Chapter 5
Desk Manager

Three significant changes have been made to the desk manager for this release. first. to allow users more flexibility we have added a scrollable CDA menu that allows more than 15 items in the CDA menu, second, for authors of DA's we have added two calls that allow you to de-install CDA's and NDA's from the system, this way you can try out DA's without rebooting, and we have added a run queue, where items in this queue get called at system task time.

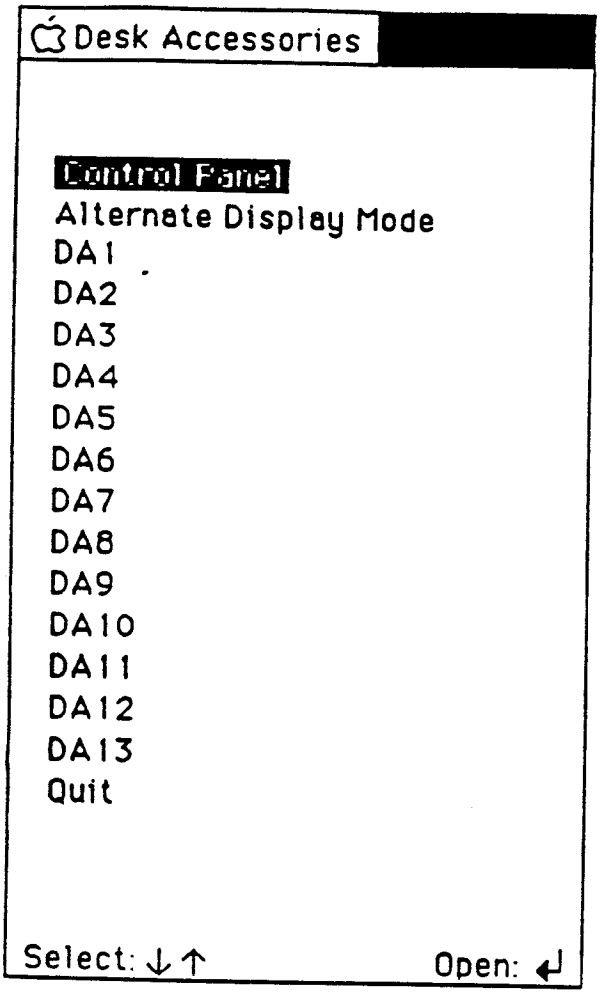
Scrollable CDA Menu

Operation of the scrolling CDA menu is simple and intuitive. As before, the up and down arrow keys are used to traverse the menu. The 'esc' key selects quit. Pressing 'enter' selects the highlighted menu item. In this new version, holding the open-apple key while pressing the up or down arrow keys moves by a page in the specified direction.

The only other changes from the previous version are that now one cannot wrap around the menu from the top to the bottom or vice versa and the menu scrolls if there are more items to be displayed. When returning to the CDA Menu from a desk accessory, the name of that desk accessory (and not the control panel) is highlighted to indicate the current choice. Only system memory limits the number of classical desk accessories which can be installed. Furthermore, an indicator shows when the user may scroll the menu up or down.

CASE A: Thirteen or fewer user CDAs installed, no scrolling necessary.

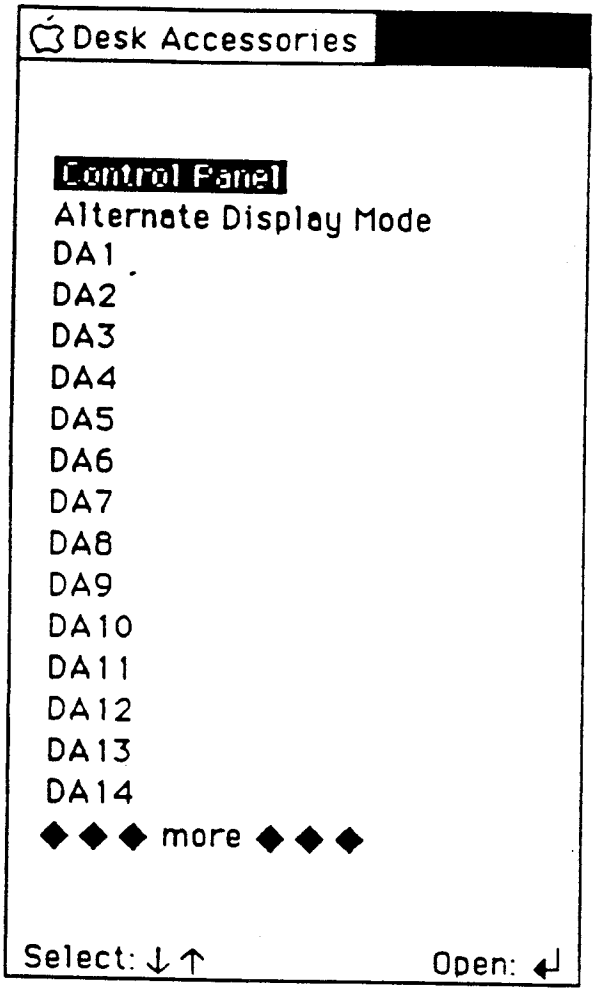
When the select classical desk accessories menu first comes up and thirteen or fewer CDAs are installed, it appears as follows:



In this case, the operation of the CDA menu is exactly as before with the exceptions noted above. Obviously, no scrolling takes place.

CASE B: Fourteen or more user CDAs installed, scrolling occurs.

In this case, the CDA menu contains a message indicating there are more CDAs to be displayed, as is shown below.



When the desk accessories scroll (so that control panel is no longer the top item), the 'more' message will appear at the top also. When the user is at the bottom of the list (at the quit item) the 'more' message on the bottom will disappear.

2. Operation of the run queue

The run queue allows the programmer to install tasks which need periodic calls. The run queue is examined at system task time (a time when the system is guaranteed free and all tools are availabe), and any item whose period has elapsed is called.

Each run item has a header which must be set up as follows:

```
Reserved      LONG          ;used as link to next task in queue
Period        WORD          ;indicates how often this item is to be called (in ticks)
Signature     WORD $A55A    ;used to make sure run queue item is not corrupt
Reserved      LONG          ;used to keep track of when this item was last called
CodeEntry                   ;Code must start here.
```

The period field of the header describes how often the item should be called. A period of 0 indicates the run item will be called as often as possible, while a period of $FFFF indicates the item should never be called. Any other value indicates that the item should be called after that

number of sixtieths of a second have elapsed. For example, a period of 60 indicates the run item should be called about every second.

NOTE: The accuracy of the timing of how often tasks are called is dependent on how often the application calls system task. The error in the period is usually not significant for periods of 60 (every second) or longer, but could be significant to very short periods. This is not an accurate timing device!

Since the RunQ entries are not called by you directly, there are some things you should know about creating them. You do not have to check the busy flag, since the system is gauranteed to be free when you are called. Since Systemtask is called by taskmaster (which can now be called by DA's) you have no gaurantee that any specific environment will exist, so be sure to save the current environment and restore it when you are done. Also, since runQ entries can survive between applications, either be sure to remove your task when your program finishes, or insure that it does not get purged from memory when your app is complete.

You also may notice a similarity between the Run queue and the heartbeat queue, they are similar but they are not exacly the same, the header on the run queue entry contains an extra field, and the task is not removed from the run queue if the count is left at 0.

AddToRunQ ($1F05)

Input
HeaderPtr LONG

Output
None

Possible Errors:
None

Adds the indicated routine to the run queue. The input is a pointer to the run items header record, the end of this chapter contains an example RunQ routine.

RemoveFromRunQ ($2005)

Input
HeaderPtr LONG

Output
None

Possible Errors:
NotInList
InvalidTag

Removes the indicated routine from the run queue. The input is a pointer to the run items header record, the end of this chapter contains an example RunQ routine.

3. Operation of RemoveCDA and RemoveNDA

These two routines are the complements of InstallCDA and InstallNDA.

RemoveCDA ($2105)
Input
idHandle LONG

Output
None

Possible Errors:
DaNotFound $0510

Removes the indicated CDA from the system. This routine does **not** dispose the handle.

RemoveNDA ($2205)
Input
idHandle LONG

Output
None

Possible Errors:
DaNotFound $0510

Removes the indicated NDA from the system. This routine does **not** dispose the handle

RunQ task example

The following is an example runQ example that will beep the speaker every 15 minutes.

```
; RunQ example task that beeps every 15 minutes.
; it is provided in MPWIIgs assembler format. The first portion is the
; task header.
BeepHdr     Record
            ds.L 1                  ; reserve 1 word for link to next runQ entry
period      dc.W $D2F0              ; number of 60th of a sec (54000=15 minutes)
            dc.W $A55A              ; signature used to test for queue integrity
            dc.L 0                  ; used by desk mgr to keep track of the time
            EndR
;
; Now the actual code of the task goes here.
BeepTask    Proc
            with BeepHdr

            _SysBeep                ; beep the speaker once

            lda #$D2F0              ; and now recharge the period for next call
            sta >period             ; NOTE:Use long addressing: DataBank unknown
            rtl                     ; and to exit use an RTL
            EndP
```

The following code can be used to install the above task into the Run queue

```
PushLong #BeepHdr
ldx #$1F05
```