



Developing Handhelds

Targeting Personal Digital Assistant Platforms



June 6, 2002

Sasmito Adibowo

Arcle Technologies

SIMPLE RELIABLE SOLUTIONS



Table of Contents

About This Document	3
General Issues	3
Opposite Usage Patterns	3
User Interface Issues	3
Memory Considerations	4
Conserving Power	4
Platform Overviews	5
Palm OS	5
Windows CE	6
Linux	6
Development Tools	7
Native Code Compilers	7
Bytecode Interpreters	8
Java Environments	8
Summary	9
Targeting Palm OS	9
Targeting Windows CE	9
Targeting Linux	10
References	10



1 About This Document

Targeting development for Personal Digital Assistant platforms may be tricky. Even though Palm OS now is the market leader, Windows CE (a.k.a. PocketPC) is catching up. Not to mention the open source community bringing Linux into the battling arena of PDA platforms.

This document is a companion to a similarly-titled presentation. The presentation is intended to act as a guide for software developers in targeting the PDA platform for application development.

2 General Issues

2.1 Opposite Usage Patterns

A handheld is normally used for a task-specific device. When a user turns on his or her handheld, that user typically wishes to do something specific with his handheld, do it, and then resume his or her previous activity. The activity being performed on the handheld is most likely to be secondary to assist the user's primary activity.

For example, a user wishes to telephone an associate. She turns on her handheld, and enters the first few letters of the name of her associate. After finding the correct number, she uses it to dial the telephone. Then she can carry on her conversation on the telephone without the assistance of the handheld.

This is contrast to a PC, where its users could be sitting in front of it for hours at a time. He may be writing reports, forecasting business, or even programming – a rather time-consuming task.

Therefore, operations on a handheld must be accomplished easily and with minimal delays. Speed in this context refers to how fast a user can accomplish a task, not how many instructions the microprocessor is able to process in a second. Also, a handheld device only serves one user at a time – the one that is holding the device in her hand. This leaves out the throughput factor in performance, leaving latency as the one and only parameter for measurement.

2.2 User Interface Issues

The primary difference that one will soon see in differentiating a handheld and a PC is its size. While it is nice to have a 21-inch monitor for our desktop systems, having the same screen size of a PDA will reduce its practicality. Even when flexible paper-like screens become popular, it will still take space larger than a shirt pocket may provide.



The small screen size limits the information that can be displayed by the device. Users must only be presented with essential information, with all related data are near, or only one or two steps away to access.

The lack of a keyboard also limits the information that the user may provide. Until intelligent holographic voice-activated computers become massively available at low cost (think the credit-card size computer in Time Trax), we have to resort to alphanumeric input for the near future. Handwriting recognition technologies – no matter how good they will become – will never be on-par with keyboard data entry, since (when they are used to it) people tend to type faster than they write. Pickboards (like those used in mobile telephones) also will never match the keyboard, since it requires several keypresses to reach the desired character.

Programming an application to receive stylus input is also different than mouse input. Unlike mice, with styluses we will never get a “stylus move” message (unless the stylus is laser-pointed, but that’s another story). We only get “stylus tapped” message, along with the tapped position. Double-clicking when using a mouse is considered normal. Double tapping with a stylus will trouble the user, since she must tap at the exact location where the last time she tapped – not to mention the annoyance it causes.

2.3 Memory Considerations

To save weight and battery, a regular handheld PC does not come with a hard drive. Thus it must its main memory to store everything, from user data to working memory. Since a handheld’s memory is usually less than a desktop computer and the burden placed to it to store persistent data, applications will encounter out-of-memory situations more often.

PC software programmers are often too lazy to handle out of memory errors. Often, the program will simply display a message that it is out of memory and then exits. While this approach may be done on a handheld, it is not considered good practice in handheld programming. By doing that, it is possible that the user may not access any data at all from the offending application, until she deletes another application to free memory. Instead, applications must use the least memory possible at startup, and gracefully reduces its functionality when it cannot allocate memory to perform those functionalities.

2.4 Conserving Power

Electricity is scarce in handheld computers. Thus every effort must be done to conserve power, since a user will be annoyed when she founds out that her new application drains her batteries even when the application is active only for several minutes.



CPU usage must be kept low. Most handheld operating systems will suspend the CPU if there are no tasks that are ready. These intermittent CPU suspensions are done while keeping the display on, which gives the impression to the user that the handheld is still running. Generally, avoid lengthy computations, favor to call functions that may block the application (especially input message queue functions), and avoid polling.

Peripheral devices such as sound generators, serial ports, modems, IR ports must be used sparingly. Those devices are definite power hogs.

3 Platform Overviews

3.1 Palm OS

The Palm operating system is made and maintained by Palm, Inc, which is now a subsidiary of 3Com. From its beginnings, Palm OS was targeted only to PDAs. Therefore, this platform is optimized to run on a specific low-cost, low-performance processor. At that time, Palm decided that the Motorola 68K processors are the most suitable for that purpose.

The Palm OS platform consists of the Palm OS software, a reference hardware design, HotSync data synchronization, platform component tools, and software interfaces to support hardware add-ons.

Even though the Palm OS kernel supports task-switching, it does not expose that functionality to its applications. Only one application may run at any time. When the user selects another application, Palm OS requests the current application to stop.

Inter-application communication is gained through the use of *launch codes*. This is a predefined value given to the application at startup. Along with the launch code, several parameters may also be passed to the launched applications. One major functionality that takes advantage of this is the *global find* function. When the user taps the silk-screen magnifier button, Palm OS launches each application in a special *find* mode, indicating that the user is looking for text in its data.

Memory in Palm OS is represented as a 32-bit real address. Installed memory cards at the device are assigned to one or more blocks in this address space.

Compensating for the lack of virtual memory functionality, memory allocation is done through two steps. The first step is acquiring a handle to a moveable memory block. The second step is locking that handle to retrieve a pointer to the block in order to access it. After using that memory block, applications should immediately unlock the block so that it is moveable again. This way – when memory is fragmented – the operating system may move memory blocks around to coalesce free space so that memory



allocation requests may be satisfied. In the current implementations of Palm OS, memory defragmentation occurs only at failure of a memory allocation request.

Persistent data storage in Palm OS is not normally done through hierarchical file systems, instead through record-based databases. This is to allow the operating system to help manage the data – such as backing up on a per-record basis. Each record belongs to a user-defined category, up to a maximum of 15 categories per database. Even though, Palm OS provides API support for installable hierarchical filesystems – those filesystems are usually located in an external device.

3.2 Windows CE

This Microsoft operating system is actually built from the ground-up. Although one of its design goals is to be compatible with the Win32 API. At version 3.0, marketing strategies has changed the name of Windows CE to Pocket PC. This operating system requires higher-performance processors and larger amounts of memory. Fortunately, it does not depend on any specific family of microprocessors.

Even though targeted for small-scale devices, the Windows CE operating system is quite sophisticated. It supports multithreading, interprocess communications, virtual memory, COM, and other various functionalities normally available to desktop operating systems.

Being a miniaturized desktop operating system, data storage in Windows CE is a virtual disk backed by the main memory. It employs a standard FAT filesystem for its primary partition.

3.3 Linux

The PDA-targeted Linux systems are branches of the main Linux kernel developed for the desktop and server operating systems. Linux systems are jointly developed and maintained by groups of people known as the *open source community*. Because of the openness of Linux, its development path is not controlled by a single company or individual. Unfortunately, this causes the lack of standardization in Linux software and systems.

Most Linux PDA systems suffer from the same processor and memory requirements as Windows CE. On the bright side, these systems are also sophisticated miniature-desktop systems with fewer restrictive limitations than the Palm OS.

The lack of standardization and a major player has impeded the acceptance of Linux to the general market. There are various Linux distributions (variants), each with its own set of applications and GUI.



At the advent of the Sharp's entrance to the PDA market with its Zaurus SL-5xxx line of products, hopefully the Linux market will converge to a point where developers may start producing mass-market applications. Sharp takes and combines best-fit Linux solutions for its PDA, along with a Java environment to attract those skeptic developers.

4 Development Tools

4.1 Native Code Compilers

4.1.1 CodeWarrior for Palm OS

Being the development tool that was used to create the operating system in the first place, CodeWarrior is now the official tool for Palm OS development. It is a Windows or Macintosh-based integrated development environment, with a syntax highlighting editor, project manager, and source level debugger.

CodeWarrior supports low-to-mid level programming with C, C++, or inline assembly. Debugging may be done by using the bundled Palm OS Emulator, or remotely through a serial or USB cable to a Palm device. Support for new C++ language features such as namespaces, RTTI, and templates are available.

Lacking a sophisticated class library or RAD tool, CodeWarrior programmers must program directly to the Palm OS API. This is beneficial in some cases, since class libraries tend to take more space that increases the memory footprint on the final product. CodeWarrior makes that up by providing wizards and pre-built code snippets and examples to assist development.

4.1.2 Microsoft eMbedded Visual Tools

This is a version of Microsoft's visual studio for Windows CE targets. It offers the same bells-and-whistles of standard Microsoft development tools such as integrated editing, debugging, and project management. Bundling Visual C++ and Visual Basic in the same package, it is provided with *no-charge* at Microsoft's website.

Debugging may be done in two modes. One mode is compiling for the Intel processor to run as a regular Win32 application under the desktop operating system. It does this by providing a set of DLLs that simulates a Windows CE environment. The other mode is remote debugging directly at the device through a serial or USB cable attached to the PC.

The Visual C++ environment offers two way of programming. Developers may choose to program using the MFC library for Windows CE, which encapsulates most of the API. On the other hand, sophisticated system programmers may choose to program directly to the API.



4.1.3 Penright MobileBuilder

Developed by PenRight, MobileBuilder allows developers to target multiple platforms from a single code base. It does so by providing a proprietary API and code generators to target the multiple platform. Programming is done using the C language, which calls the MobileBuilder API functions.

The Windows-based MobileBuilder IDE provide RAD tools, similar to other RAD environments such as Visual Basic. Common handheld user interface elements are provided through drag-and-drop environment, where the developer only code to fill-in the functions that handles events from the user interface.

MobileBuilder provides over 350 API functions, focused for handheld uses. It includes handwriting recognition, signature capture, bar code scanning, and other functionalities appropriate for handhelds. PenRight requires that separate runtime licenses are purchased for each target platform.

4.2 Bytecode Interpreters

4.2.1 PocketC

PocketC provides simple, scripting language which are similar to C (just like JavaScript). It provides both desktop and handheld development environment. The current version of PocketC does not provide any GUI tools for building user interfaces, but such tools are available from third parties.

Different versions of PocketC is provided for both Palm OS and Windows CE. Unfortunately, they are not binary nor source-compatible between versions.

4.2.2 NS Basic

NS Basic provides the BASIC programming language to handheld development. Development is generally done on Windows, using RAD tools. On Windows CE, development is also possible to be done directly on the device.

Separate versions of NS Basic are available for Palm OS and Windows CE. But the resulting programs are not binary-compatible nor source-compatible.

4.3 Java Environments

4.3.1 Sun's KVM

The reference implementation for connected limited devices, the KVM serves as the testing ground and primary source for porting Java to other small device platforms. It supports a subset of the standard edition of Java, along with support for multithreading and exceptions.

Being a reference implementation, it is only targeted to run on Palm OS handheld. The other platforms that it supports are desktop Windows and Solaris (which obviously does not run on small systems). Sun chose



Palm OS for the primary testing ground because apart from its wide market acceptance, these devices are also light and resource-constrained, which presents a moderate challenge to KVM developers.

4.3.2 Insignia Jeode PDA Edition

This is a certified PersonalJava 1.2 VM implementation. It supports JNI, and features Dynamic Adaptive Compilation (DAC) that speeds up code execution. Primary targets are Windows CE and Linux. The Linux version supports both X11 and Qt/Embedded GUI systems).

4.3.3 Waba

Specified to be a strict subset of Java, Waba aims to provide very small footprint *and* Java compatibility. It is significantly smaller than KVM and more portable. Waba may be downloaded at no cost, and is provided to the terms of the GNU GPL.

Virtual Machine and class library size takes less than 64K on Palm OS and Windows CE systems. It is able to work in less than 10K working memory. VMs are available for various other platforms, including DOS, Newton, and even TI's graphic calculator.

Size reduction implies that Waba cannot implement many features. Indeed, it doesn't support multithreading, exceptions, and only supports a very few set of classes in the Java library. On the other hand, it provides support for serial and socket communications.

5 Summary

5.1 Targeting Palm OS

Palm OS is now the market leader in PDA platforms, and it seems that it will continue to do so in the near future. The operating system was built to run on low-cost hardware, which will allow it to be used on devices with wider varieties in the future. The philosophy behind the so-called Palm Economy also seems to contribute to its success in the handheld market. Not to mention support from other major companies such as IBM, Oracle, Sybase, and SAP that promote integration with enterprise products.

On the downside, real coding for the platform is not easy. The learning curve is high, requiring the coders to master C/C++ and low-level programming techniques. With the absence of an official class library for Palm OS, developers will need to be familiar with the operating system's API. The limited capacity of the Motorola Dragonball microprocessor also poses a barrier for the platform's development.

5.2 Targeting Windows CE

Similarity – both at the programmer's side and at the user's side – of the desktop Windows and handheld Windows offers leaner learning curves. But



exactly the same similarities are also the ones that degrades CE's market acceptance – the need for faster processor and more memory ends up in more cost and larger, heavier batteries. Microsoft's intentions to make CE **the** platform for in-car computing are also worth watching for.

5.3 Targeting Linux

While still in its infancy, PDA Linux systems are gaining acceptance – especially at the geek markets. These enthusiasts are willing to put out their money simply to get the opportunity of toying with their handhelds, running a real Linux system in their hand, modifying its kernel, hacking, and such.

But the openness of the platform offers another possibility: deployment of corporate-wide custom PDAs that are integrated with the company's information systems. Already several companies are taking their chances with this platform.

Coding for a regular Linux system requires a moderate-quality hacker. Doing the same thing for a Linux handheld system will require human resources with even greater quality and expertise. This is due to there are too many things to integrate (vast arrays of libraries and toolkits), and the absence of popular GUI development tools for these platforms.

6 References

- Campbell, Tom. *Programming Windows CE applications on your Windows CE machine*. Programming Power.
- Eplin, Jerry. *A developer's perspective on Sharp's Zaurus SL- 5000D Linux/Java PDA*. Nov. 12, 2001.
- Palm, Inc. *A Flexible Architecture for Innovative Solutions*. 2002.
<http://www.palmos.com/platform/architecture.html>
- Palm, Inc. *Palm OS Memory Architecture*.
<http://oasis.palm.com/dev/kb/papers/1145.cfm?print=true>
- Palm, Inc. *Zen of Palm*.
- Palm, Inc. *Palm OS Programmer's Companion, Volume I*.
- Penright. *MobileBuilder Spec Sheet*.
- Gregory, Kimberly . *Embedded Development with Microsoft Windows CE*. April 1998.
- Handspring. *The Springboard Platform Whitepaper*. 1999
- Insignia. *Jeode PDA Edition VM*. <http://www.insignia.com/>. 2002..
<http://www.insignia.com/>. 2002.
- Richter, Jeffrey. *Developing Applications for Microsoft Windows CE: An Overview of the Windows CE SDK and Visual C++ for Windows CE*. MSDN Library. March 15, 1997.
- Metrowerks. *CodeWarrior 8 for Palm OS Platform Datasheet*.
- Microsoft Corp. *Microsoft Windows CE Technical FAQ*. MSDN Library.



- Microsoft Corp. *Microsoft Windows CE: The New Choice for Dedicated Systems*. MSDN Library.
- Microsoft Corp. *Windows CE 2.10: Frequently Asked Questions*. July 27, 1998.
- Microsoft Corp. *Windows CE Memory Architecture*. MSDN Library.
- Sharp Corporation. "SL-Series" *Memory Information for Java™ & Qt/Embedded Applications*. (revision 1.00 2002.4.4)
- Sun Microsystems. *Java 2 Platform Micro Edition (J2ME) Technology for Creating Mobile Devices White Paper*. May 19 2000.
- Wireless Developer Network. *WirelessDevNet Online Training*.
<http://www.wirelessdevnet.com>