Standard File Operations
External Reference Specification
Steven Glass
August 12, 1986

First Release — May 8, 1986

Hope and Fantasy (never made public)

Release 2 — July 23, 1986

Hope, Fantasy and a little reality thrown in to fool the marks. This was written after the first bits and pieces of code were running (still not public)

Release 3 — August 12, 1986

Reality begins to dominate. This is written as the code is about to be released to developers for the first time. (This one is public.)

Standard File Operations

The Standard File Operations Tool Set provides the standard user interface for specifying a file to be opened or saved. It allows the file to be on a disk in any drive and it allows the user to change disks in a drive.

Data Structures

Reply Record

good	boolean	True if open pressed; False for cancel.
file type	word	ProDOS file type.
aux file type	word	ProDOS aux file type.
filename	string[15]	Name of the file in prefix 0.
full path name	string[128]	The full pathname of selected file.

Housekeeping Calls

SFBootInit — Internal routine called at load time to initialize the Standard File.

No Stack Parameters

SFStartup — Call made by an application before it makes any other Standard File calls.

Stack Before Call

previous contents	
ID for Mem Mgr	**WORD**
Address of ZP to Use	**WORD**
	<-SP

Stack After Call

previous contents	
	<-SP

Standard File must be initialized before it is called. The initalization routine allows it to have and use a zero page and an ID for any memory that is needed during the standard file call. An application may choose to call initialize standard file only when it is needed freeing memory for other uses. A typical sequence of code may be

```
SFStartup (....);
SFGetFile (...);
SFShutdown (...);
```

SFShutdown — Call made by an application to shutdown after SFStartup is called.

No Stack Parameters

See SFStartup for more details.

SFVersion — Returns the version number of Standard File.

Stack Before Call

previous contents	
space for version	
	<-SP

Stack After Call

previous contents	
version number	
	<-SP

SFReset — Resets Standard File.

No Stack Parameters

This is called when the system reset (CONTROL-RESET is pressed).

SFActive Returns whether or not Standard File is active.

Stack Before Call

previous contents	
space for boolean	
	<-SP

Stack After Call

previous contents	
boolean result	
	<-SP

Returns true if SFStartup has been called and SFShutdown has not been called. Returns false if SFStartup has not been called at all or SFShutdown was called since the last SFStartup.

Useful Calls

SFGetFile This is the call that returns a path name for a file selected by the user.

```
Stack Before Call
    |  previous contents  |
    |       WhereX        | integer
    |       WhereY        | integer
    |     PromptPtr       | POINTER
    |   FilterProcPtr     | POINTER
    |    TypeListPtr      | POINTER
    |     ReplyPtr        | POINTER
    |                     |<-SP

Stack After Call
    |  previous contents  |
    |                     |<-SP
```

WhereX and WhereY describe the location on the screen (in screen coordinates) of the dialog box.

The PromptPtr points to a string to display at the top of the dialog box. This is a standard ProDOS string.

The FilterProcPtr points to a procedure which will decide whether or not a particular file will be displayed. Set this pointer to 0 to prevent this procedure from being called. The filter proc is called in full native mode as one would call a Pascal Function having one long parameter.

The calling sequence inside SFGetFile is as follows:

```
PushWord #0             ; space for result
PushLong #DirEntry      ; pointer to directory entry (27 bytes long)
jsl FilterProc
PopWord Result
```

The FilterProc must strip the four bytes off the stack and return with the result at the top of stack.

The filter proc returns with the a result on the stack as follows depending on what it wants to do with the file:

0 if the file is not to be included
1 if the file is to be included but not selectable
2 if the file is to be included and is selectable

The typelist pointer points to a record containing a list of file types to display. The list has the following form:

```
NumEntries      byte
filetype1       byte
filetype 2      byte
...
```

Set the pointer to 0 to display all file types.

The reply pointer points to a reply record described above.

When the dialog is displayed, the files from prefix 0, are shown. When the file is selected, the name of the file is returned in the reply record and prefix 0 is set the directory containing the selected file. If the user cancels the operation, prefix 0 is restored to its original state.

The Disk button works differently from the Drive button in the Macintosh. When a user pushes Next Disk, Standard File first looks at the disk in the same drive the "current" disk is in. If the "current" disk is no longer in that drive, the disk in that drive becomes the current disk. If the "current" disk is still there, Next Disk moves to the next disk in the ProDOS chain. Next Disk works this way because a user can change disks without the system knowing about it.

SFPutFile This is the call that returns a pathname for a file being saved as typed by the user.

```
Stack Before Call
     |  previous contents  |
     |      WhereX         | integer
     |      WhereY         | integer
     |     PromptPtr       | POINTER
     |    OrigNamePtr      | POINTER
     |      MaxLen         | integer
     |     ReplyPtr        | POINTER
     |                     |<-SP

Stack After Call
     |  previous contents  |
     |                     |<-SP
```

WhereX and WhereY describe the location on the screen (in screen coordinates) of the dialog box.

The PromptPtr points to a string to display at the top of the dialog box. This is a standard ProDOS string.

The OrigNamePtr points to a string that holds the original name that appears as the default when the dialog first appears.

The MaxLen parameter specifies the maximum number of characters a user may type. Most applications will use 15 for this value but if the application wants to add a suffix to the file name, it will shorten this value.

The reply pointer points to a reply record described above.

There is no filter proc in SFPutFile since the user needs to know all names on the disk to avoid any naming conflicts.

When the dialog is displayed, the files from prefix 0 are shown. When the file is selected, the name of the file is returned in the reply record and prefix 0 is set the directory containing the selected file. If the user cancels the operation, prefix 0 is restored to its original state.

The Next Disk button works differently from the Drive button in the Macintosh. When a user pushes Next Disk, Standard File first looks at the disk in the same drive the "current" disk is in. If the "current" disk is no longer in that drive, the disk in that drive becomes the current disk. If the "current" disk is still there, Next Disk moves to the next disk in the ProDOS chain. Next Disk works this way because a user can change disks without the system knowing about it.

SFPGetFile This is the call that returns a path name for a file selected by the user using a customized dialog box.

```
Stack Before Call
    |  previous contents  |
    |       WhereX        | integer
    |       WhereY        | integer
    |     PromptPtr       | POINTER
    |    FilterProcPtr    | POINTER
    |    TypeListPtr      | POINTER
    |     DialogPtr       | POINTER
    |    DialogHookPtr    | POINTER
    |     ReplyPtr        | POINTER
    |                     |<-SP

Stack After Call
    |  previous contents  |
    |                     |<-SP
```

All but two inputs are identical to the inputs to SFGetFile.

DialogPtr points to a dialog template in memory. A dialog template is a record passed to the dialog manager call GetNewModalDialog. It contains information about the dialog to be created including a bounds rectangle and a list of pointers to item templates. (See the dialog manager ERS for details.)

It is important that the template include the following items in this order:

Item	Type	ID
Open Button	Button	1
Close Button	Button	2
NextButton	Button	3
CancelButton	Button	4
ScrollBar	Scroll Bar	5
Path	UserItem	6
Files	UserItem	7
Prompt	UserItem	8

The template for the dialog used in SFGetFile is given in the appendix. The bounding rectangle for the Files user item determines how many files may be displayed. You should set the height of this rectangle to 2 plus 10 times the number of files to show. A height of 122 would allow 12 files to be seen.

DialogHookPtr is a pointer to the routine called by SFPGetFile every time ModalDialog returns a item hit. The routine is passed a pointer to the dialog port and a pointer to the item hit word. If the DialogHook routine wants to handle the item hit, it should handle it and set the hit to 0. If the DialogHook routine wants SFPGetFile to handle the item hit, it should leave it unchanged.

The routine is called as follows:

```
PushLong #DialogPort
PushLong #ItemHit
jsl DialogHook
lda ItemHit
```

Your routine must be certain to strip 8 bytes off the stack before returning to SFPGetFile.

SFPPutFile This is the call that returns a pathname for a file being saved as typed by the user using a customized dialog box.

Stack Before Call

previous contents	
WhereX	**integer**
WhereY	**integer**
PromptPtr	**POINTER**
OrigNamePtr	**POINTER**
MaxLen	**integer**
DialogPtr	**POINTER**
DialogHookPtr	**POINTER**
ReplyPtr	**POINTER**
	<-SP

Stack After Call

previous contents	
	<-SP

All but two inputs are identical to SFPutFile. The bounding rectangle for the Files user item determines how many files may be displayed. You should set the height of this rectangle to 2 plus 10 times the number of files to show. A height of 122 would allow 12 files to be seen.

DialogPtr points to a dialog template in memory. A dialog template is a record passed to the dialog manager call GetNewModalDialog. It contains information about the dialog to be created including a bounds rectangle and a list of pointers to item templates. (See the dialog manager ERS for details.)

It is important that the template include the following items in this order:

Item	Type	ID
Save Button	Button	1
Open Button	Button	2
Close Button	Button	3
Next Button	Button	4
Cancel Button	Button	5
ScrollBar	Scroll Bar	6
Path	UserItem	7
Files	UserItem	8
Prompt	UserItem	9
FileName	EditItem	10
FreeSpace	UserItem	11
Create Button	Button	12

The template for the dialog used in SFPutFile is given in the appendix.

DialogHookPtr is a pointer to the routine called by SFPPutFile every time ModalDialog returns a item hit. The routine is passed a pointer to the dialog port and a pointer to the item hit word. If the DialogHook routine wants to handle the item hit, it should handle it and set the hit to 0. If the DialogHook routine wants SFPPutFile to handle the item hit, it should leave it unchanged.

The routine is called as follows:

```
PushLong #DialogPort
PushLong #ItemHit
jsl DialogHook
lda ItemHit
```

Your routine must be certain to strip 8 bytes off the stack before returning to SFPGetFile.