

## Why Dialog Manager?

This document is the result of a couple of weeks work preparing a presentation on desk accessories for the November '86 meeting of the MACincinnati Hacker Group. It is a DA shell in TML Pascal the fields typical events that occur for a DA and executes most of its functions using the dialog manager. It has a 'draggable' window, radio buttons, text boxes, a menu, and buttons, buttons, buttons. It shows how to put up ModalDialogs without reentrancy problems, use global data, and handle activate/deactivate calls. And it shows you three ways to close a DA.

Its only 'features' are to multiply two integers and adjust your Mac clock to Daylight Savings Time...

Contained in this .pit file should be:

DialogManager.Doc	this MacWrite file.
DialogManager.acc	the executable desk accessory file.
DialogManagerDA.Pas	the Pascal source code.
DialogManagerDA.R	the RMaker source code.
DialogManagerDA.resed	the resource file, created with ResEdit.

These files were created and compiled in an HFS folder named 'Development' on a volume (disk) named '20Meg'. In order to compile them you'll need version 2.0 of TML Pascal, RMaker, and the TML Linker. You will also need a USES file named 'MacIntf' in the 'Development' folder. You may use a different folder arrangement than I have used, but you'll have to insert your own paths in the proper places in the .Pas and .R files.

I would welcome all questions and comments.

Bill Johnson  
MacCincinnati Hacker Group  
Compuserve: 72257,2064  
GEnie: W Johnson  
MacCincinnati BBS: (606) 572-5375

## TYPE

```
DAGlobals = record          { Global data. Store a handle to it in Device.dctlStorage }          Finished :
boolean;
    ButtonNo :    longint;
    Number:      longint;
    NoText :     longint;
    menuHand :   menuHandle;
    StringArray : array[1..NoStrings] of str25; end;

DeviceControlRec = record   {replace predefined DCtlEntry record in interface files }          dCtlDriver      :
Handle;
    dCtlFlags      : Integer;
    dCtlQueue      : Integer;
    dCtlQhead      : Lptr;
    dCtlQtail : Lptr;
    dCtlPosition   : longint;
    dCtlStorage    : DAGlobalsH; {define as handle to global data}          dCtlRefNum      : integer;
    dCtlCurTicks : longint;
    dCtlWindow     : GrafPtr;
    dCtlDelay: integer;
    dCtlEmask      : integer;
    dCtlMenu : integer;
end;
```

# Procedures and Functions

---

## Description/Dependencies

### Low-Level Dialog Procedures

PROCEDURE **ItemActivate** (theDialog : DialogPtr; ItemNo, HiliteMode : integer);  
==Used to activate, deactivate, and hilite buttons==

PROCEDURE **SetRadioButton** (theDialog: DialogPtr; ItemNo, ChkMark : integer);  
==Used to, well, set radio buttons==

FUNCTION **GetTxtValue** (theDialog : DialogPtr; ItemNo : Integer): longint;  
==returns the numeric value of a textbox. Textbox must only contain integers==

### Drawing Procedures

PROCEDURE **HiliteDefButton** (theDialog: DialogPtr);  
==Draw the hiliting around the default button==

PROCEDURE **DrawEText** (VAR DataRec: DAGlobals);  
==Draws user-entered text in the dialog window==

PROCEDURE **DitlText**(VAR DataRec: DAGlobals);  
==Draws text for the DA after the Dialog Mgr is done==  
==Calls: DrawEText();==

PROCEDURE **DitlLines**;  
==draw dotted lines in main window==

### Setup Procedure

PROCEDURE **InitDlogStatus**(VAR device: DeviceControlRec);  
==set buttons, cursor location, and initialize variables==

### ModalDialog Procedures

PROCEDURE **DoAbout** (DlogID : longint);  
==put up the About Box, or the Command key dialog==

PROCEDURE **EnterText** ( VAR Device: DeviceControlRec);  
==put up modal dialog to accept text input in response to selection of the ICON==

### Activate, Deactivate, Update Procedures

PROCEDURE **UpdateDA**(VAR device: DeviceControlRec);  
==field Update calls, draw the window contents==  
==Calls: DitlLines(); DitlText(); HiliteDefButton(); ==

PROCEDURE **DAactivate**(VAR device: DeviceControlRec);  
==activate call, put menu on the menu bar==

PROCEDURE **DAdeactivate**(VAR device: DeviceControlRec);  
==deactivate call, remove menu from the menu bar==

## Button Procedures

PROCEDURE **CtlRadioButtons** (VAR Device : DeviceControlRec; WhichButton : integer);  
==Manage the three radio buttons in the main dialog window==  
==Call: SetRadioButton ();==

PROCEDURE **DoDefButton**(VAR Device: DeviceControlRec);  
==Enable the Reset button & multiply the contents of the text boxes==  
==Call: GetTxtValue ();==

PROCEDURE **DoResetButton**(VAR Device: DeviceControlRec);  
==Clear and redraw the window==

PROCEDURE **DoTimeButton**(VAR Device: DeviceControlRec);  
==Put up Alert and set the clock chip ahead or back an hour==

## Event Procedures

PROCEDURE **doKeyEvent** (VAR Device: DeviceControlRec; EventPtr : EventPointer);  
==Handle all keyboard events from here==  
==Call: CtlRadioButtons (); DoResetButton(); DoTimeButton(); DoDefButton();==

PROCEDURE **Event**(VAR Device : DeviceControlRec; VAR Block : ParamBlockRec);  
==Handle \_all\_ events passed to the desk accessory==  
==Call: DoDefButton(); DoAbout(); DoResetButton(); DoTimeButton(); CtlRadioButtons ();==  
==Call: doKeyEvent(); UpdateDA(); DAactivate(); DAdeactivate();==

## OPEN Procedure

PROCEDURE **Open**(VAR Device : DeviceControlRec; VAR Block : ParamBlockRec);  
==Create main window, allocate globals, set WindowKind field, initialize variables, get Menu handle==  
==Call: InitDlogStatus();==

## CONTROL Procedure

PROCEDURE **Ctl**(VAR Device : DeviceControlRec; VAR Block : ParamBlockRec);  
==Handle all calls made to the desk accessory - accEvent, accRun, accMenu==  
==Call: Event(); DoAbout();==

## CLOSE Procedure

PROCEDURE **Close**(VAR Device: DeviceControlRec; VAR Block : ParamBlockRec);  
==Remove menu, dispose of window, dispose of globals, release Menu handle, set driver purgeable==  
==Call: DAdeactivate();==

## No Main Procedure Allowed

# Owned Resources

---

A desk accessory can have its own resources. But in order for those resources to be moved by the Font/DA Mover, and avoid conflicts with other resources in the System file (the final destination of all desk accessories) they must have resource ID numbers within a designated range; 32 resource ID per DA starting at \$C000.

Owned resources can be calculated as follows:

$$\text{\$C000} + (32 * \text{DRVR ID\#})$$

Since the DialogManagerDA is assigned the ID number 16, the owned resources will be in the range of (\text{\\$C200} -> \text{\\$C20F}), or -15872 through -15857.

## Resource File

---

DialogManagerDA.resed

The resources in this file were created with ResEdit. They could all be generated while compiling resources with RMaker (the same way that the menu resource is), but it's easier to create them with ResEdit.

### Alert Templates

ALRT, -15872 (sub ID #0)	Do-nothing alert in response to the Bullseye.
ALRT, -15871 (sub ID #1)	Alert for time change, w/ Cancel, Spring, and Fall buttons.

### Dialog Item List

DITL, -15872 (sub ID #0)	Dialog item list for main window.
DITL, -15871 (sub ID #1)	Dialog item list for the 'About' dialog.
DITL, -15870 (sub ID #2)	Dialog item list for the text entry dialog.
DITL, -15869 (sub ID #3)	Dialog item list for the 'Command Keys' dialog.
DITL, -15868 (sub ID #4)	Dialog item list for the time change alert.
DITL, -15867 (sub ID #5)	Dialog item list for the bullseye alert.

### Dialog Template

DLOG, -15872 (sub ID #0)	Dialog template for main window.
DLOG, -15871 (sub ID #1)	Dialog template for the text entry dialog.

### ICON

ICON, -15872 (sub ID #0)	Bullseye ICON. Called by the main dialog item list.
ICON, -15871 (sub ID #1)	Text entry ICON. Called by the main dialog item list.

# Resources

---

## RMaker

The following is the RMaker file for DialogManager DA.

It's functions are to:

- a) Create an output file DialogManager.acc with correct TYPE & CREATOR
- b) Convert the .Rel file to DRVr resource & assign it an ID# of 16
- c) Create a MENU resource that is owned by DRVr #16
- d) Copy the resources from the DialogManagerDA.resed file into this file

## RMaker File

---

### DialogManagerDA.R

Note the pathnames that precede the filenames. All input and output files are in an HFS folder named 'Development' on a volume named '20Meg'.

\* DialogManagerDA.R

\* Resource File for DialogManagerDA.Pas

\*Output file name and the Type and Creator flags

20Meg:Development:DialogManager.acc

DFILDMOV

TYPE DRVr = PROC

DialogManager DA,16

20Meg:Development:DialogManagerDA

type MENU

DManMENU,-15872

DialogManager;; menu title

Command Keys;; menu1

(-;; dotted line

About...;; menu3

Quit;; menu4

INCLUDE 20Meg:Development:DialogManagerDA.resed