Desk Manager
External Reference Specification

Steven Glass
John Worthington

July 11, 1986

November 26, 1985	Initial Release.
January 9, 1986	Major Changes to the Data Structures
June 18, 1986	Additional changes reflecting code actually in ROM
July 11, 1986	Changes reflect code implemented for NDAs

Summary

The Desk Manager provides the user access to desk accessories. A desk accessory is a "mini-application" that can be run at the same time as a Cortland application.

The Desk Manager provides support for two types of desk accessory's: Classic Desk Accessories (CDA) and New Desk Accessories (NDA).

Classic Desk Accessories are desk accessories that are designed to execute in a non-desktop, non-event based environment. Unlike NDAs, a classic accessory gets full control of the machine during what is basically an interrupt state (generated by a keypress). The desk accessory is responsible for saving any of the application's memory that it uses as well as handling all I/O.

New Desk Accessories are Macintosh Style desk accessories that are designed to execute in a desktop, event based environment. NDAs run in a window and get "control" when that window is the topmost window. Just what kind of control a NDA has is described below.

How CDA's are Used

A user activates a CDA from the CDA menu. The CDA menu is displayed by pressing OPEN APPLE-CONTROL-ESCAPE. Two CDA's are built into the system:

Control Panel
Alternate Display Mode

Any others (up to eleven) are loaded from disk. From the CDA menu, a user can select any of the DA's currently in the system. The desk accessory is activated and retains control until it shuts down. When it shuts down, the Desk Manager re-displays the CDA Menu. Only when the user selects Quit from the CDA menu does the original application resume operation.

When can the CDA Menu be displayed?

The Desk manager gets control whenever the user presses OPEN-APPLE CONTROL-ESC. Before it displays the CDA Menu, it checks the system busy flag. If something in the system is busy, the Desk Manager schedules a wake-up with the scheduler. The next time the system flag is free, the scheduler will wake up the Desk Manager which then can display the CDA menu. This guarentees that CDA's have all system resources available to them when they are called.

Execution Environment

Classic Desk Accessories have a single entry (activation) point. When the CDA gets control, the processor is in full native mode (16-bit m and x registers). The desk accessory menu is still displayed on the screen in whatever was mode requested by the user (in the control panel). The CDA must execute the RTL in full native mode for the Desk Manager to work correctly.

When the desk manager displays the CDA menu, it saves the text pages in bank 0 and 1, $E0 and $E1 along with pages 0 and 1 of bank 0 (system direct page and stack). (Only the screen holes used by the Desk Manager are preserved.) These parts of memory are restored by the Desk Manager when the user selects Quit from the CDA menu. Thus a

CDA can feel free to use almost all of this memory as it sees fit. The exception is the stack. Since, the Desk Manager's return address is on the stack (along with other Desk Manager variables), the CDA cannot cut the stack back any farther than it is when it gets control.

A CDA must take care using any other memory in the system that it does not already own. A CDA can use the Memory Manager to obtain additional memory outside "special memory", but it cannot rely on being able to obtain any more of bank 0 and 1.

Form In Memory and On Disk

Classic Desk Accessories have a simple form. They are load files kept on the system disk in the DESK.ACCS subdirectory of the SYSTEM directory and have a file type $B9. The CDA starts with an idetification section as follows.

```
StartOfDA   dc i1'NameLength'      ; this combined with the characters that
            dc c'Name of DA'       ; follow make a ProDOS string
            dc i4'StartOfDACode'   ; This is a pointer to the start of the code
```

The identification structure contains the name of the desk accessory and a pointer to the activation routine.

How NDAs are Used

New Desk Accessories are loaded by the operating system at boot time. An application that wants to make NDA's available to the user does not have to do a lot of work. If the Application uses TaskMaster, it need only make three calls:

DeskStartup	to initialize the Desk Manager
FixAppleMenu	to put the list of NDAs in the Apple Menu
DeskShutdown	to shut down the Desk Manager

TaskMaster will handle opening NDAs in response to menu selections, calling SystemTask and SystemClick when appropriate. Calling SystemEdit when a selection is made from the Edit Menu, and closing a desk accessory in response to the Close item of the File Menu.

Applications that do not use TaskMaster must do the following to support new desk accessories.

call DeskStartup	To initialize the Desk Manager.
call FixAppleMenu	To put the list of NDAs in the Apple Menu
call OpenNDA	When the user selects an NDA from the Apple Menu
call SystemTask	Frequently (at least every time through the event loop).
call SystemClick	When a MouseDown event occurs in a system window.
call SystemEdit	When a desk accessory is active and the user selects undo, cut, copy, paste or clear from the edit menu.
close an NDA	When the user selects close from the file menu. You can use CloseNDA or CloseNDAbyWinPtr to do this.
DeskShutdown	To shut down the Desk Manager

Execution Environment

NDAs have four entry points: open, close, action and init. For each of these entry points the processor is in Full Native Mode. There is no direct page available so the NDA must obtain it from the stack. The open routine returns a long word on the stack. The action routine is passed information in all three registers.

A NDA can assume that the following tools are loaded and initialized:

QuickDraw	
Event Manager	
Window Manager	
Menu Manager	
Control Manager	
Scrap Manager	
LineEdit	?
Dialog Manager	?
Print Manager	?

(Note there may be others. This part is not well thought out yet.). The NDA is responsible for saving and restoring important globals like the current graf port (other important globals will be added to this list as we think of them).

Form In Memory and On Disk

New Desk Accessories have a different form than CDAs. They are still load files kept on the system disk in the DESK.ACCS subdirectory of the SYSTEM directory but they have a file type $B8 and The NDA starts with an identification section as follows.

```
StartOfDA   dc i4'PtrToOpen'          ; Pointer to the open routine
            dc i4'PtrToClose'         ; Pointer to the close routine
            dc i4'PtrToAction'        ; Pointer to the action routine
            dc i4'PtrToInit'          ; Pointer to the init routine
            dc i2'Period'             ; How often the NDA gets run codes
            dc i2'EventMask'          ; Describes what events it wants
            dc c' MenuLine \H**'      ; The text which describes the menu item
```

The open routine must return a pointer to its window on the stack. When it calls the open routine, the desk accessory manager puts 4 bytes of zero on the stack before it pushes the RTL address.

The close routine has no inputs and no outputs.

The action routine is passed the action code in the a register. When the action is to handle an event, the x and y registers contain a pointer to the event record (low word in x, high word in y). The possible action codes are:

Event	1	The only events that can be passed to a DA are ButtonDown, ButtonUp, KeyDown, AutoKeyDown, Update and Activate. Update and Activate events for a desk accessory are always passed on. The first three are passed on only if the EventMask indicates they should be passed on.

Run	2	The time period specified has passed.
Cursor	3	This is passed to a desk accessory if it is the front window each time SystemTask is called. The purpose is to allow the desk accessory to change the cursor when it is over the NDA's window.
Menu	4	This is passed to a desk accessory if an item from a system menu is selected. The item id is passed in the a register, the menu id is passed in the x register.
Undo	5	This is passed to a desk accessory if the application
Cut	6	determines that the user has selected one of these edit
Copy	7	commands from Edit menu. The action call should
Paste	8	return a boolean in the A-register indicating whether
Clear	9	or not the the command was handled.

The InitRoutine is a routine that is called every time DeskStartup is called for each NDA that is installed.

The Period field describes how often the DA should be called with the "Run" action code. A period of 1 is every 60th of a second. A period of 2 is every 30th of a second. A period of 60 is every second. A period of $FFFF is never. A period of 0 is as often as possible. The action routine is called with the "run" code from SystemTask. The application should be calling SystemTask every time through its event loop.

The MenuLine is a line of text that will be passed to the Menu Manager to appear in the Apple Menu. The line must start with a waste character since the MenuManager puts something here. The line must also have a back slash in it (\) and an H followed by two place holder characters. These characters will be replaced with the menu item ID for the desk accessory when FixAppleMenu is called.

Desk Manager Routines

The desk manager has a number of calls but very few that a user would make. These are described below.

Initialization Calls

DABootInit — Internal routine called at boot time to initialize the DA manager.

Stack Before Call

	previous contents
	<-SP

Stack After Call

	previous contents
	<-SP

DAStartup — Call made by an application before it makes any other desk manager calls.

Stack Before Call

	previous contents
	<-SP

Stack After Call

	previous contents
	<-SP

DAShutdown — Call made by an application before shutdown if it has made any desk manager calls.

Stack Before Call

	previous contents
	<-SP

Stack After Call

	previous contents
	<-SP

DAVersion — Returns the version number of the Desk Manager.

Stack Before Call

	previous contents
	space for version
	<-SP

Stack After Call

	previous contents
	version number
	<-SP

DAReset — Resets the Desk Manager.

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

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

DAStatus — Returns whether or not the Desk Manager's startup call has been issued.

```
Stack Before Call
     |  previous contents     |
     |  space for boolean     |
     |                        |<-SP

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

State Save Calls

The state saving calls are internal routines used by the desk manager to preserve the machine state. Careless use of any of these calls could prevent an application being interrupted from running when the current interrupt is over.

SaveScrn — SaveScreen will save the 80-column text screens in bank 00, 01, E0 and E1. This new image of the screen will be used for subsequent calls to RestScreen.

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

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

An important thing to note is that only the screen holes used by the Desk Manager are preserved.

RestScrn — RestoreScreen will restore the screen area saved by the Desk Manager.

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

Stack After Call
```

```
|  previous contents  |
|                     k-SP
```

An important thing to note is that only the screen holes used by the Desk Manager are preserved.

SaveAll — Saves all the variables that the Desk Manager preserves when the CDA menu is activated.

```
Stack Before Call
|  previous contents  |
|                     k-SP

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

RestAll — Restores all the variables that the Desk Manager preserves when the CDA menu is activated.

```
Stack Before Call
|  previous contents  |
|                     k-SP

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

HouseKeeping

InstallNDA — Installs the new desk accessory in the system.

```
Stack Before Call
|  previous contents  |
|    handle to ID     | handle pointing to ID structure of DA
|                     k-SP

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

InstallCDA — Installs the classic desk accessory in the system

```
Stack Before Call
     |  previous contents     |
     |    handle to ID        | handle pointing to ID structure of DA
     |                        k-SP

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

Classic Desk Accessory Routines

ChooseCDA — Activates the Desk Manager and displays the CDA menu.

```
Stack Before Call
     |  previous contents     |
     |                        k-SP

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

ChooseCDA causes the Desk Manager to display the CDA Menu as if the user key stroke interupt has occured. I'm not sure there is any valid reason for a program to make this call.

SetDAStrPtr — Lets a program change the names of the Classic Desk Accessories.

```
Stack Before Call
     |  previous contents     |
     |    DA id num           |
     |    Pointer to Str      |
     |                        k-SP

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

This routine is used to localize the desk accessories in ROM. It allows the built-in desk accessories to have different names.

GetDAStrPtr — Returns the pointer to the name string of a classic desk accessory.

```
Stack Before Call
     |  previous contents     |
     |    space for result    |
     |    DA id num           |
     |                        k-SP
```

```
Stack After Call
     |  previous contents     |
     |  pointer to DA str     |
     |                        k-SP
```

New Desk Accessory Routines

OpenNDA — Opens the specified DA by ID number.

```
Stack Before Call
     |  previous contents     |
     |  Space for RefNum      | word
     |      ID Num            | ID number returned from Menu Manager (word)
     |                        k-SP

Stack After Call
     |  previous contents     |
     |      RefNum            |
     |                        k-SP
```

This call is made when an application discovers that the user has selected an NDA from the Apple Menu. The ID Num passed is the same ID returned by the Menu Manager and set up by the FixAppleMenu call.

CloseNDA — Closes the specified DA.

```
Stack Before Call
     |  previous contents     |
     |      RefNum            |
     |                        k-SP

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

The RefNum is the RefNum returned by the open call. This call is very similar to the call on the Macintosh. Like the Macintosh, it is unlikely to be used by an application since NDAs are closed when the mouse goes down in the CloseBox and SystemClick handles this.

CloseNDAbyWinPtr — Closes the NDA whose window pointer is equal to the one that is passed.

```
Stack Before Call
     |  previous contents     |
     |      WinPtr            | POINTER
     |                        k-SP

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

This call is handy when trying to close a desk accessory because the user chose close from the file menu. When the user chooses close, the application uses the FrontWindow call to see what window is to be closed. If the front window is not an application window, the application can pass the pointer to CloseNDAbyWinPtr.

Possible Errors

NotSysWindow	This is returned when the window pointer is not the pointer to a window owned by a NDA.

CloseAllNDAs — Closes all open NDAs.

No stack parameters.

FixAppleMenu — Adds the names of the NDAs to the Apple menu.

Stack Before Call

previous contents	
StartingID	ID to use for first NDA (**word**)
	←SP

Stack After Call

previous contents	
	←SP

This call is used to put the names of the currently installed NDAs in the Apple menu. They are appended to the Apple Menu and the ID of the first NDA name is the ID passed. The next NDA gets ID+1 and so on.

GetNumNDAs — Returns the number of NDAs currently installed.

Stack Before Call

previous contents	
Space for integer	**word**
	←SP

Stack After Call

previous contents	
Number of NDAs	**integer**
	←SP

SystemClick — Called when application detects mouse down in a system window.

Stack Before Call

previous contents	
Ptr to EvtRecord	
Window Ptr	
FindWindow Result	
	←SP

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

This call is slightly different than the equivalent Macintosh call. One additional input is passed on the stack. This additional input is the result of the FindWindow call (that is where in the system window the mouse when down). This is different because the Desk Manager has no way to find out this information unless it is passed by the application.

Note: If the application is using TaskMaster, it never needs to make this call. TaskMaster does the work for it.

SystemEdit — Passes standard menu edits to system windows.

```
Stack Before Call
    |   previous contents       |
    |   Space for result        |
    |      EditType             |  Word
    |                           |<-SP

Stack After Call
    |   previous contents       |
    |   Processed Flag          |  BOOLEAN
    |                           |<-SP
```

The valid Edit Types are

1 Undo
2 Cut
3 Copy
4 Paste
5 Clear

The processed flag returns true if the top window is a system window and false otherwise.

SystemTask — Called periodically by an application to support Desk Accessories doing periodic actions.

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

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

For eack open desk accessory, SystemTask causes the accessory to perform the periodic action defined for it, if any such action was defined and if the proper time period has passed since the action was last performed. For example, a clock accessory can be defined such that the second hand is to move once every second; the periodic action for the accessory will be to move the second hand to the next position, and SystemTask will alert the accessory every second to perform that action.

Note: If the application is using TaskMaster, it never needs to make this call. TaskMaster does the work for it.

SystemEvent — This is the entry point the Event Manger uses to the Desk Manager.

Stack Before Call

previous contents	
space for boolean	word
Event What	word
Event Message	long
Event When	long
Event Where	point (long)
Event Mods	word
	<-SP

Stack After Call

previous contents	
boolean	
	<-SP

System Event returns true if the event is processed by a DA and false if it is to be sent on to the application. The CDA activation keystroke is processed in this way.

Note: An application would never make this call.

SystemMenu — Called by the Menu Manager when a desk accessory adds a menu to the System Menu Bar and an item from it is selected.

Stack Before Call

previous contents	
MenuID	
ItemID	
	<-SP

Stack After Call

previous contents	
	<-SP

Note: An application would never make this call.

Summary of Calls

Required Tool Calls

DeskBootInit
DeskStartup
DeskShutdown
DeskVersion
DeskReset
DeskStatus

State Saving Calls

SaveScrn
RestScrn
SaveAll
RestAll

HouseKeeping Calls

InstallCDA
InstallNDA

Classic Desk Accessory Calls

ChooseCDA
SetDAStrPtr
GetDAStrPtr

New Desk Accessory Calls

OpenNDA
CloseNDA
CloseNDAbyWinPtr
CloseAllNDAs
FixAppleMenu
GetNumNDAs
SystemClick
SystemEdit
SystemTask
System Event
SystemMenu