCnt    4

#define DtDayLoc   21
#define DtDayCnt    5

#define DtWkDyLoc  26
#define DtWkDyCnt   3


/* Fields in Time_Fields (Hour : 5, Minute : 6,
                          Second : 6, Millisecond : 10) */

#define TmHrLoc     1
```

```
#define TmHrCnt     5

#define TmMinLoc    6
#define TmMinCnt    6

#define TmSecLoc   12
#define TmSecCnt    6

#define TmMSecLoc 18
#define TmMSecCnt 10

/* Fields in Zone_Info (TimeZone : int, UseTimeZone : 1,
                        Daylight : 1, UseDaylight : 1) */

#define ZnZnLoc      1
#define ZnZnCnt     16

#define ZnUseZnLoc 17
#define ZnUseZnCnt  1

#define ZnDLLoc      1
#define ZnDLCnt     16

#define ZnUseDLLoc 17
#define ZnUseDLCnt  1

/* Fields in TimeStamp (Hour : 5, Day : 5, Second : 6,
                        Minute : 6, Month : 4, Year : 6) */

#define TSHrLoc    1
#define TSHrCnt    5

#define TSDayLoc   6
#define TSDayCnt   5

#define TSSecLoc 11
#define TSSecCnt  5

#define TSMinLoc 16
```

```
#define TSMinCnt  6

#define TSMonLoc 22
#define TSMonCnt  4

#define TSYrLoc  26
#define TSYrCnt   6

typedef union
        {
            long 1;
            Date_Fields df;
            Time_Fields tf;
            Zone_Info   zi;
            TimeStamp   ts;
        }
         TimeKluge;
```

6.53.3. TimeDefs

File TimeDefs.h contains the types used by the Time module.

See the document "The Time Server" in the *Accent Programming Manual.*

Code:

```
/*
 * Internal_Time:  the date and time in Greenwich Mean Time.
 *       This is the Accent time standard.  To optimize space
 *       usage, we store time as an ordered pair, the first
 *       representing the number of weeks which have passed
 *       since 17-Nov-1858, when the Smithsonian time standard
 *       began. The second represents the number of
 *       milliseconds which have passed in that week.
 */

typedef          struct
```

```
                        {
                        short                   Weeks;
                        Long                    MSecInWeek;
                        }
                                Internal_Time;

#define Eq_Internal_Time(a,b)  ((a).Weeks == (b).Weeks && \
  (a).MSecInWeek == (b).MSecInWeek)

/*
 * Date_Fields:  fields necessary for representing date information
 *       without respect to the time.  Used in by User_Time.
 */

typedef         struct
                        {
                        int             Year;
                        unsigned
                        int             Month : 4;
                        unsigned
                        int             Day : 5;
                        unsigned
                        int             Weekday : 3;
                        }
                                Date_Fields;

/*
 * Time_Fields:  fields necessary for representing time information
 *       without respect to the date.  Used in User_Time.
 */

typedef         struct
                        {
                        unsigned
                        int             Hour : 5;
                        unsigned
                        int             Minute : 6;
                        unsigned
                        int             Second : 6;
```

```
                        unsigned
                        int             Millisecond : 10;
                        }
                                Time_Fields;

/*
 * Zone_Info:  this record allows the user to specify and
 *       receive time zone information.  Both the zone number
 *       and application of daylight savings time may be either
 *       specified explicitly, or defaulted, depending upon the
 *       settings of the UseTimeZone and UseDaylight bits.
 *       Used in User_Time.
 */

typedef         struct
                        {
                        int             TimeZone;
                           /* Increasing minutes west from GMT.
                              GMT = 0, EST = 5*60,
                              CST = 6*60, ...
                              Used only if UseTimeZone is
                              set. */
                        unsigned
                        int             UseTimeZone : 1;
                          /* True when TimeZone field is valid,
                             else false when local time zone is
                             to be used. */
                             unsigned
                        int             Daylight : 1;
                          /* True if daylight savings time is to
                             be applied.  Used only if
                             UseDaylight is set. */
                        unsigned
                        int             UseDaylight : 1;
                          /* True if Daylight savings field is
                             valid, else false when the system
                             default for daylight  savings time
                             application is to be used. */
                        }
```

```
                                        Zone_Info;

/*
 * User_Time:  Date and Time information broken down into fields
 *       as the user would want to use it for input and output.
 *       Both the time zone index and application of daylight
 *       savings time can be either explicitly specified, or
 *       defaulted through use of the ZoneInfo fields.  The
 *       Weekday field is unused for input.
 */


typedef         struct
                        {
                        Date_Fields     Date;
                        Time_Fields     Time;
                        Zone_Info       Zone;
                        }
                                User_Time;


/*
 * The following flag values may be ORed together to form
 * TimeFormat values.
 */

#define         TF_Weekday      0000001
                               /*
                                * If set output the day of
                                * the week according to the
                                * setting of TF_FullWeekday
                                * else don't output the
                                * day of the week
                                */

#define         TF_FullWeekday  0000002
                               /*
                                * If set output full text
                                * for the weekday else the
```

```
                                * 3-letter abbreviation
                                * (Monday/Mon)
                                */

#define         TF_NoDate       0000004
                               /* If set do not output date
                                * and ignore flags through
                                * TF_NoTime
                                */

#define         TF_FullMonth    0000010
                               /* If set output full text
                                * for the month when the month
                                * is alphabetic else the
                                * 3-letter abbreviation
                                * (March/Mar)
                                */

#define         TF_FullYear     0000020
                               /*
                                * If set output the year as
                                * a 4-digit number else the year
                                * is is output as a 2-digit
                                * number if in the range
                                * 1900-1999 (1982/82)
                                */

/*
 * The next six settings are mutually exclusive
 */

#define         TF_Dashes       0000000
                               /*
                                * Output date as day-month-year
                                * (22-Mar-60)
                                */
#define         TF_Spaces       0000040
                               /*
                                * Output date as day month year
```

```
                                * (22 Mar 60)
                                */
#define         TF_Reversed     0000100
                               /*
                                * Output date as month day, year
                                * (Mar 22, 60)
                                */
#define         TF_Slashes      0000140
                               /*
                                * Output date as month/day/year
                                * (03/22/60)
                                */
              /* #000200 is reserved for future expansion*/
              /* #000240 is reserved for future expansion*/
#define         TF_ANSI         0000300
                               /*
                                * Output date according to ANSI
                                * X3.30-1971.  Also slightly
                                * affects time formatting.
                                * (600322)
                                */
#define         TF_ANSI_Ordinal 0000340
                               /*
                                * Similar to TF_ANSI but 3-digit
                                * day-of-year instead of
                                * month and day.
                                * (60082)
                                */
#define         TF_DateFormat   0000340
                               /*
                                * A mask allowing us to examine
                                * the above.
                                */

#define         TF_NoTime       0000400
                               /*
                                * If set do not output time and
                                * ignore flags through
                                * TF_NoColumns
```

```
                                */

/*
 * The next two settings are mutually exclusive
 */

#define         TF_NoSeconds     0001000
                                /*
                                 * If set do not output the
                                 * seconds
                                 */
#define         TF_Milliseconds  0002000
                                /*
                                 * If set output milliseconds
                                 * as hh:mm:ss.sss else
                                 * omit them
                                 * (17:00:00.001/17:00:00)
                                 */

#define         TF_12_Hour       0004000
                                /*
                                 * If set output the time in
                                 * 12-hour format with 'am' or
                                 * 'pm' following the time
                                 * else output in 24-hour
                                 * format.  Note that exact
                                 * NOON outputs neither 'am'
                                 * nor 'pm' because 12:00am
                                 * is 0000 and 12:00pm is
                                 * 2400.  Use of TF_12_Hour
                                 * with TF_ANSI or
                                 * TF_ANSI_Ordinal is
                                 * supported but not
                                 * recommended for a number
                                 * of reasons.  If used
                                 * with either ANSI format,
                                 * however, the codes 'A', 'P',
                                 * and 'N' are used for am, pm,
                                 * and noon, respectively.
```

```
                                * (5:00:00pm/17:00:00)
                                */
#define         TF_TimeZone     0010000
                               /*
                                * If set output the time zone
                                * as -zzz after the time else
                                * omit it
                                * (17:00:00-EDT/17:00:00)
                                */

#define         TF_NoColumns    0040000
                               /*
                                * If set output numeric
                                * date/time quantities in the
                                * smallest fields into which
                                * they will fit, else output
                                * them in fixed size fields.
                                * If not set, the date/time
                                * will be output in fixed
                                * length fields, thus making
                                * this format appropriate for
                                * columnar display.  Note that
                                * TF_FullMonth and
                                * TF_FullWeekday are currently
                                * NOT padded with blanks,
                                * even if TF_NoColumns
                                * is off.
                                */
#define         TF_BlankPad     0020000
                               /*
                                * If set, pad fixed-width
                                * numbers with blanks instead
                                * of zeroes WHERE REASONABLE.
                                * Ignored if TF_NoColumns is
                                * set.
                                */

#define         TF_Never        0100000
                               /*
```

```
                                * If set allow the
                                * distinguished time value
                                * NEVER to be output as the
                                * string 'Never' else signal
                                * an error
                                */

/*
 * The following flags are returned to indicate which fields
 * of the date and time were present upon parsing a date/time
 * string.
 */

#define         TP_Weekday      0000001 /* Weekday present */
#define         TP_Date         0000002 /* Date present */
#define         TP_Time         0000004 /* Time present */
#define         TP_Zone         0000010 /* TimeZone present */
#define         TP_Never        0000020 /* Time input was
                                           NEVER */
#define         TP_RESERVED     0177740 /* Reserved for
                                           expansion */

/*
 * String_255:  The maximum length string.  We parse dates from
 *         such strings. This definition should probably be
 *         somewhere else.
 */

typedef         String255       String_255;
```

6.54. Ts/Tsdefs

Files Ts.h and Tsdefs.h contain the routines and definitions for the typescript manager.

See the document "The Typescript Manager" in the *Accent Programming Manual.*

6.54.1. Ts

Code:

```
#include <AccCall.h>
#include <TSDefs.h>
#include <AccentType.h>

extern void InitTS ();
  /*    Port Rport;    */

extern TString255 Typescript_Version ( );
  /*    Typescript REMOTE_PORT;    */

extern Typescript STSOpen ( );
  /*    Port REMOTE_PORT;
        Viewport VP;
        Port ENV;
   */

extern Typescript STSOpenWindow ( );
  /*    Port REMOTE_PORT;
        Window W;
        Port ENV;
   */

extern Typescript STSFullOpen ( );
  /*    Port REMOTE_PORT;
        Viewport VP;
        Port ENV;
        TString255 FONTNAME;
```

```
        Boolean DOWRAP;
        short DISPPAGES;
   */
extern Typescript STSFullOpenWindow ( );
  /*    Port REMOTE_PORT;
        Window W;
        Port ENV;
        TString255 FONTNAME;
        Boolean DOWRAP;
        short DISPPAGES;
   */

extern Typescript STSOpenTerm ( );
  /*    Port REMOTE_PORT;
        Viewport VP;
        Port W;
        Port ENV;
        TString255 FONTNAME;
        short DISPPAGES;
        pKeyTab KEYTRANTAB;
        long TABLESIZE;
   */

extern Boolean STSFullLine ( );
  /*    Port REMOTE_PORT;    */

extern Character STSGetChar ( );
  /*    Port REMOTE_PORT;    */

extern TString255 STSGetString ( );
  /*    Port REMOTE_PORT;    */

extern void STSPutChar ( );
  /*    Typescript REMOTE_PORT;
        Character CH;
   */

extern void STSPutString ( );
  /*    Typescript REMOTE_PORT;
```

```
        TString255 S;
   */

extern void STSFlushInput ( );
  /*    Typescript REMOTE_PORT;     */


extern void STSFlushOutput ( );
  /*    Typescript REMOTE_PORT;     */

extern void STSChangeEnv ( );
  /*    Typescript REMOTE_PORT;
        Port ENV;
   */

extern Window STSGrabWindow ( );
  /*    Typescript REMOTE_PORT;
        Port KPORT;
   */

extern void STSPutCharArray ( );
  /*    Typescript REMOTE_PORT;
        pTSCharArray CHARS;
        long CHAR_COUNT;
        short FIRSTCH;
        short LASTCH;
   */

extern GeneralReturn STSMoveCursor ( );
  /*    Typescript REMOTE_PORT;
        short *CHARPOS;
        short *LINENUM;
   */

extern GeneralReturn STSCursorOp ( );
  /*    Typescript REMOTE_PORT;
        CursorOp OP;
        short COUNT;
        short *CHARPOS;
```

```
        short *LINENUM;
   */

extern GeneralReturn STSDeleteOp ( );
  /*    Typescript REMOTE_PORT;
        DeleteOp OP;
        short COUNT;
        short *CHARPOS;
        short *LINENUM;
   */

extern GeneralReturn STSScreenOp ( );
  /*    Typescript REMOTE_PORT;
        ScreenOp OP;
        short TOPLINE;
        short BOTTOMLINE;
        short COUNT;
   */

extern GeneralReturn STSCursorPos ( );
  /*    Typescript REMOTE_PORT;
        short *CHARPOS;
        short *LINENUM;
        short *X;
        short *Y;
   */

extern void STSVpSize ( );
   /*   Typescript REMOTE_PORT;
        short *CHARWIDTH;
        short *LINES;
        short *PIXWIDTH;
        short *PIXHEIGHT;
   */

extern void STSInputStatus ( );
  /*    Typescript REMOTE_PORT;
        Boolean *EMPTY;
        Boolean *FULL;
```

```
    */

extern GeneralReturn STSSetInput ( );
  /*    Typescript REMOTE_PORT;
        TString255 INPUTSTR;
    */

extern GeneralReturn STSGiveKey ( );
  /*    Typescript REMOTE_PORT;
        EventRec UNTRANKEY;
        Boolean *FULLLINE;
    */

extern GeneralReturn STSChangeKeyTran ( );
  /*    Typescript REMOTE_PORT;
        pKeyTab KEYTRANTAB;
        long TABLESIZE;
    */

extern GeneralReturn STSMoreMode ( );
  /*    Typescript REMOTE_PORT;
        Boolean *ON;
    */
```

6.54.2. Tsdefs

Code:

```
#if 0
#include <BuiltInDefs.h>
#else
#include <C_Types.h>
#include <OldBuiltInDefs.h>
#endif
#include <Sapphdefs.h>
#include <keytrandefs.h>

typedef short CursorOp;
```

```
typedef short DeleteOp;

typedef short ScreenOp;

typedef Character TSCharArray[];

typedef TSCharArray *pTSCharArray;

typedef String255 TString255;

typedef Port Typescript;

#define ts_home 0
#define ts_boln 1
#define ts_eoln 2
#define ts_up 3
#define ts_down 4
#define ts_left 5
#define ts_right 6
#define ts_delchar 0
#define ts_ereoln 1
#define ts_erboln 2
#define ts_bell 0
#define ts_clear 1
#define ts_redisplay 2
#define ts_scroll 3
```

6.55. ViewKern

File Viewkern.h attempts to call the kernel protected graphics operations. If those operations fail, then it calls the window manager's graphics operations instead. See also the section "Sapphire Files" in this document.

The routines are described in the document "The Window Manager" in the *Accent Programming Manual.*

Code:

```
#include <AccentType.h>

#include <SapphDefs.h>

extern void VPROP();
  /*    Viewport        destvp;
        RopFunct        funct;
        short           dx;
        short           dy;
        short           width;
        short           height;
        Viewport        srcvp;
        short           sx;
        short           sy;
   */

extern void VPColorRect();
  /*    Viewport        vp;
        RectColorFunct  funct;
        short           x;
        short           y;
        short           width;
        short           height;
   */

extern void VPScroll();
```

```
  /*     Viewport         destvp;
         short            x;
         short            y;
         short            width;
         short            height;
         short            xamt;
         short            yamt;
    */

extern void VPLine();
  /*     Viewport         destvp;
         LineFunct        funct;
         short            x1;
         short            y1;
         short            x2;
         short            y2;
    */

extern void VPString();
  /*     Viewport         destvp;
         Viewport         fontvp;
         RopFunct         funct;
         short           *dx;
         short           *dy;
         VPStr255         str;
         short            firstch;
         short           *lastch;
    */

extern void VPChArray();
  /*     Viewport         destvp;
         Viewport         fontvp;
         RopFunct         funct;
         short           *dx;
         short           *dy;
         pVPCharArray     chars;
         Long             chars_cnt;
         short            firstch;
         short           *lastch;
```

```
    */

extern void VPChar();
  /*    Viewport        destvp;
        Viewport        fontvp;
        RopFunct        funct;
        short           *dx;
        short           *dy;
        Char             ch;
    */

extern void VPPutString();
  /*    Viewport        destvp;
        Viewport        fontvp;
        RopFunct        funct;
        short           dx;
        short           dy;
        VPStr255        str;
        short           firstch;
        short           lastch;
    */

extern void VPPutChArray();
  /*    Viewport        destvp;
        Viewport        fontvp;
        RopFunct        funct;
        short           dx;
        short           dy;
        pVPCharArray    chars;
        Long             arsize;
        short           firstch;
        short           lastch;
    */

extern void VPPutChar();
  /*    Viewport        destvp;
        Viewport        fontvp;
        RopFunct        funct;
        short           dx;
```

```
        short           dy;
        Char            ch;
*/
```

6.56. ViewPt

File ViewPt.h contains the routines for viewports. Also see Sapphire Files.

The routines are described in the document "The Window Manager" in the *Accent Programming Manual.*

Code:

```
#include <AccentType.h>
#include <SapphDefs.h>
#include <Sapphfiledefs.h>
#include <ViewKern.h>


extern void InitViewPt ();
  /*    No parameters    */

extern Viewport MakeViewport ( );
  /*   -Viewport ServPort;
        short   x;
        short   y;
        short   w;
        short   h;
        short   rank;
        Boolean memory;
        Boolean courteous;
        Boolean transparent;
   */

extern void DestroyViewport ( );
  /*    Viewport ServPort;    */

extern void ViewportState ( );
  /*    Viewport ServPort;
        short    * curlx;
        short    * curty;
```

```
        short     * curwidth;
        short     * curheight;
        short     * curRank;
        Boolean   * memory;
        Boolean   * courteous;
        Boolean   * transparent;
           */

extern void ModifyVP ( );
  /*    Viewport ServPort;
        short    newlx;
        short    newty;
        short    newwidth;
        short    newheight;
        short    newrank;
        Boolean wantVpChEx;
   */

extern void EnableNotifyExceptions ( );
  /*    Viewport ServPort;
        Port notifyPort;
        Boolean changed;
        Boolean exposed;
   */

extern Integer GetVPRank ( );
  /*    Viewport ServPort;     */

extern void ReserveScreen ( );
  /*    Viewport ServPort;
        Boolean reserve;
   */       —

extern Viewport GetFullViewport ( );
  /*    Viewport ServPort;     */

extern void VPtoScreenCoords ( );
  /*    Viewport ServPort;
        short    x;
```

```
        short   y;
        short    * scrX;
        short    * scrY;
   */

extern void ScreenToVPCoords ( );
  /*    Viewport ServPort;
        short   scrX;
        short   scrY;
        short    * x;
        short    * y;
   */

extern void ViewROP ( );
  /*    Viewport ServPort;
        RopFunct funct;
        short   dx;
        short   dy;
        short   width;
        short   height;
        Viewport srcVP;
        short   sx;
        short   sy;
   */

extern void ViewColorRect ( );
  /*    Viewport ServPort;
        RectColorFunct funct;
        short   x;
        short   y;
        short   width;
        short   height;
   */

extern void ViewScroll ( );
  /*    Viewport ServPort;
        short   x;
        short   y;
        short   width;
```

```
        short   height;
        short   Xamt;
        short   Yamt;
   */

extern void ViewLine ( );
  /*    Viewport ServPort;
        LineFunct funct;
        short   x1;
        short   y1;
        short   x2;
        short   y2;
   */

extern void ViewString ( );
  /*    Viewport ServPort;
        Viewport fontVP;
        RopFunct funct;
        short    * dx;
        short    * dy;
        VPStr255 str;
        short   firstCh;
        short    * lastch;
   */

extern void ViewChArray ( );
  /*    Viewport ServPort;
        Viewport fontVP;
        RopFunct funct;
        short    * dx;
        short    * dy;
        pVPCharArray chars;
         Long chars_Cnt;
        short   firstCh;
        short    * lastch;
   */

extern void ViewChar ( );
  /*    Viewport ServPort;
```

```
        Viewport fontVP;
        RopFunct funct;
        short    * dx;
        short    * dy;
        char ch;
   */

extern void ViewPutString ( );
  /*    Viewport ServPort;
        Viewport fontVP;
        RopFunct funct;
        short    dx;
        short    dy;
        VPStr255 str;
        short    firstCh;
        short    lastch;
   */

extern void ViewPutChArray ( );
  /*    Viewport ServPort;
        Viewport fontVP;
        RopFunct funct;
        short    dx;
        short    dy;
        pVPCharArray chars;
         Long chars_Cnt;
        short    firstCh;
        short    lastch;
   */

extern void ViewPutChar ( );
  /*    Viewport ServPort;
        Viewport fontVP;
        RopFunct funct;
        short    dx;
        short    dy;
        char ch;
   */
```

```
extern void FontSize ( );
  /*     String  * name;
         short     * PointSize;
         short     * Rotation;
         short     * FaceCode;
         short     * maxWidth;
         short     * maxHeight;
         short     * xOrigin;
         short     * yOrigin;
         Boolean   * fixedWidth;
         Boolean   * fixedHeight;
    */

extern void FontCharWidthVector ( );
  /*     Viewport ServPort;
         char ch;
         short     * dx;
         short     * dy;
  */

extern void FontStringWidthVector ( );
  /*     Viewport ServPort;
         VPStr255 str;
         short    firstCh;
         short    lastch;
         short     * dx;
         short     * dy;
    */

extern Viewport GetSysFont ( );
  /*     Viewport ServPort;     */

extern Viewport LoadFontData ( );
  /*     Viewport ServPort;
         pVPIntegerArray FontData;
          Long FontData_Cnt;
    */

extern void SetFontChar ( );
```

```
  /*    Viewport ServPort;
        pVPIntegerArray OneCharData;
         Long OneCharData_Cnt;
        short   WordsAcross;
        char ch;
        short   CharWidth;
   */

extern void PutViewportBit ( );
  /*    Viewport ServPort;
        short   x;
        short   y;
        Boolean value;
   */

extern Boolean GetViewportBit ( );
  /*    Viewport ServPort;
        short   x;
        short   y;
        Boolean  * value;
   */

extern void PutViewportRectangle ( );
  /*    Viewport ServPort;
        RopFunct Funct;
        short   x;
        short   y;
        short   width;
        short   height;
        pVPIntegerArray Data;
        Long Data_Cnt;
        short   WordsAcross;
        short   ux;
        short   uy;
   */

extern Boolean GetViewportRectangle ( );
  /*    Viewport ServPort;
        short   x;
```

```
        short   y;
        short   width;
        short   height;
        pVPIntegerArray  * Data;
        Long  * Data_Cnt;
        short    * WordsAcross;
        short   ux;
        short   uy;
   */

extern void PushRegion ( );
  /*    Viewport ServPort;
        short   regionNum;
        short   leftx;
        short   topy;
        short   width;
        short   height;
   */

extern void DeleteRegion ( );
  /*    Viewport ServPort;
        short   regionNum;
   */

extern void DestroyRegions ( );
  /*    Viewport ServPort;     */

extern void ModifyRegion ( );
  /*    Viewport ServPort;
        short   regionNum;
        short   leftx;
        short   topy;
        short   width;
        short   height;
   */

extern void SetRegionCursor ( );
  /*    Viewport ServPort;
        short   regionNum;
```

```
        CursorSet cursorImage;
        short   cursIndex;
        CursorFunction cursFunc;
        Boolean track;
   */

extern void GetRegionCursor ( );
  /*    Viewport ServPort;
        short   regionNum;
        CursorSet  * cursorImage;
        short    * cursIndex;
        CursorFunction  * cursFunc;
        Boolean  * track;
   */

extern void SetRegionParms ( );
  /*    Viewport ServPort;
        short   regionNum;
        Boolean absolute;
        short   speed;
        short   minx;
        short   maxx;
        short   miny;
        short   maxy;
        short   modx;
        short   posx;
        short   mody;
        short   posy;
   */

extern void GetRegionParms ( );
  /*    Viewport ServPort;
        short   regionNum;
        Boolean  * absolute;
        short    * speed;
        short    * minx;
        short    * maxx;
        short    * miny;
        short    * maxy;
```

```
        short     * modx;
        short     * posx;
        short     * mody;
        short     * posy;
   */

extern void DestroyVPCursors ( );
  /*     CursorSet ServPort;  */

extern void GetCursor ( );
  /*     CursorSet ServPort;
         short    cursIndex;
         Pattern * pCursorData;
         short     * xOffset;
         short     * yOffset;
   */

extern void GetCursorSet ( );
  /*     CursorSet ServPort;
         struct OffsetRec   * Offsets;
         pPatternMapArray   * pCursorData;
          Long   * pCursorData_Cnt;
   */

extern void SetCursor ( );
  /*     CursorSet ServPort;
         short    cursIndex;
         Pattern pCursorData;
         short    xOffset;
         short    yOffset;
   */

extern CursorSet CreateCursorSet ( );
  /*     Viewport ServPort;
         struct OffsetRec *Offsets;
         pPatternMapArray pCursorData;
         Long pCursorData_Cnt;
   */
```

```
extern void SetCursorPos ( );
  /*    Viewport ServPort;
        short    x;
        short    y;
   */

extern void SetScreenColor ( );
  /*    Viewport ServPort;
        Boolean invert;
   */

extern void VPNIY ( );
  /*    No parameters     */

extern void XVPNIY ();
  /*    No parameters     */

extern void ViewPtException();
  /*    No parameters     */
```

6.57. WindowUtils

File WindowUtils.h provides several useful functions for manipulating windows.

Include Files, Defines, and Typedef

Code:

```
#include <stdio.h>
#include <SapphDefs.h>

#define DotDotDot (char)(0214)
                /* '...' in one char */
#define titleThreshold 5
                /* num of chars displayable in title line */
                /*   before we even try to display a title */
                /*   is rather arbitrary */


typedef char PackAryChar[256];
```

6.57.1. Function ShowPathAndTitle

Code:

```
extern void ShowPathAndTitle ();
 /*     char *_s;     */
```

Abstract:

This procedure is called to display the current path and the given string in the title line.

Environment: Assumes that UserWindow is initialized within PascalInit.

Parameters:

s String to be displayed

Side Effects:

None

6.57.2. Function ShowWindowErrorFlag

Code:

```
extern void ShowWindowErrorFlag ();
 /*     No parameters     */
```

Abstract:

Turns on the error Icon flag in the window associated with this process.

6.57.3. Function RemoveWindowErrorFlag

Code:

```
extern void RemoveWindowErrorFlag ();
 /*     No parameters     */
```

Abstract:

Turns off the error Icon flag in the window asociated with this process.

6.57.4. Function ShowWindowRequestFlag

Code:

```
extern void ShowWindowRequestFlag ();
 /*     No parameters     */
```

Abstract:

Turns on the request Icon flag in the window associated with this process.

6.57.5. Function RemoveWindowRequestFlag

Code:

```
extern void RemoveWindowRequestFlag ();
 /*     No parameters     */
```

Abstract:

Turns off the request Icon flag in the window associated with this process.

6.57.6. Function ShowWindowAttentionFlag

Code:

```
extern void ShowWindowAttentionFlag ();
 /*     No parameters     */
```

Abstract:

Turns on the attention Icon flag in the window associated with this process.

6.57.7. Function RemoveWindowAttentionFlag

Code:

```
extern void RemoveWindowAttentionFlag ();
 /*     No parameters     */
```

Abstract:

Turns off the attention Icon flag in the window associated with this process.

6.57.8. Function ComputeProgress

Code:

```
/* extern void StreamProgress (); */

 extern void ComputeProgress ();
  /*     long Current, Max;     */
```

Abstract:

Shows progress in the title-line progress bar, as an amount of a total.

Environment: Assumes that UserWindow is initialized within PascalInit.

Parameters:

Current How far the operation has gotten

Max Total amount for the operation

Side Effects:

None

6.57.9. Function RandomProgress

Code:

```
 extern void RandomProgress ();
  /*     No parameters     */
```

Abstract:

Shows random progress (something is happening, but we're not sure how much) in the title-line progress bar.

Environment: Assumes that UserWindow is initialized within PascalInit.

Side Effects:

None

6.57.10. Function QuitProgress

Code:

```
extern void QuitProgress ();
 /*     No progress     */
```

Abstract:

Turns off the title-line progress bar.

Environment: Assumes that UserWindow is initialized within PascalInit.

Side Effects:

None

6.57.11. Function MultiLevelProgress

Code:

```
extern void MultiLevelProgress ();
 /*     short Level;
        long Current, Max;
  */

/* extern void MultiStreamProgress (); */
```

Abstract:

Shows progress in the selected progress bar, as an amount of a total.

Environment: Assumes that UserWindow is initialized within PascalInit.

Parameters:

Level Which progress bar to use

Current How far the operation has gotten

Max — Total amount for the operation

Side Effects:

None

6.57.12. Function QuitMultiProgress

Code:

```
extern void QuitMultiProgress ();
 /*     short Level;     */
```

Abstract:

Turns off the selected progress bar.

Environment: Assumes that UserWindow is initialized within PascalInit.

Parameters:

Level Which progress bar to use

Side Effects:

None

INDEX

—