Chapter 11
List Manager

The list manager has been changed to be more flexible for programmers. The bulk of the changes involve allowing one to pass an item number, rather than a list record pointer, to the routines. Furthermore, programmers no longer need to maintain a list record. Only the handle to the list control (CtlHandle) is needed. This handle is returned by the first call to the list manager (either createlist or, preferably, NewControl2).

NOTE: The original list manager documentation does not make clear what values to store in the listView.listMemHeight and listRect field. The following formula must be true for these fields in your list records:

listView * listMemHeight +2 = listRect.v2-listRect.v1

If you pass a zero for the listView field listmanager2 will adjust the listRect.v2 field and set the listView field so that the above condition holds.

NOTE: When passing a listView of zero, the bottom boundary of the listRect may change slightly.

Another bit has been added to the listType field. Bit 2 of listType determines whether the scroll bar will be on the outside (bit2 = 0) of the listRect or on the inside (bit2 = 1). If this bit is set to a one, the list manager adjusts the right side of the listRect to the proper value (on a call to CreateList or NewControl2) and the resets the bit. This works with the old style control records also.

When using resources with the list manager, it is important to leave the ListRef field is unpurgeable when using a sortlist call. The reason is that if the list is purged after it is sorted, the next list manager call will reload the list in its unsorted state.

New Calls

DrawMember2 Call $111C

```
inputs
        ItemNum     WORD   item number to redraw in the list
        CtlHandle   LONG   Handle of the list control
outputs
        none
```

This call functions just like DrawMember except that instead of passing a member pointer, you pass an item number. Passing 0 redraws the entire list.

NextMember2 Call $121C

```
inputs
        space       WORD   room for result
        ItemNum     WORD   item number to start search from
        CtlHandle   LONG   Handle of the list control
outputs
        ItemNum     WORD number of next selected member, 0 if no more
```

This call functions just like NextMember except you pass an item number and Control handle. If you pass 0 it starts seaching from the beginning of the list.

ResetMember2 Call $131C

inputs			
	space	WORD	room for result
	CtlHandle	LONG	Handle of the list control
outputs			
	ItemNum	WORD	number of next selected member 0 if no more

This call functions just like ResetMember except you pass the lists control handle and get back an item number. ItemNum is 0 if there are no records selected.

SelectMember2 Call $141C

inputs			
	ItemNum	WORD	item number to select
	CtlHandle	LONG	Handle of the list control
outputs			
	none		

This call functions just like SelectMember except that you pass a control handle and an item number.

SortList2 Call $151C

inputs			
	ComparePtr	LONG	pointer to comparison routine. NIL for standard compare
	CtlHandle	LONG	Handle of the list control
outputs			
	none		

This call functions just like SortList except that you pass it a control handle.

NewList2 Call $161C

inputs			
	DrawPtr	LONG	Pointer to draw routine
	ListStart	WORD	Member to draw at top of list
	ListRef	WORD	Pointer, handle, or resource ID of list.
	listRefVerb	WORD	Describes ListRef: 0 indicates pointer, 1 is a handle, 2 is resource ID
	ListSize	WORD	Number of items in the list
	CtlHandle	LONG	Handle of the list control returned by NewControl2
outputs			
	none		

This call is similar to NewList. Instead of passing a pointer to a list record, all of the relevent parameters are passed on the stack. A negative one ($FFFF or $FFFFFFFF) for the ListStart, ListRef, listRefVerb, DrawPtr, or ListSize indicates that the field should not change. A zero for DrawPtr indicates the default draw routine should be used, and any non-zero value indicates the address of a custom draw routine.