About this Test

Palm OS® Release 5 is coming! As a developer, that means a new level of devices from licensees that incorporate ARM®-compatible chips. And that means more opportunities. To take advantages of those opportunities, you want to make sure your applications work on Palm OS 5 and the OS of today.

In order to make that easy for you, we have developed the following Palm OS 5 Compatibility Test Cases. It is designed to provide an overview of the new features of Palm OS 5 as well as a Palm OS 5 testing process for your application.

The goal of this test is to insure that an application works equally well on a 68K device running Palm OS 4 as it does on a ARM-based device running Palm OS 5. With this in mind the focus of the below test cases is to detect those compatibility issues relating specifically to the move to Palm OS 5. General compatibility issues and or non optimal use of the new feature set in Palm OS 5 may become evident during the testing cycle, but finding these more general errors is not the focus of the test. For this reason, you should be testing beyond the test cases outlined here. For a more general understanding of how to test your application for compatibility with the Palm OS see the last section in this document, General OS Compatibility.

The first step is to read up on Palm OS 5. There is a brief introduction in this document and more information can be found at http://www.palmos.com/dev/support/docs/palmos5/os5overview.html.

The second step is to download the latest version of the Palm OS 5 Simulator from the Developer Seeding Area. Note that in the below test cases, the term Simulator is used to refer to the Palm OS 5 Simulator. Please do not confuse this with the older Mac based Simulator or the Palm OS Emulator (POSE).

The third step is to go through the below test cases and test your application on the Simulator. Testing time will vary somewhat based on the size and type of application, but in general, the target is a 3 hour test. This is assuming that no bugs are found and does not include regression of found bugs.

Finally, the last step is to report your tested and, if need be, fixed application. This is a simple and short step that provides us with valuable information concerning the level of compatibility among existing 68K applications. Consider that taking the time to do this helps the Palm OS Platform, which in turn helps you as a developer of applications that run on the platform.

Palm OS 5 Overview

Palm OS 5 runs on ARM-compliant processors rather than the Motorola Dragonball 68K type processors that have been in all Palm Powered handhelds to date. In order for current 68K based applications to run as is on Palm OS 5, the Palm Application Compatibility Environment (PACE) was added to the system. Among other things, PACE translates 68K instructions to ARM, swaps the byte order of integers (integers are stored in opposite endianess on 68K and ARM), and adjusts for structure packing differences. This means that well behaved applications will continue to work completely oblivious to the fact that they are running in PACE.

In addition, there are also a number of new features and characteristics that can affect compatibility (Not all these features will be present on all Palm OS 5 devices. Licensee decide whether to include them)

- High-density support for 320x320 screens - Assumptions about screen characteristics can lead to unpredictable results. Applications that stick to current Window Manager APIs to draw to the screen will continue to work. Of course, getting and modifying a bitmap's bits without knowing if it is high resolution or not can lead to unpredictable results.
- UI Color Themes - Palm OS 5 includes a color theme picker that modifies the system palette. It is now more important to set the transparency on bitmaps and not assume any default color palette when choosing colors to use

in your application. An example might be assuming that red text will stand out on a while background only to have your application end up running in a color theme where the default background is red.
- Sampled Sound APIs - Devices that only have the new sound hardware will have to convert simple old style sounds into wave files in order to play them. This means that applications such as games which rely on heavy use of the old Sound Manager APIs may see decreased performance.
- Speed - Generally, the OS will be running much faster than on previous 68K-based handhelds. Developers may need to add appropriate delays to keep applications usable.

For more information about the Palm OS 5 feature set and characteristics please see the Palm OS 5 Development Overview at http://www.palmos.com/dev/support/docs/palmos5/os5overview.html

In testing third party applications against Palm OS 5 the most common errors we have found are:

- Direct access of UI object's data structures. Most of the time we have found that developers are getting or setting characteristics that are easily accessible through standard APIs or glue routines. For example:
  - rect from a field object
  - numItems of a list object
  - formID of a form
  - bounds of a control object
  - attr of a field object

  In Palm OS 5 you can not assume that a pointer to system owned object is pointing to the real object rather then a shadow structure of that object thus any attempted access or action that depends on this will fail.
- Memory read write errors such as accessing low memory, system globals, unallocated chunks, hardware registers, etc. Many of these errors are a result of reading or writing to an offset from a NULL pointer. However, accessing hardware registers and system globals is likely more deliberate and not supported moving to Palm OS 5.
- Attempting to draw before a valid draw window is set
- Incorrectly modifying the screen's bitmap's bits
- Relying on trap patching. Attempting to patch a trap will not cause any problems, but when running in Palm OS 5 it will simply return an error code. Your application needs to validate that your patch was successful and either work without the patch or exit gracefully.

Setting up the Palm OS Simulator for Testing

The Palm OS Simulator is the Palm OS 5 system compiled native for Intel. It is not hardware emulation like Palm OS Emulator, but rather the real OS running on top of the Device Abstraction Layer (DAL). The Palm OS core is made up of the various system DLLs you see when you open up the Simulator folder on your PC. As a development tool you use the Simulator much in the same way you would the Palm OS Emulator. You can drag and drop applications onto it for testing and debug to it from CodeWarrior or the Palm Debugger. For more detailed information on the Simulator and how to use it, see the *Testing with Palm OS Simulator* documentation and the *Palm OS Simulator FAQ*, both available via the developer website.

The test cases detailed below should all be done with the Simulator running an English high resolution color debug ROM. The default settings for all tests are:

1. Open up the NTFullDbg_enUS.rom in the enUS subdirectory
2. Right click on the Simulator an pop up "Settings">"Display" to choose display settings.
3. Make sure that the "Color depth" is set to "65536 colors"
4. "Allowed screen depths" are all check boxes checked
5. "Low Density Mode" is not checked
6. "Allow direct screen access" is not checked
7. Under "Settings">"Memory" both "Storage is write protected" and "PACE extended checks" are checked
8. "Settings">"Enable sound" is also be checked

Some of the tests specify different setting and under only those tests should this basic setup be altered.

Compatibility Test Cases

Verify Application Launch

These tests will verify that your application runs properly in response to a variety of different launch codes and launch circumstances.

Test normal launch and exit of application

1. Launch the application by tapping the application icon in the Launcher.
2. Exit the application by tapping the Launcher icon.
3. Re-launch the application by again tapping the application icon from within the Launcher.

The application should launch correctly. It should exit back to the Launcher error free and again launch correctly.

Launching via global find

If your application handles global find:

1. Add some text within the application to search for (for example "PalmSource")
   .
2. While still running the application, tap on the Find icon and search for that text.
3. Exit the application, and again tap on the Find icon and search for that text.

Under both circumstances the application should find the text searched for. In the first case, where you were already the active application, tapping on the found text from the Find dialog should return you to the application (preferably taking you to the searched for text). In the second case your application should be successfully launched (once again it is preferable that the application opens to show the searched for text).

If you do not handle global find, your application should not handle the global find's launch code (sysAppLaunchCmdFind). Verify this by:

1. From the Launcher, tap on the Find dialog and enter any text to search for.
2. Keep tapping "Find More" button until there are no more applications to search.

The global find dialog should work as expected. Your application should NOT draw to the find dialog if it has not searched your application.

Launching via expansion card

Even if you are not shipping your application on an expansion card, it is important to consider that users might add your application to a card with a card reader/writer. There are two launch codes sent to an application when launched via the card: sysAppLaunchCmdCardLaunch and sysAppLaunchCmdNormalLaunch. Even if you have not modified your application to support a card launch, the Launcher will still attempt to launch the application via sysAppLaunchCmdNormalLaunch.

1. Install your application on to the Simulator.
2. Tap the "Menu" icon and select "Copy..."
3. Copy your application from the device to the "RAMDisk" expansion card.

4. Return to the Launcher and tap the "Menu" icon and choose "Delete..."
5. Delete the application you are testing and any related libraries and PDBs.
6. Return to the Launcher and choose the "RAMDisk" from the Launcher categories.
7. Tap your application icon to launch your application
.
8. Tap the Launcher icon to return to the Launcher.

As with a normal launch, the application should launch and go to the opening screen of the application. It should exit back to the Launcher error free and again launch into the opening screen.

Auto-launching from the card

The Palm OS 5 Simulator does not currently provide the ability to mount and unmount the RAM disk card. The below mentioned VFSMakeStart.prc utility can be used to simulate unmounting and then mounting the "RAMdisk" expansion card. The important point of the test is not the vfsStart utility application, but rather the fact that it gets unmounted and mounted. The complete source to the project can be downloaded here and you can modify it to more suit your application if need be.

1. Download the vfsStart application from the Developer Seeding Area web page.
2. Install the VFSStart.prc from this project to your Simulator (Do not use HostFS at the same time as this application).
3. Install your application on to to the Simulator.
4. From within the Simulator start the VFSStart application.
5. Enter the Creator ID of your application.
6. Tap the "Make Start" button to make your application the start.prc on the "RAMdisk" card.
7. Return to the Launcher and tap the "Menu" icon and choose "Delete..."
8. Delete the application you are testing and any related libraries and PDBs.
9. Return to the Launcher and again start the VFSStart application.
10. Tap the "Remount" button to simulate a unmount and mount of the card.
11. Tap the Launcher icon to return to the Launcher.

As with a normal launch, the application should launch and go to the opening screen of the application. It should exit back to the Launcher error free and again launch into the opening screen.

Handling the Reset Launch code

1. Install the application on to the Simulator.
2. Right click on the Simulator and choose Reset>Soft to simulate a soft reset of the device.
3. Run the application once and exit.
4. Again, right click on the Simulator and choose Reset>Soft to simulate a soft reset of the device.

There should be no errors associated with your application and the reset should work correctly

Alarms

If your application handles alarms:

1. Set an alarm to occur in a minute or two.
2. Exit the application.
3. Wait for the alarm to go off.
4. If present, tap the "Go To" button from the Alarm dialog.

The alarm should be handled error free.

Additional Launch Codes

If your application handles any of the other launch codes defined in SystemMgr.h, you should test that the application successfully launches and exits via these launch codes. Possible examples are sysAppLaunchCmdAttention, sysAppLaunchCmdNotify, or sysAppLaunchCmdCustomBase.

Basic Navigation

Note that for the below tests switching color depth or display density will cause a system soft reset. The point of the tests is to confirm that the application will run under different display setting NOT that it can handle a change in display settings while running.

High Resolution, 16 bit color

1. Launch your application normally.
2. Traverse through the application's various forms, alerts, and modal dialogs using both menu and screen (button) navigation.

The application's various forms should all display properly, bitmaps and fonts should be readable. In particular, if you are doing any custom drawing (bitmaps, gadgets) watch for whether or not they are properly redrawn on a form after a menu or modal dialog has been dismissed. If you have no modal dialogs or menus, you may want to set an Alarm in the Calendar application and then launch into your application so you can see how it reacts when a dialog is dismissed.

High Resolution, 8 bit color

1. Right click on the "Settings">"Display">"Color depth" and choose "256 colors".
2. Again traverse through application's various forms, alerts, and modal dialogs using both menu and screen (button) navigation.

The application's various forms should all display properly, bitmaps and fonts should be readable. In particular, if you are doing any custom drawing (bitmaps, gadgets) watch for whether or not they are properly redrawn on a form after a menu or modal dialog has been dismissed. If you have no modal dialogs or menus, you may want to set an Alarm in the Calendar application and then launch into your application so you can see how it reacts when a dialog is dismissed.

Low Resolution, 8 bit color

1. Right click on the Simulator and choose "Settings">"Display">"Low Density Mode".
2. Again traverse through application's various forms, alerts, and modal dialogs using both menu and screen (button) navigation.

The application's various forms should all display properly, bitmaps and fonts should be readable. In particular, if you are doing any custom drawing (bitmaps, gadgets) watch for whether or not they are properly redrawn on a form after a menu or modal dialog has been dismissed. If you have no modal dialogs or menus, you may want to set an Alarm in the Calendar application and then launch into your application so you can see how it reacts when a dialog is dismissed.

High Resolution, 1 bit color

1. Reset the the device to High Density by right clicking on the Simulator and unchoose "Settings">"Display">"Low Density Mode".
2. Change the Simulator's bit depth to 1 bit black and white by right clicking on the Simulator and choosing "2 colors" from "Settings">"Display">"Color depth" and then unchecking all but the "1 bpp" check box from "Settings"> "Display">"Allowed Screen Depths...".

3. Again traverse through the application's various forms, alerts, and modal dialogs using both menu and screen (button) navigation.

The application's various forms should all display properly, bitmaps and fonts should be readable. In particular, if you are doing any custom drawing (bitmaps, gadgets) watch for whether or not they are properly redrawn on a form after a menu or modal dialog has been dismissed. If you have no modal dialogs or menus, you may want to set an Alarm in the Calendar application and then launch into your application so you can see how it reacts when a dialog is dismissed.

***Before going on with the rest of the test case make sure you set the Simulator back to the default display setup outlined above.*

UI Control Verification

Field Input and Edit Menu

1. Launch your application normally
2. For every field or custom rich text control verify that text entry works
3. If applicable, test Cut, Copy and Paste as well
4. Verify that text entry via the system pop up Keyboard works

Fields should handle text entry. If the application saves and restores text in a field, or sets text to a default value upon entry, you should verify that this is working correctly.

UI Objects

1. Launch your application normally
2. Tap through all of the various UI elements

- lists and pop-up lists
- buttons and repeating buttons
- checkbox
- tables
- menus
- sliders
- gadgets
- scrollbars
- popup triggers
- selector triggers

All UI elements should function error free and display properly.

Stress Testing

Gremlins

In order to maximize the effectiveness of Gremlins, it is ideal to use a number of different seed numbers.

1. With your application installed, right click on the Simulator and choose "Gremlins..."
2. Select your application, choose any seed number and set it to stop after 100000 events.
3. Start Gremlins.
   Note: If you do not care to watch the whole process you can choose "Disable display" from the Gremlins dialog to decrease the time it takes to run your gremlins. You also may also want to uncheck "Enable

Sound" setting for the duration of this test.

4. Repeat steps 1 through 3 with 9 unique seeding numbers for a grand total of 1,000,000 events.

The application should run error free.

Tips on writing your application to test well with Gremlins

Some applications do not test well under Gremlins because random taps either do not reach all the functionality or quickly eliminate some of the core components that allow for the application to run fully functioned. If this is the case with your application you will need to detect if Gremlins is running using the SysGremlins call (defined in the Palm OS Reference) or the Host Control API call HostGremlinIsRunning (defined in the Palm OS Emulator User Guide). Some suggestions for common situations follows.

Exit Buttons - If your application has some mechanism for exiting besides the Launcher icon you may find that Gremlins end up testing the Launcher rather than your application. This is because you are posting an appStop event to the system. The easiest way around this is to instead post a keyDown event with the command mask on and a chr of vchrLaunch. This will simulate a tap on the Launcher icon instead of forcing an exit. The Simulator is smart enough to ignore this event when running Gremlins.

Connection Dialogs - For processes that take a little while, such as connection dialogs, it is commonplace to have a Cancel button. However, Gremlins will typically hit the cancel button before the task is completed. You probably do want to test the cancel functionality as well, so instead of ignoring the cancel button completely, you want to limit how often the button will work. One good way is to check if gremlins is running, generate a random number and only proceed with the cancel action if the number generated is a particular number. Choosing the range to use for this random number should be based on how long it takes for the connection and the process that is going on.

Passwords - Applications with passwords upon entry will find that Gremlins never gets in at all, but rather spends all of its time testing the password dialog. Similar to the above example, you will want to test to see if Gremlins is running and automatically pass through the password dialog and on to the application some percent of the time.

App specific tests

In the end, you know your application best. A generic set of test cases will never test your application as well as you can with customized tests. Any special functionality your application implements needs targeted test cases. Some areas that you may want to consider are:

- Communications - If your application supports the sending and receiving of data, this functionality should be tested. In particular, any low level IR or serial communication type functionality should be tightly scrutinized.
- ARM-native vs. non ARM-native code - If you implement an ARM native portion of your application, you will need to compile a Windows Intel-native DLL to test this out in the simulator. Further, if your application also targets pre OS 5 devices, you will need to test out the 68K implementation of those sections of code that are written native for ARM or Intel. For this you should use the Palm OS Emulator with an OS 4.1 debug ROM.

General OS Compatibility

For the purpose of this document, "General OS Compatibility" refers to an application that meets the following criteria:

- Backwards and forwards compatibility within a range of Palm OS versions.
- Compatibility across licensee devices.
- Robustness - An application should be free of programming errors that result in such things as memory leaks, fatal exceptions, frozen devices, etc.

- System Consistency - An application should never inhibit the system from being able to perform its core tasks or prevent other applications from running correctly. For example, it is not acceptable for an application to stop the OS from posting low battery warnings, exiting to the Launcher, or notifying other applications that one of their alarms has gone off. Further, applications that change the default behavior of buttons, the color palette, sounds, etc. need to make sure they reset the device to the devices default behavior upon exit (unless the user is fully aware that these changes extend past the boundaries of the individual application.)
- UI Consistency - There is a level of UI consistency required to define something as being compatible with the system. This is the most vague of all the components that make up the criteria for compatibility. However, if the end user experience is so different from the typical behavior an application running on a Palm OS device that it would confuse the user, the application is detrimental to the entire user experience of the Palm and is not a compatible application. This is not to say that all applications must look exactly the same, but that there is a minimal requirement that includes things like making the edit menu available for text fields, consistent short cuts for common tasks such as beaming or deletion of a record, supporting the user-set system preferences, and not obscuring the Launcher icon. More information on UI consistency can be found in the Palm OS User Interface Guidelines document.

Applications should be tested with the above. Developing the right test cases can be a complicated task and, of course, the best course is to hire experienced QA personnel or contract a QA company. A couple of useful tips specifically for QA testing on Palm OS devices are:

- Take a look at the tests involved in the official Palm OS Compatibility Test offered through Quality Partners QA. The official test kit can be found at http://www.qpqa.com/palm/cs-index.html. It is a good idea to get the official logo, but even if you do not plan on taking the test through Quality Partners, you can use the test kit as a starting point for how to test your application.
- Use the Palm OS Emulator and the licensee specific versions to test on a large variety of devices
- Always use debug ROMs and make sure that all the options in the Emulator's debug setting menu are checked. Some developers feel that if something does not happen on a release ROM then it does not matter. The difference is that release ROMs do their best to hide any potential problems whereas debug ROMs uncover them. Using debug ROMs is the best way to minimize the chances that you miss a critical bug during your testing
- There are a number of books on QA and the processes behind it. One such text that may interest you is Testing Computer Software.