



Welcome to the

Palm OS 2.0 Cookbook for Windows

Navigate this online document as follows:

To see bookmarks	Type Ctl-7
To see information on Adobe Acrobat Reader	Type Ctl-?
To navigate	Click on any blue hypertext link any Contents entry arrows in the menu bar





U.S. Robotics®

**Palm OS™ 2.0 Cookbook
for Windows**

**Some information in this manual may be out of date.
Read all Release Notes files for the latest information.**

©1996, 1997 U.S. Robotics, Inc. All rights reserved.

Documentation stored on the compact disk may be printed by licensee for personal use. Except for the foregoing, no part of this documentation may be reproduced or transmitted in any form by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from U.S. Robotics.

U.S. Robotics, the U.S. Robotics logo and Graffiti are registered trademarks, and Palm Computing, HotSync, the Palm OS, and the Palm OS logo are trademarks of U.S. Robotics and its subsidiaries.

All other trademarks or registered trademarks are the property of their respective owners.

ALL SOFTWARE AND DOCUMENTATION ON THE COMPACT DISC ARE SUBJECT TO THE LICENSE AGREEMENT.

Contact Information:

Metrowerks U.S.A. and international	Metrowerks Corporation 2201 Donley Drive, Suite 310 Austin, TX 78758 U.S.A.
Metrowerks Canada	Metrowerks Inc. 1500 du College, Suite 300 Ville St-Laurent, QC Canada H4L 5G6
Metrowerks Mail order	Voice: 1-800-377-5416 Fax: 1-512-873-4901
U.S. Robotics, Palm Computing Division Mail Order	U.S.A. and Canada: 1-800-881-7256 elsewhere 1-408-848-5604
Metrowerks World Wide Web	http://www.metrowerks.com
U.S. Robotics, Palm Computing Division World Wide Web	http://www.usr.com/palm
Registration information	register@metrowerks.com
Technical support	support@metrowerks.com
Sales, marketing, & licensing	sales@metrowerks.com
CompuServe	goto Metrowerks

Contents

Contents	.5
About This Document	.9
Palm OS SDK Documentation	.9
What This Guide Contains	10
Conventions Used in This Guide	10
1 Designing Palm OS Applications	11
Application Design Process	12
Designing Your Application	13
Getting Started	13
Setting Up a Preliminary User Interface	14
Summary of UI Design Rules	16
Summary of UI Design Recommendations	16
Creating a User Interface Prototype	17
Integrating Programs With the Palm OS Environment	17
Avoiding Potential Pitfalls	19
Achieving Optimum Performance	20
Making your Application Run on Different Devices	21
Working With Databases	22
User Interface Guidelines	23
Understanding the Palm OS UI Design Philosophy	24
Creating Fast Applications	24
Matching Use Frequency and Accessibility	24
Creating a Palm OS User Interface	26
Navigation Guidelines	27
Preferences Guidelines	28
Data Entry Guidelines	28
Command Execution Guidelines	30
Guidelines for Screen Layout	31
Guidelines for Dialog Box Layout	32
Creating Easy-to-Use Applications	33
Palm OS Resource Selection: List or Table?	34
List Resource	34

Contents

Tables	35
Summary	35
Localization Guidelines	36
General Localization Guidelines	36
Palm OS Localization Guidelines	37
2 Debugging in Standalone Mode on the Device	39
Debugging on the Device with CodeWarrior Debugger	39
Resetting the Device	41
Soft Reset	41
Soft Reset + Up Arrow	41
Hard Reset	42
Copying a Database From the Device to the PC	43
3 PalmPilot Console Commands	45
System Commands	46
Card Info Commands	46
Heap Utility Commands	47
Chunk Utilities.	47
Database Utilities	48
Miscellaneous Utilities	50
Record Utilities	51
Resource Utilities.	52
Debugging Utilities.	53
Debugging Memory Management	54
Displaying Memory Information	55
CardInfo	55
Chunk Info	56
Heap Check	56
Heap Dump	57
Heap List	57
Heap Total	58
Manipulating Memory	58
Heap Compact.	59
Heap Fill	60
Heap Scramble.	60

Memory Manager Debug Mode	61
4 Testing Palm OS Applications	65
The PalmPilot Compatibility Program	65
Creator IDs	66
Testing Application Integration	66
Debugger Nub Signal	68
Index	69

Contents



About This Document

The Palm OS Cookbook is part of the Palm OS Software Development Kit (SDK). This introduction provides an overview of the SDK documentation, discusses what materials are included in this document and what conventions are used.

Palm OS SDK Documentation

The following documents are part of the SDK:

Document	Description
Palm OS 2.0 Tutorial	21 Phases step developers through using the different parts of the system. Example applications for each phase are included in the SDK.
Developing Palm OS 2.0 Applications. Part I: Interface Management	A programmer's guide and reference document that discusses all important aspects of developing an applications.
Developing Palm OS 2.0 Applications. Part II. System Management.	A programmer's guide and reference document for all system managers, such as the string manager or the system event manager.
Developing Palm OS 2.0 Applications, Part III. Memory and Communications Management	Programmer's guide and reference document about <ul style="list-style-type: none">• Memory management; both the database manager and the memory manager.• The Palm OS communications library for serial communication.• The Palm OS network library, which provides basic network services.
Palm OS 2.0 Cookbook.	Provides a variety of design guidelines, including localization, UI design, and optimization. Information about using CodeWarrior for Pilot to create projects and executables.

About This Document

What This Guide Contains

What This Guide Contains

This section provides an overview of the chapters in this guide.

- [Designing Palm OS Applications](#) provides information that's helpful for all stages of the application development process. This includes
 - Following interface design guidelines
 - Integrating well with other applications
 - Preparing for localizationThere's also information on [Achieving Optimum Performance](#) and on [Avoiding Potential Pitfalls](#).
- [Debugging in Standalone Mode on the Device](#) explains how to download your application to the device for testing there. It includes other ways of interaction with the device, such as downloading a database, and information about system resets.
- [PalmPilot Console Commands](#) provides descriptions and syntax for PalmPilot-specific commands available in the Debugger Console window and a section on memory debugging.
- [Testing Palm OS Applications](#) helps you avoid the most commonly encountered problems by providing suggestions on areas for testing. It also briefly discusses the PalmPilot Compatibility program.

Conventions Used in This Guide

This guide uses the following typographical conventions:

This style...	Is used for...
<code>fixed width font</code>	Code elements such as function, structure, field, bitfield.
<u><code>fixed width underline</code></u>	Emphasis (for code elements).
bold	Emphasis (for other elements).
blue and underlined	Hot links.



Designing Palm OS Applications

This chapter helps you design an application that's fast, robust, and consistent with other applications on the device. An introduction to the application design process is provided in Chapter 1 of Part I of "Developing Palm OS Applications." This chapter's focus is how to

- Avoid potential problems.
- Make your application integrate well with others.
- Achieve the best performance possible.
- Localize with the minimum amount of work.

The information was collected from engineers, testers, and other experts who designed, developed, and tested the four applications shipped with the first Palm OS device. A section on localization helps those developers that intend to make their application available in other languages.

Note that paying attention to user interface guidelines and, if applicable, to localization guidelines early in your development cycle will save you time and trouble later.

The chapter discusses these topics:

- [Designing Your Application](#)
- [User Interface Guidelines](#)
- [Localization Guidelines](#)

Note: Be sure to read [Avoiding Potential Pitfalls](#) for information on the problems developers encounter most frequently.

Application Design Process

The table below provides an overview of the application design process. Each step in the process is discussed in more detail below.

Step	Task	Guidelines
1	Design preliminary user interface.	User Interface Guidelines (detail), Summary of UI Design Rules (overview)
2	Create UI prototype.	
3	Design program prototype.	Integrating Programs With the Palm OS Environment , Creating Easy-to-Use Applications
4	Perform preliminary user testing. If satisfactory, go to step 5. If not, go back to step 2.	Testing Palm OS Applications
5	Localize if desired.	Localization Guidelines
6	Fine-tune performance, prepare for QA testing.	Avoiding Potential Pitfalls , Achieving Optimum Performance , Working With Databases , Making your Application Run on Different Devices

Development Tip: If you want to delete only the Preferences resources on the device, you can go to the Memory application and delete the desired application there while holding the down arrow (scroll) button.

Designing Your Application

This section provides Palm OS application design guidelines. It discusses these topics:

- [Getting Started](#)
- [Setting Up a Preliminary User Interface](#)
- [Creating a User Interface Prototype](#)
- [Integrating Programs With the Palm OS Environment](#)
- [Avoiding Potential Pitfalls](#)
- [Achieving Optimum Performance](#)
- [Making your Application Run on Different Devices](#)
- [Working With Databases](#)

Getting Started

As you're starting to design your Palm OS application, keep in mind the differences between a hand-held device and a desktop computer. Note that these differences are not just arbitrary restrictions, but reflect what your users want to do with the device.

For a successful, easy-to-use application, follow these guidelines:

- Remember that the Palm OS device is meant for data viewing and gathering, not for data processing.
- Off-load all heavy-duty processing onto the desktop computer.
- Design for the small square screen. Strip all complexity from the user interface that you possibly can. Examine each feature on the desktop and ask yourself whether it's really needed, then try to implement it if it is.
- If you do decide to implement a feature, remember that edit-in-place and easy access by one tap can reduce the complexity of your user interface.

Setting Up a Preliminary User Interface

After you've decided how your application will appear on the Palm OS device, the next step is to create a preliminary user interface. All Palm OS applications are based on a set of resource templates included with your development environment. In Palm OS 2.0, you can quickly design your application's user interface using Constructor.

For detailed information, see the "CodeWarrior Constructor for Palm OS Manual" in the CodeWarrior Documentation folder. The "Palm OS Tutorial" in the Palm OS SDK folder steps you through creating an application using Constructor.

The Palm OS development environment provides a set of resource templates that application developers use to implement the buttons, dialogs, and other UI elements. The table on the next page maps user interface elements to resources. The ResEdit name is included for developers that intend to continue using that tool. It's not relevant for Constructor users.

All resources are discussed in detail in Chapter 3, "Palm OS User Interface Resources," of "Developing Palm OS Applications, Part I." Specific design recommendations for some of the elements are provided below under [Summary of UI Design Rules](#) and [Summary of UI Design Recommendations](#).

A detailed specification of all user interface guidelines is provided in [User Interface Guidelines](#).

UI Element and Functionality	Example	Resource(s)
Command button— Execute command.		Button (tBTN)
Push button (also called radio button)— Select a value		Push button (tPBN)
Hot text entry— Invoke dialog that changes text of the button.		Selector trigger (tSLT)
Increment arrow— Increment / decrement values, or scroll.		Button (tBTN) or repeating button (tREP)
Check box— Toggle on or off.		Checkbox (tCBX)
Popup list— Choose a setting from a list.		Popup trigger (tPUT) Popup list (tPUL) List (tLST)
Menu— Execute commands not found on screen as buttons and so on.		Menu Bar (MBAR) Menu (MENU)
Text field— Display text (single or multiple lines).		Field (tFLD)
Scrollbar— Use together with fields or tables.		Scrollbar

Summary of UI Design Rules

Note: All developers are urged to include the first set of rules listed here in their test plan. Applications that don't follow these rules may cause problems for other applications on the device.

Here's a summary of rules you should follow to make your application consistent with other Palm OS applications:

- Whenever a field for user input is available, make sure that:
 - System keyboard is available via shortcut
 - System keyboard is available via menu
 - Graffiti input is possible (regular strokes and shortcuts)
 - Cut, copy, paste, and undo are possible
- Be sure to handle the clipboard correctly. If you use it, allow users to copy and paste between applications; if you don't, make sure it's intact when you exit.
- Don't gray out unusable menu items or other unusable UI elements. Remove them instead.
- Don't change or obscure the Graffiti status indicator area.
- Don't change or obscure the behavior of the silk-screened icons.
- Don't nest dialog boxes too deeply.
- Consider overloading the buttons. If you do overload, release the buttons at every possible opportunity. This is useful only for certain applications, such as games.

Summary of UI Design Recommendations

Here are some additional recommendations that make your application easier to use:

- If you can, provide command strokes for each menu command.
- If you can, allow finger navigation. For finger navigation, buttons need to be big enough for the system to recognize which button has been pushed. This is done by the Palm OS system software.

- Support Graffiti navigation:
 - Left-right-forward-backward movement as part of a field's behavior.
 - Getting to next and previous screen using the down/up and up/down keystrokes.
- Provide help dialogs (tips) for modal dialogs where possible.
- Include the Graffiti Help dialog using the function `SysGraffitiReferenceDialog`.
- Have the cursor ready and visible if there's only one field for text entry (saves one tap).

Creating a User Interface Prototype

In Palm OS 2.0, you can create a user interface prototype using the Constructor tool. This tool allows you to lay out the elements of a form using a drag and drop interface.

Because an easy to use and robust interface is so critical to the success of any application, you're strongly encouraged to create a prototype early on, then modify it as appropriate to suit your needs.

Integrating Programs With the Palm OS Environment

When users work with a Palm OS application, they expect to be able to switch to other applications, receive alarms, and so on. Your application will integrate well with others if you follow the guidelines in this section. Integrate with the system software as follows:

- Handle sysAppLaunchCmdNormalLaunch
- Handle or ignore other application launch codes as appropriate. A complete discussion of all launch code is provide in Chapter 2, *Control Flow*, of *Developing Palm OS Applications, Part I*.
- Handle system preferences properly. System preferences determine the display of
 - Date formats
 - Time formats
 - Number formats

Designing Palm OS Applications

Designing Your Application

- First day of week (Sunday or Monday)

Note that you have to handle system preferences even if you don't plan to localize your application.

- Allow the system to post these messages:
 - alarms
 - low-battery warnings
 - system messages during synchronization

In addition, follow these rules:

- Store state information in the application preferences database, not in the application record database. Call `PrfGetAppPreferences` and `PrfSetAppPreferences` to save and restore preferences. This is important if your application returns to the last displayed view by default.
- If your application uses the serial port, be sure to free the port when you no longer need it so that the HotSync application can use it.
- If your application supports private records, be sure they are unavailable to the global find when they should be hidden.
- The application name is defined in two ways:
 - The application name (required) is specified in the PilotRez panel of your CodeWarrior project and used by HotSync, the About box, the Memory display, and the database header.
 - The application icon name (optional) is a string resource in the application's resource file. It is used by the launcher screen and in the 2.0 Button Assignment preferences panel. You assign the name using the Constructor Project Settings panel.

Using the icon name is useful if you plan to localize your application.

Note: If you use an application icon name, make it short!

- Together with the application name, each application displays a application icon in the launcher. The size of this application icon should be 22 x 22 pixels. Each application should only have one application icon. It should be numbered 1000. You assign it in the Constructor Project Settings panel. Constructor assigns it type `tAIN`.

Avoiding Potential Pitfalls

Certain problems are encountered by application developers again and again. To avoid them, ask yourself these questions:

- Do you have a Creator ID for your application?
Each application (not just each company) has to have a Creator ID. Note that the Creator ID is only needed for the application (database of type APPL) not for all other databases. For more information, see www.usr.com/palm/crid.html
- Did you base your application on Phase 20 of the tutorial? That phase has the most extensive functionality and is therefore your best choice for starting your own application.
- Did you use C library calls in your application? If you did, change them to corresponding Palm OS calls. See *Developing Palm OS Applications*, Part I and Part II.

In addition follow these rules:

- Don't plan on using the extra 64K memory on the PalmPilot Professional for your application. This memory is reserved for the net library.
- Use only one event loop to simplify debugging.
- To conserve the battery, avoid continual polling. If your application is in a wait loop, poll at short intervals (for example, every tenth of a second) instead. The event loop of the Hardball example application included with your Palm OS SDK illustrates how to do this.
- Call `dmSetDatabaseInfo` when creating a database to assign a version number to your application. Databases default to version 0 if the version isn't explicitly set.
- Call `dmDatabaseInfo` to check the database version at application start-up.

Achieving Optimum Performance

Because the Palm OS device has limited heap space and storage, optimization is critical. The Palm OS device currently has no wait cursor, so users will expect rapid response.

To make your application as fast and efficient as possible, optimize for heap space first, speed second, code size third.

Follow these guidelines to optimize memory use:

- Allocate handles for your memory to avoid heap fragmentation.
- Sort on demand; don't keep different sort lists around. This makes your program simpler and requires less storage.
- Dynamic memory is a potential bottleneck. Don't put large structures on the stack.
- Arrange subroutines within the application to avoid 32K jumps .
- To have your application run well within the constraints of the limited dynamic heap (32K), follow these guidelines:
 - Allocate memory chunks instead of using global variables where possible.
 - Switch from one UI form to another instead of stacking up dialog boxes.
 - Edit database records in place; don't make extra copies on the dynamic heap.
- Avoid placing large amounts of data on the stack. Global variables are preferable to local variables (however, chunks are preferable to global variables). Your application only has 2K of stack space.

Making your Application Run on Different Devices

Starting in spring 1997, two different versions of Palm OS are available. Because the different versions of the OS provide different functionality, it's critical that you check the system version to be sure all functionality is included.

The Expense sample application in the `Examples` folder illustrates how to check the system version before starting the application.

Note: The Palm OS 1.0 PIM applications don't perform version checking because they are built into ROM. If you decide to compile one of their example application versions and download it onto a 2.0 device, it will crash.

Here is a summary of the appropriate calls to make to check the system version of an application:

```
FtrGet (sysFtrCreator, sysFtrNumROMVersion,
        &romVersion);
// romVersion is 0x02003000 for Personal or
// Professional
// romVersion is 0x01XXXXXX for 1000 and 5000.

FtrGet (netFtrCreator, netFtrNumVersion,
        &netVer);
// if this function returns ftrErrNoSuchFtr,
// there is no netlib available. Otherwise
// netVer will contain the version of netlib
// in the system. The current shipping
// version of NetLib is 0x01003000.
```

Designing Palm OS Applications

Designing Your Application

Version numbers have the format 0xMMmfsbbb, where:

- MM is major version
- m is minor version
- f is bug fix
- s is stage:
 - 3-release
 - 2-beta
 - 1-alpha
 - 0-development
- bbb is build number for non-releases

For example

V1.12b3 is 0x01122003

V2.00a2 is 0x02001002

V1.01 is 0x01013000

Making your application run on the different devices is discussed in some detail in Chapter 1, of *Developing Palm OS Applications, Part I*.

Working With Databases

Working properly with databases makes your application run faster and synchronize without problems. Follow these suggestions:

- When the user deletes a record, call `DmRemoveRecord` to remove all data from the record, not `DmDeleteRecord` to remove the record itself. That way, the Desktop application can retrieve the information that the record is deleted the next time there is a HotSync.

Note: If your application doesn't have an associated conduit, call `DmDeleteRecord` to completely remove the record.

- Keep data in database records compact. To avoid performance problems, Palm OS databases are not compressed, but all data are tightly packed. This pays off for storage and during HotSync.

- All records in a database should be of the same type and format. This is not a requirement, but is highly recommended to avoid processing overhead.
- Be sure your application modifies the flags in the database header appropriately when the user deletes or otherwise modifies information. This flag modification is only required if you're synchronizing with the PalmPilot PIM applications.
- Don't display deleted records.

User Interface Guidelines

The Palm OS device is designed for rapid entry and quick retrieval of information. To maximize performance, the UI should minimize navigation between windows, opening of dialog boxes, and so on. The layout of application screens needs to be simple so that the user can pick up the product and use it effectively after a short time. It's especially helpful if the UI of your application is consistent with other applications on the device so users work with familiar patterns.

This section helps you design a user interface that's intuitive, easy to use, and consistent with other applications on the device. You learn about these issues:

- [Understanding the Palm OS UI Design Philosophy](#)
- [Creating a Palm OS User Interface](#)
- [Creating Easy-to-Use Applications](#)

Note: Guidelines for implementing specific user-interface objects, such as information on the size of buttons or the font for labels, is provided in Chapter 3, "Palm OS User Interface Resources," of "Developing Palm OS Applications, Part I."

Understanding the Palm OS UI Design Philosophy

This section considers some issues that underlie the design of a user interface for the Palm OS device. It discusses these topics:

- [Creating Fast Applications](#)
- [Matching Use Frequency and Accessibility](#)

Creating Fast Applications

On a PC, users don't mind waiting a few seconds while an application loads because they plan to use the application for a certain amount of time.

The Palm OS paradigm, in contrast, resembles that of a watch: People want instant access to information. Speed is therefore a critical design objective for hand-held organizers and is not limited to execution speed of the code. The total time needed to navigate, select, and execute commands can have a big impact on overall efficiency.

The user should be able to keep up with someone on the telephone when setting up appointments, looking up phone numbers, and so on. Priorities include the ability to:

- Execute key commands quickly
- Navigate to key screens quickly
- Find key data quickly (for example, phone numbers)

Matching Use Frequency and Accessibility

PC user interfaces are typically designed to display commands as if they were used equally. In reality, some commands are used very frequently while most are used only rarely. Similarly, some settings are more likely to be used than others. For example, a 3 p.m.- 4 p.m. meeting occurs much more frequently than a 3:25 to 4:15 meeting.

More frequently used commands and settings should be easier to find and faster to execute.

- Frequently executed software commands should be accessible by one tap.
- Infrequently used commands may require more user action.

Frequency	Example	Accessibility
Several times per hour.	Checking today's schedule or to-do items.	One tap.
Several times per day.	One hour meeting starting at the top of the hour.	One tap, write in place.
Several times per week.	Setting a weekly meeting (repeating event).	Several taps, second dialog box.

To make your application easily accessible, follow these guidelines:

- Minimize the number of taps to execute a function or change a setting.
- Provide command buttons for commonly executed multistep operations. Command buttons streamline execution.
- Minimize the need to change screens.
- Minimize the number of dialogs users have to open and close.
- Avoid dialogs within dialogs unless it's an infrequently used feature.

Choose the appropriate UI object when making a speed versus screen layout decision:

- Buttons on the screen provide instant access but take up valuable screen space.
- Push buttons are faster than popup lists and should be used if they fit on the screen reasonably.
- Popup lists are faster than manual input or increment/decrement buttons
- Popup lists can be cumbersome if there are too many items on the list or if the list needs to scroll.

Creating a Palm OS User Interface

The small screen and pen-based user interaction require a different UI paradigm than a desktop computer. Here are some guidelines for making your application's interface consistent with other applications, including the PIM applications.

- Provide an application icon for the Launcher. To launch an application, users navigate to the launcher screen and tap on an icon. Choose a short icon name and an easy to recognize icon.

Specify the Application Icon Name and Application Icon using the Project Settings panel in Constructor.

- Provide a base screen that offers an overview of all available information. This screen is typically a list view. Not all applications need a base screen.
- Allow users to view most record information by pressing the navigation keys. Each event, to-do item, address, memo page, and so on is called a record.
- Organize records into user-defined categories if that makes sense. Categories usually result in more efficient screen use. Users can switch between categories using a popup menu or can display all records at once.
- Detailed information and advanced navigation require the use of a stylus. See [Data Entry Guidelines](#) for different data entry modes.
- Don't require double taps.
- Don't gray out menu commands or other UI elements; instead, remove an element when it's not available.

This section provides information on a variety of UI design issues:

- [Navigation Guidelines](#)
- [Preferences Guidelines](#)
- [Data Entry Guidelines](#)
- [Command Execution Guidelines](#)
- [Guidelines for Screen Layout](#)
- [Guidelines for Dialog Box Layout](#)

Navigation Guidelines

Users can move through applications by the following methods:

- **Switching applications.** Users press the physical buttons representing the PIM applications or access a launcher to switch applications.

On a Palm OS 2.0 devices, users can assign each button to the application of their choice using a Preferences panel.

When switching to an application, the user is either presented with a standard first screen or returned to the last place in that application.

- **Switching views.** Each PIM application has two or more views (or modes) typically
 - a list view (or view mode)
 - an edit view (or edit mode).

The user taps on records or uses command buttons to toggle between these views.

Edit mode gives users access to the Details button for settings that affect the entire record. They can also access specific menu commands for records. In many applications, tapping on a record switches the application to edit mode and displays an input cursor.

- **Switching categories of records.** A popup menu in the top right corner lets users switch between categories. The popup menu is found in the list view of applications that support categories.
- **Switching records in applications.** Depending on the application, the user can scroll through lists of records, then tap on a record or a Details button for further information.
- **Cycling through categories.** Holding the button on the hard case cycles through all categories.
- **Scrolling.** Records too long to display in one screen are scrollable. On-screen scroll buttons allow users to move up or down one line at a time. The physical arrow buttons allow users to move up and down one page at a time.

Scrollbars were introduced in 2.0. Scrollbars are optional. Developers have to consider the trade-off between taking up 7 pixels of horizontal space (the width of the scroll bar) vs. providing convenient scrolling for long lists of records.

Preferences Guidelines

Palm OS 2.0 has improved preferences facilities. They are available through launch codes, discussed in Chapter 2 of “Developing Palm OS Applications, Part I.”

The system now offers application-specific panels, sticky panels, and quick switch, as follows:

- **Application-specific panels.** Applications can add application-specific preferences panels to follow the system panels when the user cycles through the preferences. To do so, use the common code provided in `CW PalmPilot SDK:Examples:Formats` to make the pull-down menu available. If the application uses the common code, a Done button inserts itself if the panel was called from the application, not sequentially following another panel.
- **Sticky panels.** When users bring up a preference panel from the launcher, exit the panel, then bring it up again, the system returns to the last panel used.
- **Quick switch.** Applications can now use the launch codes `sysAppLaunchCmdPanelCalledFromApp` and `sysAppLaunchCmdReturnFromPanel`, which allow an application to let users change preferences without first selecting the launcher, then selecting the application again.

Data Entry Guidelines

Users can enter data by the following methods:

- **Graffiti.** Graffiti characters are written in the text area on the digitizer and appear on the screen at the cursor location. The user specifies the cursor location by tapping directly on the screen with the stylus.

Some controls accept input from Graffiti: For example, in the time selector dialog, you can write the time into the Graffiti area and it appears as start time or end time. The “next field” stroke switches between start and end time. The “Return” stroke dismisses the dialog.

For 2.0 applications, users expect that your application includes the Graffiti Reference option. You can include this option by calling `SysGraffitiReferenceDialog`. See Tutorial Phase 4.

- **On-screen keyboard.** In place of using Graffiti, the user can tap an on-screen keyboard with the stylus. Any text is entered into a temporary window. When the user dismisses the keyboard, the system inserts that text at the cursor location.
- **Controls.** Buttons, check boxes, and popup lists provide a quick way to enter settings and select options.
- **HotSync.** The user can type data on the PC and download it to the Palm OS device.
- **Auto-creation.** Many applications, such as the DateBook or the Memo Pad provide an auto-create feature. If the user starts to write in a list view with no record selected, a new record is created with no additional interaction.

To provide a consistent interface, follow these guidelines when designing the data entry interface for your application:

- Let users perform basic data entry in place.
- Provide a Details dialog for more elaborate data entry.
- Use the following format in the Details dialog:

Item (right-justified): Value(left-justified)

for example:

Set Date: 4-1-96

Auto-off after: 2 minutes

- Provide only one interface per function, that is, allow users to interact with an application through either a button, menu, or popup list. Don't provide both a button and a menu for the same actions.

Command Execution Guidelines

Users can execute commands by the following methods:

- **Command buttons.** Users execute common commands by tapping on command buttons at the bottom of the screen.
- **Menus.** Commands not represented by command buttons can be accessed via a simple menu system. The user taps on a menu hard icon in the digitizer area to invoke a menu bar. Provide menu shortcuts if possible.

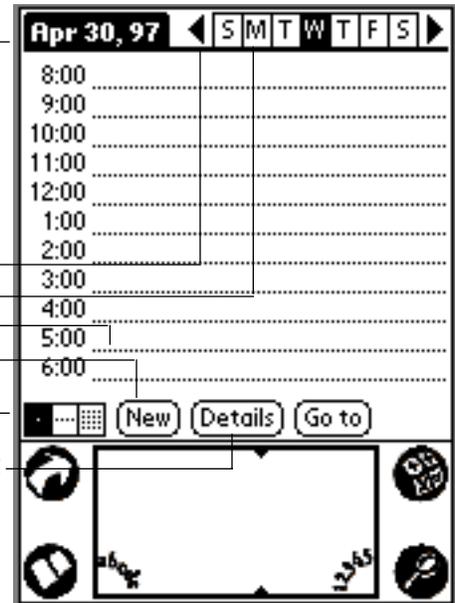
Note: If you provide shortcuts, make sure that each shortcut is unique among all commands available at that time.

- **Graffiti menu command shortcuts.** Users can write a special Graffiti stroke and a command keystroke to execute a menu command. This is analogous to keyboard shortcuts on a personal computer. For example, writing the command stroke symbol (a bottom-left to top-right line) and “C” allows the user to copy the selected text.

Guidelines for Screen Layout

The illustration below provides some interface guidelines. Each guideline is numbered and explained in more detail below.

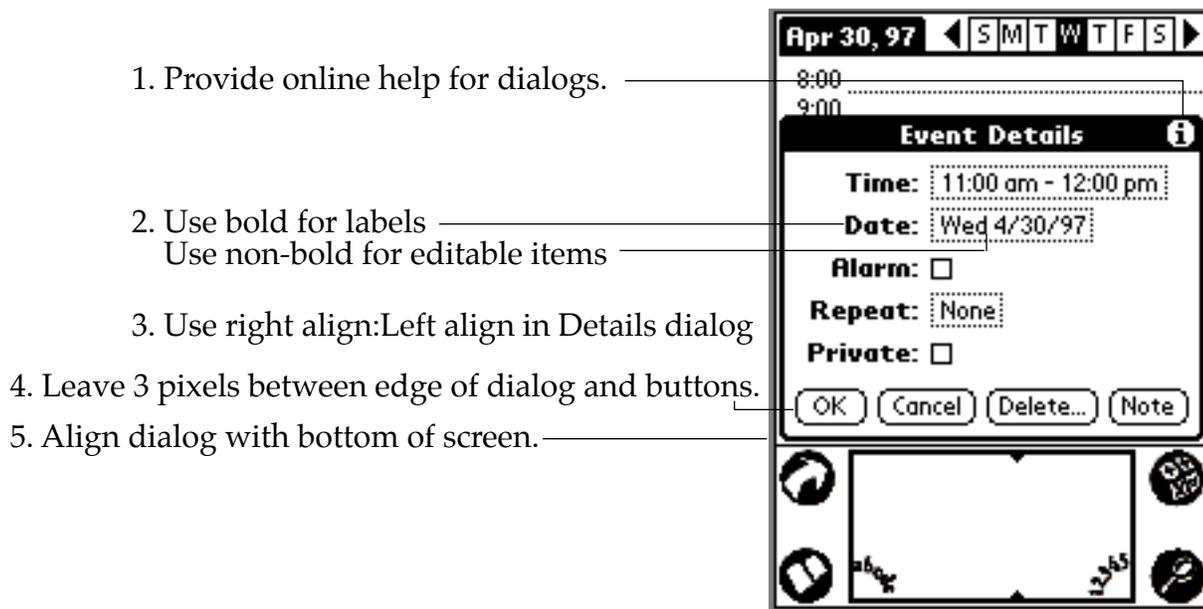
1. Provide a title bar
2. Go to the edge of the screen
3. Use resources provided with environment.
This example uses
 - repeating buttons
 - push buttons
 - fields
 - buttons
4. Align buttons at the bottom of the screen.
5. Leave one pixel above and below font height



1. In the title bar for each screen, provide both the application name and the name of the screen, if possible. Otherwise, provide the most relevant information.
2. Always go to the edge of the screen; that is, don't use borders. This practice maximizes screen real estate available to the application. The non-active area of the LCD and the case provide a natural margin.
3. Use the resources provided with the development environment and use the recommended values for width, height, and so on, provided in Chapter 3, "User Interface Resources" of "Developing Palm OS Applications, Part I."
4. Align buttons with the bottom edge of the screen.
5. For text surrounded by borders, leave one pixel above and below the font height.
6. For controls that can be displayed in groups, have at least two pixels to the left and right of the text label. The exception is command buttons, which require wider margins to accommodate the rounded border.

Guidelines for Dialog Box Layout

The illustration below provides some guidelines for dialog box interfaces. Each guideline is numbered and explained in more detail below under the same number.



1. Provide online help for dialogs.
2. Use bold for labels
Use non-bold for editable items
3. Use right align:Left align in Details dialog
4. Leave 3 pixels between edge of dialog and buttons.
5. Align dialog with bottom of screen.

1. Provide online help for dialogs. If you associate a Help ID with a form in Constructor, the system will add the “i” icon and handle presentation of the dialog.
2. Use bold face for labels, nonbold for editable items.
3. In the details dialog, right-align the label and left align the editable field.
4. When using buttons in dialogs, leave a space of 3 pixels between the edge of the dialog and the buttons.
5. Align dialogs, with the bottom of the screen. Leave the screen title bar visible if possible.

Creating Easy-to-Use Applications

Users must be able to pick up a PalmPilot device and, with no training or instruction, navigate between applications (without getting stuck) and execute basic commands within five minutes. Advanced commands should be easily accessible but should not be in the way.

The design must therefore fit the following criteria:

- Indicate clearly where in an application the user is. The PIM applications and modal dialog boxes have black title bars that usually indicate the application name and view.
- Make it obvious to the user how to get to different views. The command buttons provide the best example of achieving this.
- Use buttons for important commands.
- Accomplishing common tasks should be fast and easy. Minimizing steps helps not only speed but ease of use.

Ease of use amounts to a series of trade-offs. Striking the best balance for the most people is the biggest challenge of UI design. For example:

- Consistency reduces the time needed to learn an application by limiting the number of things that people need to keep in their head at once. The user should not have to memorize an entire set of rules to use the device easily, for example, the up arrow key should not do different things on different screens.
- Choose the number of buttons on the screen diligently:
 - The fewer buttons on the screen, the less time it takes to learn how to use the product.
 - However, keeping a few frequently used buttons on screen helps reduce the time spent learning basic functionality.
- Advanced features should not be in the way for beginners, but should not require multiple-step searching.
- If possible, make your application consistent with the Palm OS device's native applications; users are used to interacting with them and will easily get used to your application if you follow these rules.

Palm OS Resource Selection: List or Table?

Many developers find it difficult to decide whether to choose a list or a table for certain components of their application. The decision depends on a variety of factors, addressed in the next two sections.

List Resource

A list resource is meant for static data. Users can choose from a pre-determined number of items. Examples include:

- The time list in the time edit window of the datebook
- The Category pull-down.

While a list may have arrows for scrolling on screen, it doesn't automatically scroll when the user pushes the hard scroll buttons on the device itself. Applications have to add that behavior to the event handling they provide.

An application can use a list in two ways:

- Initialize a structure with all data for all entries in the list and let the list manage its own data.
- Provide list drawing functions but don't keep any data in memory. The list picks up the data as it's drawing.

Not keeping data in memory avoids unacceptable memory overhead if the list is large and the contents of the list depends on choices made by the user. An example would be a time conversion application that provides a list of clock times for a number of cities based on a city the user selects. Note that only lists but not tables can pick up the display information on the fly like this.

Formatting can be an issue for lists: While it's possible to imitate a multi-column display, lists really consist of rows of text and automatic column wrapping is not part of the capabilities of the list resource.

Tables

Tables support multi-column displays and generally handle them well. Examples are

- the List view of the ToDo application
- the Day view in the Datebook

You can attach scrollbars to the fields of the table. Just like lists, tables tell the application when the user selects an item.

A problem may arise if non-text elements are used in the table. For example, assume you have a table with two columns. In the first column is an icon that displays information, the second column is a text column. The table only allows users to select elements in the first column that are as high as one row of text. If the icon is larger, only a narrow strip at the top of the column can be selected.

Summary

Use tables when you need quality text handling (including editing in place). Be careful if you work with non-text items in some of the columns, the selection region may be smaller than you need.

Use lists when users select from a predefined list (e.g. categories) or if the application determines the information to be displayed on the fly (based on previous user selections). Remember that you are responsible for scroll button handling and that editing can be non-trivial.

Localization Guidelines

If you're planning to localize the Palm OS software you're developing, start by looking at the localized versions of the four PIM applications on the device. Then plan your application's interface, keeping in mind localization issues. This section first discusses [General Localization Guidelines](#), then some specific [Palm OS Localization Guidelines](#).

General Localization Guidelines

When you start planning for the localized version of your application, follow these guidelines:

- Don't use the English language version of the software as a guide when designing the layout of the screen.
Try to allow
 - extra space for strings
 - larger dialogues than the English version requires
- Abbreviations may be the best way to accommodate the particularly scarce screen real estate on the Palm OS device.
- Don't put strings in code. Separate any language-dependent strings from the code itself. If you have to print directly to the screen, remember that a one-line warning or message in one language may need more than one line in another language.
- Using a fine granularity is usually helpful. You can then concatenate strings as needed (and in the order needed) to arrive at a correct translation.
- Consider using string templates. For example, the MemoPad application uses the template: Memo # of %. The application can replace # and % to change the text.

Palm OS Localization Guidelines

Pay special attention to the following issues when localizing a Palm OS application.

- **Preferences.** Your application needs to check and use the settings users have chosen in the system preferences. The default preferences at startup are different for the different localized versions, though they can be overridden. The following settings are available:
 - Date formats
 - Time formats
 - Start day of week
 - Number formats

Note that there is no currency setting. There are also no separate settings for long and short dates; the long date format is mapped to the short date format.

- Capital letters with accent marks may be the same size as the letters without (in effect, the letter itself becomes one pixel shorter).
- Remember that most resources, for example, lists, fields, and tips, scroll if you need more space.
- Palm OS 2.0 has several new functions that facilitate localization: `LocGetNumberSeparators` lets you extract the thousandth and decimal separator. `DeLocalizeNumber` and `LocalizeNumber` lets you convert from the default US format to a specified country format (see `Preferences.h`).



Debugging in Standalone Mode on the Device

The CodeWarrior Debugger is used to download your application to the device. You can then use the Debugger running on the PC to run or step through the application on the device.

The Palm OS Debugger was used in earlier releases for downloading the application to the device and for related activities. You can still use it for those cases, as described in this chapter, but should use the CodeWarrior Debugger as your main debugging tool.

Note: Use the “Targeting Palm OS” manual in the CodeWarrior Documentation folder for detailed information about the Debugger.

This chapter first steps you through downloading and debugging an application with the CodeWarrior Debugger, then briefly discusses using the Palm OS Debugger for downloading. The chapter discusses these topics:

- [Debugging on the Device with CodeWarrior Debugger](#)
- [Resetting the Device](#)
- [Copying a Database From the Device to the PC](#)

Debugging on the Device with CodeWarrior Debugger

While you can test the application on the device by downloading it with the PalmPilot debugger, you can't debug it there. To do that, you have to use the CodeWarrior Debugger which lets you download and debug applications. This process is described in detail in the tutorial and summarized here.

Debugging in Standalone Mode on the Device

Debugging on the Device with CodeWarrior Debugger

To download an application for use with the CodeWarrior Debugger, follow these steps:

1. **Make sure that the cradle is correctly connected with the PC and the device is seated properly in the cradle.**
2. **Open your project in CodeWarrior.**
3. **Make sure the debugger is enabled.**
4. **Choose Project > Remove Object Code and Project > Reset File Paths to start with a clean project.**
5. **Press F7 to build (make) the executable.**

CodeWarrior creates a new .prc file and also a new .prc.psym file.

6. **Double-click the newly created .psym file.**

CodeWarrior prompts you to enter the “shortcut .2” sequence on the device.

7. **Turn on the device and enter the sequence: \mathcal{Q} .. 2**
8. **In CodeWarrior, choose Run.**

The Debugger appears, ready for debugging your application on the device.

There are two things you should note when debugging with the CodeWarrior Debugger:

- If you start the debugging session with the PalmPilot device in the Memory application, you have to tap the screen once after executing the “shortcut .2” sequence before you can choose run the Debugger.
- At the end of the debugging session, quit the debugger via the normal means if possible. This allows the debugger to clean up on the device and reset the device. If you don't quit normally, two potential problems may result:
 - The device's serial port remains open. This uses up batteries and prevents HotSync from working.

Any breakpoint code that the debugger inserted into the target application will still be there. This can only be cleaned up by exiting the debugger gracefully (File > Exit) or deleting the application.

Resetting the Device

As you debug your application, you may have to reset your device.

Any reset is normally performed by sticking a bent-open paper clip or a large embroidery needle into the small hole in the back of the PalmPilot device. This hole, known as the “reset switch” is in the middle of the serial number sticker. Depending on additional keys held down, the reset behavior varies. You can choose between:

- [Soft Reset](#)
- [Soft Reset + Up Arrow](#)
- [Hard Reset](#)

Soft Reset

A soft reset clears all of the dynamic heap (Heap 0, Card 0). The storage heaps remain untouched. The operating system restarts from scratch with a new stack, new global variables, restarted drivers, and a reset comm port. All applications on the device receive a `SysAppLaunchCmdReset` message.

Soft Reset + Up Arrow

Holding the up-arrow down while pressing the reset switch with a paper clip causes the same soft reset logic with the following two exceptions:

- The `SysAppLaunchCmdReset` message is not sent to applications. This is useful if there is an application on the device that crashes upon receiving this message (not uncommon) and therefore prevents the system from booting.
- The OS won't load any system patches during startup. This is useful if you have to delete or replace a system patch database. If the system patches are loaded and therefore open, they cannot be replaced or deleted from the system.

Hard Reset

A hard reset is performed by pressing the reset switch with a paper clip while holding down the power key. This has all the effects of the soft reset. In addition, the storage heaps are erased. As a result, all programs, data, patches, user information, etc. are lost. A confirmation message is displayed asking the user to confirm the deletion of all data.

The `SysAppLaunchCmdReset` message is sent to the application at this time. If the user selected the “Delete all data” option, the digitizer calibration screen comes up first. The default databases for the four main applications is copied out of the ROM.

If you hold down the up arrow key when the “Delete all data” message is displayed, and then press the other four application buttons while still holding the up arrow key, the system is booted without reading the default databases for the four main applications out of ROM.

Copying a Database From the Device to the PC

At times, it's useful to investigate a device database in some detail by using the full suite of debugging tools.

To copy either an application database or a data database from the Palm OS device to the PC, follow these steps:

1. **In the Console window of the PalmPilot Debugger, enter the command**

```
dir 0
```

The console window retrieves and prints a list of databases on the device. You can verify the name of the database you wish to copy.

2. **In the Console window of the PalmPilot Debugger, type**

```
export 0 dbname
```

and press Enter.

dbname is the case-sensitive name of the database you wish to copy

- **Don't** confuse the application with the database.
- If dbname has one or more spaces in it, enclose it in double quotes.

In the Console window, you should see output like the following:

```
Exporting record #0....  
Exporting record #1....  
Success!!
```

The database has been copied into the Debugger folder in the SDK folder under the name it had on the device.

Debugging in Standalone Mode on the Device

Copying a Database From the Device to the PC



PalmPilot Console Commands

This chapter provides descriptions and syntax for PalmPilot-specific commands available in the Debugger Console window. It presents the commands in the order they are listed in response to the Help command:

- [System Commands](#)
- [Card Info Commands](#)
- [Heap Utility Commands](#)
- [Chunk Utilities](#)
- [Database Utilities](#)
- [Miscellaneous Utilities](#)
- [Record Utilities](#)
- [Resource Utilities](#)
- [Debugging Utilities](#)

Note: Console commands are not case sensitive!

This is followed by a section on [Debugging Memory Management](#).

PalmPilot Console Commands

System Commands

Command	Result	Syntax
Feature	Display, get, register, or unregister feature(s).	<code>feature [options...]</code> <code>-all</code> Display a list of all known features <code>-unreg creator num</code> Unregister feature (only for RAM features) <code>-get creator num</code> Get a feature <code>-set creator num value</code> Set a feature
Kinfo	Get kernel info.	<code>kinfo [options...]</code> <code>-sem <id> all</code> Get semaphore info <code>-tmr <id> all</code> Get timer info

Card Info Commands

Command	Result	Syntax
card-format	Format a memory card.	<code>cardformat <cardNo> <cardName></code> <code><manufName> <ramStoreName></code> <i>Note:</i> <code>cardNo</code> , <code>manufName</code> , and <code>ramStoreName</code> are strings.
cardinfo	Get info on a memory card.	<code>cardinfo <cardNo></code>
storeinfo	Get info on a memory store.	<code>storeinfo <cardNo></code>

Heap Utility Commands

Command	Result	Syntax
HL	List all the heaps on a memory card.	hl <cardNo>
HI	Initialize a memory heap.	hi <hex heapID>
HD	Do a heap dump.	hd <hex heapID>
HT	Do a heap total.	ht <hex heapID>
HC	Compact a memory heap.	hc <hex heapID>
HChk	Do a heap check.	hchk <hex heapID>
HS	Do a heap scramble.	hs <hex heapID>
HF	Allocate all free bytes, except specified number of bytes.	hf <hex heapID> [free bytes]

Note: For more information, see [Debugging Memory Management](#).

Chunk Utilities

Cmd	Result	Syntax
New	Allocate a new chunk in a heap.	new <hex heapID> <hex chunkSize> [options..] -n Nonmovable -c Fill contents -o <ownerID> Set owner (0-14) -near <ptr> In same heap as <ptr> so <heapID> is ignored -lock Pre-lock
Free	Dispose of a chunk.	free <hex chunk ptr/ID> ^a [options..] -card <cardNo> Card number when local ID specified instead of chunk ptr

PalmPilot Console Commands

Cmd	Result	Syntax
Lock	Lock a chunk.	<code>lock <hex chunk ptr/ID>^a [options..]</code> <code>-card <cardNo></code> Card number when local ID specified instead of chunk ptr
Unlock	Unlock a chunk.	<code>unlock <hex chunk ptr/ID>^a [options..]</code> <code>-card <cardNo></code> Card number when local ID specified instead of chunk ptr
Info	Get info on a chunk.	<code>info <hex chunk ptr/ID>^a [options..]</code> <code>-card <cardNo></code> Card number when local ID specified instead of chunk ptr
Resize	Resize an existing chunk.	<code>resize <hex chunk ptr/ID>^a <hex newSize> [options..]</code> <code>-c</code> Check and fill contents <code>-card <cardNo></code> Card number when local ID specified instead of chunk ptr
Set-Owner	Set the owner of a chunk.	<code>setowner <hex chunk ptr/ID> <owner> [options..]</code> <code>-card <cardNo></code> Card number when local ID specified instead of chunk ptr

a. The hex chunk ptr/ID is the value in the Start column of a heap dump (HD).

Database Utilities

Cmnd	Result	Syntax
Import	Import a database.	<code>import [-r] <cardNo> <filename></code> <code>-r</code> Import from resource file <code>-u</code> Create new unique IDs <code>-d</code> Update creation and modification dates Note: This command is not case sensitive.
Export	Export a database.	<code>export [-m] <cardNo> <filename></code> <code>-m</code> Export as resource file

PalmPilot Console Commands

Cmnd	Result	Syntax
Create	Create a database.	<code>create <cardNo> <name> [options...]</code> -t <type> Database type, 4 characters -c <creator> Database creator, 4 characters -v <version> Database version -r Resource database
Del	Delete a database.	<code>del <cardNo> <databaseName></code> Note: This command is case sensitive.
Dir	List database directory.	<code>dir <cardNo> <searchOptions> [displayOptions]</code> <searchOptions>: -t Search by type -c Search by creator -latest Search for only latest version of each database <displayOptions>: -a Show all info for each database -n Show name -id Show chunk ID -s Show size -r Show number of records -at Show attributes -v Show version -d Show dates -m Show modification -i Show info fields -tc Show type and creator Note: Follow any option with - to turn it off.
Open	Open a database.	<code>open <cardNo> <name> [options...]</code> -r Open read-only -p Leave open
Close	Close a database.	<code>close <access ptr></code>

PalmPilot Console Commands

Cmnd	Result	Syntax
Opened	List all open databases.	opened
Set-Info	Set info in a database.	setinfo <cardNo> <dbName> [options...] <options>: -v <version> Set version -m <modification #> Set modification number -n <name> Set name

Miscellaneous Utilities

Command	Result	Syntax
SimSync	Simulate a sync on a database. Deleted records are removed and dirty flags are cleared.	simsync [access ptr] ^a
SysAlarmDump	Display the alarm table	sysalarmdump

a. Use the opened command to find the access ptr for a database.

Record Utilities

Command	Result	Syntax
Attach-Record	Attach a record to a database.	attachrecord <access ptr> ^a <record handle> <index> [options...] -r Replace existing record
Detach-Record	Detach a record from a database.	detachrecord <access ptr> ^a <index>
AddRecord	Add a record to a database.	addrecord <access ptr> ^a <index> <record text>
DelRecord	Delete a record from a database.	delrecord <access ptr> ^a <index>
Change-Record	Replace record in a database.	changerecord <access ptr> ^a <index> <record text>
List-Records	List records in a database. Display flags representing record attributes. From left to right, flags stand for <ul style="list-style-type: none"> • deleted • dirty • busy • secret followed by a category value.	listrecords <access ptr> ^a
Set-RecordInfo	Set info on a database record.	setrecordinfo <access ptr> ^a index [options...] -a <hex attr> Set attributes -u <uniqueID> Set unique ID

PalmPilot Console Commands

Command	Result	Syntax
MoveRecord	Move a record from one index to another.	<code>moverecord <access ptr>^a <from> <to></code>
FindRecord	Find a record by unique ID.	<code>findrecord <access ptr>^a <id></code>

a. Use the opened command to find the access ptr for a database.

Resource Utilities

Command	Result	Syntax
GetResource	Get a resource.	<code>getresource -t <type> -id <id></code>
List-Resources	List resources in a database.	<code>list resources <access ptr></code>
SetResource-Info	Set resource info.	<code>setresourceinfo <access ptr> <index> [options...] -t <resType> Set resource type -id <resID> Set resource ID</code>
AddResource	Add a resource to a database.	<code>addresource <access ptr> -t <type> -id <id> <record text></code>
DelResource	Delete a resource from a database.	<code>delresource <access ptr> <index></code>
Change-Resource	Change a resource in a database.	<code>changeresource <access ptr> <index> <resource text></code>

Debugging Utilities

Command	Result	Syntax
DM	Display memory.	dm <addr> [<count>]
SB	Set byte	sb <addr> <value>
MDebug	Set memory manager debug mode. Sets the level of memory checking used whenever a memory manager API function is executed. This command is helpful in tracking memory corruption bugs.	mdebug [options..] Shortcuts: -full Full checking (slowest) -partial Partial checking (faster) -off No checking (fastest) Fine tuning (Which heaps are checked/scrambled): -a Check/scramble all heaps each time -a- Check/scramble affected heap only Heap checking: -c Check heap(s) on some Mem calls -ca Check heap(s) on every Mem call -c- Turn off heap checking Heap scrambling: -s Scramble heap(s) on some Mem calls -sa Scramble heap(s) every Mem call -s- Turn off heap scramble Free chunk checking: -f Check free chunk contents -f- Don't check free chunk contents Minimum dynamic heap free space recording: (recorded in the global GMemMinDynHeapFree) -min Record minimum free space in dynamic heap -min- Don't record minimum free space

Debugging Memory Management

This section looks in some detail at debugging memory management. You can make sure that your application manages memory correctly by using commands that can be executed from the Palm OS Debugger Console window or from the CodeWarrior Debugger Console window.

The memory management verification includes making sure your application isn't

- Corrupting memory
- Causing unnecessary heap fragmentation
- Causing leaks by leaving unused memory chunks around

This section assumes that you understand how memory is structured in the Palm OS environment. In particular, you need to know:

- How memory is subdivided into heaps
- The different types of memory chunks (movable and nonmovable) in heaps
- How the data manager stores information

“Palm OS Memory Management” in “Developing Palm OS 2.0 Applications, Part III” describes this in detail.

Two classes of memory management debugging commands are available. All commands are discussed in the next two sections:

- [Displaying Memory Information](#)
- [Manipulating Memory](#) (commands that actively manipulate memory or put the memory manager into different modes)

Displaying Memory Information

A number of commands are available for displaying how memory is allocated. At the highest level, there are commands for displaying the available heaps and summary information about each heap. At the lowest level, there are commands for displaying how each heap is subdivided and for getting information on each chunk in a heap.

This section discusses the following commands, which are presented in alphabetical order:

- [CardInfo](#)
- [Chunk Info](#)
- [Heap Check](#)
- [Heap Dump](#)
- [Heap List](#)
- [Heap Total](#)

CardInfo

Syntax `cardinfo <cardno>`

Description Displays the total amount of RAM and ROM on a memory card. Also displays the total amount of free space in RAM and the total number of heaps. This command is especially helpful when you want a snapshot overview of the memory situation. The current Palm OS device has only one memory card slot, so the `<cardno>` argument has to be 0 (for the first memory card).

Example `cardinfo 0`
Name: Card 0

PalmPilot Console Commands

Displaying Memory Information

```
Manuf: Palm Computing
Version: 0001
CreationDate: 12345678
ROM Size: 00000000
RAM Size: 0003811A
Free Bytes: 000315F6
Number of heaps: #4
```

Chunk Info

Syntax Info -card <cardNo> <localID>

Description Most useful for converting a local ID to a chunk pointer or handle. Given a card number and local ID (in hex), returns information about the chunk that the local ID refers to. This includes a pointer to the start of the chunk data, the handle of the chunk if it's a movable chunk, the flags, size, owner ID, lock count, and the heap ID of the heap in which the chunk resides.

Example

```
info -card 0 8123
Info on Chunk at: 00B2B56C
Handle: 80B2B212
Flags: 0000
Size: 005A
Owner: #1
LockCount: #0
Heap ID: 0001
```

Heap Check

Syntax HChk <heapID>

Description Checks the integrity of a heap and displays an error message if it detects an error in the heap structure. The HD (Heap Dump) and HT (Heap Total) commands also check the integrity, but sometimes HChk is more convenient because it doesn't print extra information.

Example hchk 0
Heap OK

Heap Dump

Syntax HD <heapID>

Description Displays all the chunks in a heap. For each chunk, shows

- Start address
- Handle if it's a movable chunk
- Local ID
- Requested size
- Actual size,
- Lock count,
- Owner ID,
- Flags
- some identifying information about the chunk, including
 - Database it belongs to,
 - Resource type,
 - ID or record number.

If you dump the dynamic heap (heapID 0), the identifying information indicates whether or not the chunk is allocated by one of the system managers.

Example hd 1

Heap List

Syntax HL <cardno>

Description Displays a list of heaps on a memory card. For each heap, shows the heap ID, pointer, size, free bytes, biggest free chunk, and flags. The heap ID is important because it must be passed to the commands that display more detailed information about each heap.

PalmPilot Console Commands

Manipulating Memory

Example hl 0

Heap Total

Syntax HT <heapID>

Description Displays summary information about a heap. This information includes the total heap size; the number of master pointers available for movable chunks (`numHandles`); and the total number of free chunks, movable chunks, and nonmovable chunks currently allocated.

Example ht 1

Displaying Heap ID: 0001, mapped to 00B2B20A

Heap Summary:

flags:	0000	
size:	10000	
numHandles:	#200	
Free Chunks:	#1	(FC42 bytes)
Movable Chunks:	#3	(0094 bytes)
Non-Movable Chunks:	#0	(0000 bytes)

Manipulating Memory

This section describes the commands available for manipulating memory. For example, there are commands for filling a heap, compacting a heap, and scrambling a heap. These commands are all useful when you want to test the behavior of your application in various memory situations.

Another command, `MDebug`, puts the memory manager into different modes that are helpful for tracking down intermittent memory problems.

This section discusses the following commands, which are presented in alphabetical order:

Heap Compact

Syntax HC <heapID>

Description Forces a heap to be compacted.

This command essentially merges all the free chunks together. Normally, the memory manager compacts a heap only when a memory allocation fails.

Example hc 0
Heap Compacted

Heap Fill

Syntax HF <heapID> [freeBytes]

Description Fills a memory heap with memory chunks until there are only <freeBytes> free bytes.

This command is useful for testing the behavior of your application in low-memory situations. If <freeBytes> is not specified, then the entire heap is filled, with 0 free bytes remaining.

Example hf 1 100

Heap Scramble

Syntax: HS <heapID>

Description Scrambles a heap. As a result, any movable chunks in the heap are moved to another location in the heap.

Heap scrambling is useful for detecting situations where your application may have retained a pointer to an unlocked chunk. After a scramble, any pointers to unlocked chunks are invalid and will most likely cause your application to crash.

Example hs 1
Heap Scrambled

Memory Manager Debug Mode

Syntax: MDebug [options..]

Shortcuts:

- full : Full checking (slowest)
- partial : Partial checking (faster)
- off : No checking (fastest)

Fine Tuning:

Which heaps are checked/scrambled:

- a : check/scramble ALL heaps each time
- a- : check/scramble affected heap only

Heap Checking:

- c : check heap(s) on some Mem calls
- ca : check heap(s) on every Mem call
- c- : turn off heap checking

Heap Scrambling:

- s : scramble heap(s) on some Mem calls
- sa : scramble heap(s) every Mem call
- s- : turn off heap scramble

Free Chunk Checking:

- f : check free chunk contents
- f- : don't check free chunk contents

Min Dynamic Heap free space recording:
(Recorded in the global GMemMinDynHeapFree)

- min : record minimum free space in dynamic heap
- min- : don't record minimum free space

Description Puts the memory manager into the selected debug mode. Can put it into modes where heaps are automatically scrambled and checked after your application makes certain memory manager calls. Once you put the memory manager into one of these debug modes, your application is dropped into the Debugger as soon as a corrupted heap is detected.

PalmPilot Console Commands

Manipulating Memory

Most of the time, the `MDebug` command is used with the `-partial` argument. This argument puts the memory manager into a mode where on every memory manager call, the memory manager scrambles and checks the heap that memory manager call operates on.

A more stringent mode is entered with the `-full` argument: The memory manager checks every heap on every memory manager call. Performance degradation from this command is usually quite bad, so it's best not to use it until the last stages of verification.

The other options (`-a`, `-c`, `-s`, `-f`) can be used to fine-tune the debug mode. You'll notice that `-partial` and `-full` are merely shortcuts that set the other options for you. After setting the mode by using `-partial` or `-full`, you can further fine-tune debug mode by entering one or more of these other options.

The last option, `-min`, makes the memory manager record the minimum amount of free space ever detected in the dynamic heap. After setting this option, you can run your application for awhile and then enter the `MDebug` command with no options, to display the minimum amount of free space ever detected in the dynamic heap. Using `-min` is helpful in determining the "breathing room" in the dynamic heap.

Examples:

```
mdebug -partial
Current mode = 001A
  Only Affected heap checked/scrambled per call
  Heap(s) checked on EVERY Mem call
  Heap(s) scrambled on EVERY Mem call
  Free chunk contents filled & checked

  Minimum dynamic heap free space recording OFF

mdebug -full
Current mode = 003A
  Every heap checked/scrambled per call
  Heap(s) checked on EVERY Mem call
  Heap(s) scrambled on EVERY Mem call
```

Free chunk contents filled & checked

Minimum dynamic heap free space recording OFF

mdebug -min

Current mode = 007A

Every heap checked/scrambled per call

Heap(s) checked on EVERY Mem call

Heap(s) scrambled on EVERY Mem call

Free chunk contents filled & checked

Minimum dynamic heap free space recording on:

Current value: 0 bytes

mdebug

Current mode = 007A

Every heap checked/scrambled per call

Heap(s) checked on EVERY Mem call

Heap(s) scrambled on EVERY Mem call

Free chunk contents filled & checked

Minimum dynamic heap free space recording on:

Current value: 15918 bytes

mdebug -a-

Current mode = 005A

Only Affected heap checked/scrambled per call

Heap(s) checked on EVERY Mem call

Heap(s) scrambled on EVERY Mem call

Free chunk contents filled & checked

Minimum dynamic heap free space recording on:

Current value: 15918 bytes

PalmPilot Console Commands

Manipulating Memory



Testing Palm OS Applications

This chapter discusses testing Palm OS applications in some detail in the following sections:

- [The PalmPilot Compatibility Program](#)
- [Creator IDs](#)
- [Testing Application Integration](#)
- [Debugger Nub Signal](#)

The PalmPilot Compatibility Program

If you participate in the PalmPilot Innovator program, you can have your application tested as a PalmPilot compatible product and take advantage of certain US Robotics marketing activities. There are two levels, Gold Products compatibility and Platinum Products compatibility.

For both levels, applications are sent to a third-party testing lab.

- Gold products are tested successfully for Gold Level PalmPilot Compatibility. They may use the PalmPilot compatibility logo on product and packaging and qualify for a basic level of marketing support from U.S. Robotics.
- Platinum products are tested successfully for Platinum Level PalmPilot Compatibility. They may use the PalmPilot compatibility logo on product and packaging and qualify for participation in co-marketing activities with U.S. Robotics.

For more information about the criteria, see:

www.usr.com/palm/crid.html

Creator IDs

Each Palm OS application has a distinct creator ID. A creator ID is a 4-byte value used to tie together all the databases related to the application.

Creator IDs are unique to the application, not the creator of the application. Each database on the PalmPilot has an application value and a type. The type value should be set to `sysFileTApplication` for the executable's database and can be set to any value for other databases associated with an application.

Creator IDs need to be either all caps or mixed case. The Palm OS creator IDs differ from the creator ID and type that appear in the CodeWarrior 68K Project > Project Settings dialog boxes.

The creator ID for a Palm OS application is assigned in the PilotRez Project Settings panel.

- The Type should be set to `APPL`. Type is a 4-byte value.
- For information about creator IDs, see www.usr.com/palm/crid.html.

The system uses the Creator ID in various ways:

Creator ID and type is used by PalmPilot system launcher window to determine which databases are applications that should be displayed for selection.

The memory application uses a creator ID and type to determine names of applications for display and to calculate total memory used by an application.

Testing Application Integration

A Palm OS application should fit in with the environment in a way that users expect. For example, users need instant access to other applications such as the calculator; they expect to have access to Graffiti and the on-screen keyboard; and they expect to access information with the global find.

Here are some guidelines:

- Be sure your application does not obscure or change the Graffiti area, silk-screened buttons, and power button.
- Follow the guidelines listed in [User Interface Guidelines](#) and pay special attention to these points:
 - Test that the different user input modes (e.g., Graffiti and keyboard) are available for each field.
 - Test that menu items work with shortcuts as advertised.
 - Put limits on the length of fields and test them.
 - Test that any growable control, such as the launcher window or the menus, scrolls correctly.
- Be sure your application uses the System Preferences for numeric formats, date, time, and start day of week. Do this even if you're not planning on localizing your software.
- Test that your application properly handles system messages during and after synchronization.
- Test that deleted records are not displayed.
- Test that your application doesn't exceed the maximum number of categories: 15 categories and the obligatory category "Unfiled" for a total of 16.
- Test that your application properly handles the global find. Generally, searches and sorts aren't case sensitive.
- If your application allows for private records, test that they are hidden properly and that a global find ignores them.
- Don't obscure Graffiti shift indicators.
- Test that your application uses a consistent default state when the user enters it:
 - Some applications have a fixed default; for example, the Date Book always displays the current day when launched.
 - Other applications return to the place the user exited last. In that case, remember to provide a default if that place is no longer available. Because of HotSync and Preferences, don't assume the application data is the same as it was when the user looked at it last.

Testing Palm OS Applications

Debugger Nub Signal

- Test for performance. Launching, switching, and finding should be fast.
- If your application uses sounds, be sure it uses the Warning and Confirmation sounds properly.

Debugger Nub Signal

If the Palm OS device appears frozen and there is a very small square flashing in the upper left corner of the screen, the device has crashed and has activated debugger mode. At this point, you can:

- Reset the device OR
- Connect the device to the Desktop Debugger.

Index

Numerics

2.0

- button assignment 27
- localization 37
- preferences 28
- scroll bars 27

32K jumps 20

A

- AddRecord command 51
- AddResource command 52
- alarms 18
- allocating handles 20
- APPL database 18
- application design
 - accessibility 25
 - assigning version number 19
 - base tutorial phase 19
 - buttons 33
 - command buttons 25
 - continual polling 19
 - data entry 28
 - dialogs 25
 - ease of use 33
 - handling system messages 67
 - minimizing taps 25
 - removing deleted records 23
 - See Also* UI design
 - switching applications 27
 - using lists 34
- application icon 18, 26
 - size 18
- application icon name 18
- application name 18
- application preferences database 18
- application record database 18
- AttachRecord command 51

B

- Button resource 15, 25
- buttons
 - assignment by end-user 27
 - choosing number 33

- in dialog 32
- position 31
- traversing categories 27

C

- C library calls 19
- cardformat command 46
- cardinfo command 46, 55
- categories 26, 27
 - maximum number 67
 - traversing with button 27
- ChangeRecord command 51
- ChangeResource command 52
- Checkbox 15
- chunk info 56
- clipboard 16
- Close command 49
- command buttons 25
- command strokes 16
- compatibility logo 65
- compatibility program 65
- Console commands 45–??
- copying database from device 43
- Create command 49
- creator ID 18, 66

D

- data entry, Graffiti 28
- database version number 19
- databases
 - moving from device to desktop 43
- date format system preferences 17
- debugger nub signal 68
- debugging
 - memory management 54
 - memory manager (modes) 61
 - on device 39–??
 - with CodeWarrior debugger, CodwWarrior debugger 39
- Del command 49
- deleted records 23, 67
- deleting preferences 12

Index

- deleting records 22
- DelRecord command 51
- DelResource command 52
- design process 12
 - getting started 13
- desktop responsibilities 13
- DetachRecord command 51
- Details 27
- Details dialog
 - format 29
- device
 - debugger nub signal 68
 - resetting 41
- dialog boxes 20
- dialogs
 - design 32
 - online help 32
- Dir command 49
- dirty flag 22
- DM command 53
- dmDatabaseInfo 19
- DmDeleteRecord 22
- DmRemoveRecord 22
- dmSetDatabasInfo 19
- double taps 26
- drivers, restarting 41
- dynamic memory 20

E

- edit-in-place 20
- editable items
 - lables 32
- event loop
 - example program 19
 - one event loop 19
- Export command 48

F

- Feature command 46
- Field 15
- find, access to global find 66
- FindRecord command 52
- finger navigation 16
- first day of week 17

- font
 - lables 32
- Free command 47

G

- GetResource command 52
- global find 66, 67
 - and private records 18, 67
- global variables 20
- global variables,erasing 41
- Gold Level PalmPilot Compatibility 65
- Graffiti 16, 28
- Graffiti Help 17
- Graffiti navigation 17
- Graffiti reference 28
- Graffiti status indicator area
 - not obscuring 16
- graying-out UI elements 16
- GUI design
 - reducing complexity 13

H

- handles
 - allocation 20
- hard reset 42
- HC command 47, 59
- HChk command 47, 56
- HD command 47, 57
- heap check 56
- heap compact 59
- heap dump 57
- heap fill 60
- heap fragmentation 20
- heap list 57
- heap scramble 60
- heap space 20
- heap total 58
- help dialogs 17
- Help ID 32
- HF command 47, 60
- HI command 47
- HL command 47, 57
- HotSync 22
 - modifying flags 22

HS command 47, 60
HT command 47, 58

I

icons
 application icon 18
ID *See Also* creator ID
ID *See Also* creator ID
Import command 48
Info command 48, 56
Innovator program 65

K

keyboard 29
Kinfo command 46

L

labels, font 32
launch codes
 handling 17
launcher 26
 application icon name 18
List resource 15, 34
ListRecords command 51
ListResources command 52
localization
 general guidelines 36
 Palm OS guidelines 37
Lock command 48
logo
 compatibility logo 65
low-battery warnings 18

M

manipulating memory 58
MDebug command 53
MDebug command options 61
Memory application 12
memory management debugging 54
memory manager debug mode 61
memory, manipulating 58
Menu Bar resource 15
Menu Resource 15

menus 30
modal dialogs, help *See Also* Details dialog 17
modes 27
MoveRecord command 52

N

name *See* application name 18
navigation 27
New command 47
number format system preference 17

O

on-screen keyboard 29
online help 32
Open command 49
Opened command 50
optimization 20
 dynamic memory 20
 sorting 20
overloading buttons 16

P

PalmPilot Professional 19
 version checking 20
patches
 loading during reset 41
performance 20
physical scrolling 27
Platinum Level PalmPilot Compatibility 65
Popup list 15, 25
Popup trigger 15
preferences 28
 deleting 12
 quick switch 28
 restoring 18
 saving 18
PrfGetAppPreferences 18
PrfSetAppPreference 18
private records 18, 67
Push button 15, 25

Q

quick switch, preferences 28

Index

R

- Repeating button 15
- reset 41
 - hard reset 42
 - loading patches 41
 - soft reset 41
- Resize command 48
- restoring preferences 18

S

- saving preferences 18
- SB command 53
- screen layout 31
- scrolling 27
- secret records 67
- Selector trigger 15
- serial port 18
- SetInfo command 50
- SetOwner command 48
- SetRecordInfo command 51
- SetResourceInfo command 52
- shortcuts 30
- silk-screened icons, not obscuring 16
- SimSync command 50
- Simulator
 - 32K jumps 20
- Simulator Console commands 45–??
- soft reset 41
- sorting 20
- stack space 20
- state information, storing 18
- storage heaps, erasing 42
- storeinfo command 46
- switching applications 27
- switching categories 27
- switching views 27
- synchronization messages 18, 67
- synchronizing
 - modifying flags 22
- SysAlarmDump command 50
- sysAppLaunchCmdNormalLaunch 17
- sysAppLaunchCmdPanelCalledFromApp 28
- SysAppLaunchCmdReset 41
- sysAppLaunchCmdReturnFromPanel 28

- SysGraffitiReferenceDialog function 17
- system keyboard 16
- system messages 18, 67
- system preferences 17

T

- tAIN resource 18
- taps
 - double taps 26
 - minimizing 25
- testing
 - application integration 67
 - guidelines 67
- time format system preferences 17
- tips 17
- title bar 31
- tutorial (use for application design) 19

U

- UI design 23
 - avoiding dialog box stacking 20
 - button alignment 31
 - design elements 14
 - design philosophy 23
 - dialogs 32
 - screen layout 31
 - title bar 31
- UI design rules 16
 - clipboard 16
 - command strokes 16
 - finger navigation 16
 - Graffiti navigation 17
 - Graffiti status indicator area 16
 - graying-out UI elements 16
 - help dialogs 17
 - overloading buttons 16
 - ready cursor 17
 - silk-screened icons 16
- Unlock command 48
- user input 16
 - cut, copy, paste, undo 16

V

- version checking (device version) 20
- version number 19

W

wait cursor 20