The Palm Operating System – Technology Overview

The Palm Computing Division of U.S. Robotics set out to create a compelling, low-cost, small form-factor, hand-held computer that connects seamlessly to Windows and Macintosh personal computers. The search for appropriate operating system software for this device led to the discovery of a gap in functionality between proprietary microcontroller-based system software on the low-end and PDA operating systems on the high end. To fill this gap U.S. Robotics created the Palm Operating System.

The Need for a New Operating System

Low-cost, hand-held devices, such as electronic organizers, cell phones and pagers, have relied on proprietary, microcontroller-based system software to achieve affordable price points and long battery life (see Figure 1.) The resulting devices are limited in software functionality, provide poor user interfaces and little or no connectivity to the desktop.

On the high-end are full multi-window, device-independent operating systems for PDAs and Personal Communicators. While these operating systems facilitate graphical user interfaces, applications development, and robust functionality, they require system resources that prevent the creation of fast, low-power, low-cost devices. Furthermore, even these high-end systems are not designed from the ground up to provide seamless connectivity with desktop PCs.

The Need For Connectivity

To make hand-held devices fast, power-efficient, and easy to operate, focused functionality is essential. Data viewing, lookup and limited data gathering is better suited for a hand-held device. The PC is ideal for data management, batch data entry, printing, backup, archive and configuration tasks.

To leverage resources appropriately, applications need to be partitioned between the hand-held and the PC, thereby streamlining the hand-held device with essential functionality and leveraging the PC to extend that functionality.

What is the Palm Operating System?

The Palm Operating System (Palm OS) is a hand-held computer operating system that enables low-cost, low power, small form-factor devices to integrate seamlessly with Windows or Macintosh personal computers. With their integrated connectivity, these devices actually extend the usefulness of the PC.

The Palm OS consists of two parts:

- Highly efficient operating system software that runs on a 68000-based hand-held computing devices.
- Windows or Macintosh-based system software that manages synchronization of the hand-held and the PC.

These interdependent components extend the traditional definition of an operating system to reflect the integral nature of connectivity in system design.

The first device in the Palm OS product line is Pilot, The Connected Organizer. A closer look at the Palm OS design goals will clarify what makes it work so well.

Design Goals

The Palm OS was designed with the following objectives in mind. On the hand-held side, it is designed for:

- **Speed and efficiency**: Provide nearly instantaneous response to user input while running on a Motorola 68000 type processor, requiring only 32K system memory.
- **Low-cost and low-power**: Provide months of battery life on 2 AAA batteries, utilizing standard, low-cost memory and processing components.
- **Small form-factor devices**: Facilitate the creation of pocket sized devices with user interface objects designed specifically for small displays; enable fast efficient data entry without the use of a keyboard.
- **Integrated PC connectivity**: Use record IDs, status flags and common data storage to facilitate communication and synchronization with desktop software.
- **Standard application development**: Facilitate application development in C or C++ using Metrowerks compilers and other common development tools.

On the PC side, it is designed for:

- **Efficient synchronization**: Synchronize hand-held data with multiple PC data sources without user intervention.
- **Extendability**: Enable Independent Software Vendors (ISVs) to develop "conduits", links between a wide range of desktop and hand-held applications.
- **Communication independence**: Insulate conduit developers from the communications protocols to facilitate synchronization via a variety of physical links.
- **Standard conduit development**: Under Windows, conduits are DLLs and are developed using standard C or visual basic tools.

To understand what makes the Palm OS work so well, a closer look at how it achieves these design goals is in order, starting with the hand-held side of the Palm OS.

Speed and Efficiency

The Palm OS memory manager facilitates fast access to system software, applications, and data, yet requires a minimum of nonvolatile, dynamic memory.

To enable the creation of devices with a limited amount of dynamic memory, the Palm OS data manager stores data in a database-like structure rather than a traditional file system. Traditional file systems require large amounts of memory because they first read all or a portion of a file into the memory buffer, then later write the updated memory buffer back to the disk or storage medium. Because of the latency involved with reading and writing to disk, data access is generally slower. For devices that store data in memory, it makes more sense to access and update data directly in place.

The Palm OS uses a database model to store data. It works with small chunks of data, less than 64K each. The system stores each data record, such as an address book entry, as a chunk. Chunks can be scattered throughout the memory space and accessed in place. This storage method speeds data access by allowing the system to add, delete and modify records in place.

The system groups related records, such as appointments or address book entries, in databases, which replace the traditional file concept. For example, the system stores the collection of all address book records in a database. The Palm OS works like a traditional file system with a fraction of the overhead. Applications can create, open, delete and close databases as necessary, just as files on traditional systems are created, opened, deleted and closed.

The Palm OS data manager facilitates fast data access and sorting. Each database may contain one or more presorted index lists to facilitate fast sorting. For example, the Address book application stores index lists of records by name and by company. This enables the application to switch data views instantaneously between the two.

In addition to accessing data in place, the Palm OS executes applications in place out of ROM or RAM (see Figure 2). The efficient program execution results in instant switching between applications, and eliminates the need for a wait cursor.

Low-Power Usage

The Palm OS minimizes power consumption with efficient power management. It supports three modes of operation: sleep mode, idle mode, and running mode.

When there is no user activity for a number of minutes, or when the user hits the off key, the device enters sleep mode. In sleep mode, the device appears turned off: the display is blank, the digitizer is inactive, the main clock is stopped, and the processor is not powered. To awaken from sleep mode, the device must receive an interrupt signal from a hardware button.

When the device is on but has no user input to process, it enters idle mode. In idle mode, the main clock is running, the device appears to be on, and the processor clock is running but the processor is halted.

In sleep or idle mode, when the system detects user input, it enters running mode. In running mode, the processor executes instructions. The device stays in running mode only as long as it takes to process the user input, then returns to idle mode.

Support for Small Form-Factor Devices

The Palm OS is designed specifically for small form-factor devices. In addition its feature set is targeted to small devices to eliminate unnecessary system overhead.

To facilitate data entry for small form-factor devices, the Palm OS provides optional support for Graffiti power writing technology. On other platforms, Graffiti is added to the operating system, requiring a Graffiti window to float on the screen for text input, thereby obscuring the screen. The Palm OS devices can integrate and support data entry through a dedicated Graffiti writing area. This integration enables Graffiti to work more efficiently, and maximizes screen usage for the application.

With Graffiti it is possible to limit the size of the writing area to less than one square inch, thereby enabling the creation of small devices. Because Graffiti recognizes each stroke as you write it, it is actually possible to enter strokes right on top of each other. When the user enters the stroke for a letter, the character appears, thereby readying the writing area for the next stroke.

The Palm OS supports user-interface elements designed specifically for small-screen devices. These elements enable developers to create robust applications on screens with a minimum of 160 x 160 available pixels.

Integrated PC Connectivity

The Palm OS data manager provides built-in functionality to facilitate efficient synchronization with the desktop.

Each database on the hand-held has a header that contains an attribute flag to indicate that at least one record in the database has changed since the last synchronization. In the event that no records have changed, the synchronization process can bypass the unchanged database entirely. Other systems must use a less-reliable date and time stamp on files to determine their status.

Most systems require that the entire file on the hand-held device be sent across the serial line and synchronized whenever a single record has changed. In the Palm OS database, each individual record header contains a status attribute that indicates whether the record is old, new, modified, deleted or archived. This header information reduces the amount of data transmitted across the serial line during a synchronization because usually only new or modified records must be sent.

In addition to the status attribute, each record has a unique record ID that matches records stored on the PC with records stored on the hand-held. In other systems, establishing a match between two records is more time consuming and less reliable because they must match key fields to determine that two records are alike. This requires that more data be processed simply to determine a match.

The Palm OS also has built-in archiving. There are two methods for removing a record on the hand-held: the user either deletes or archives it. When the user deletes a record, the deleted record's status flag is set to "delete" and the record data is removed. The system frees the data chunk from memory, but retains the record ID and marks it with a delete flag so that the PC knows to delete the same record during synchronization. For archived records, the system sets the status flag "archive." During synchronization, the system copies archive records into an archive file on the PC then removes the archived records from both the active PC file and hand-held device.

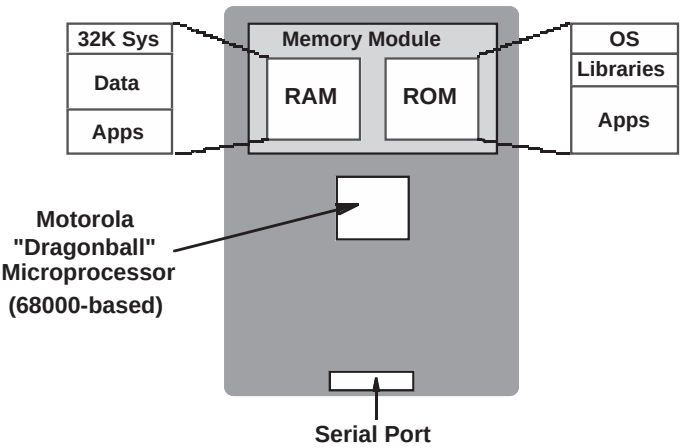

Figure 2

Standard Development Environment

In order to speed application development, the developer creates Palm OS applications using standard development tools on a Macintosh computer. The development environment uses Apple's Macintosh Programmers Workshop (MPW) and Metrowerks' CodeWarrior compiler. The developer writes the applications using the C language, although assembly language and C++ are also available. The developer debugs applications on the Macintosh using the source level debugger provided with CodeWarrior.

The Palm OS includes libraries that reproduce the functionality of the Palm OS system software under the Macintosh environment. In order to test applications on the Macintosh, the developer simply links the application code with Palm OS "simulator" libraries to generate a Macintosh executable. This executable runs from within the Macintosh based "simulator" that displays a window on the screen representing the hand-held's's display area. The application displays in this window as if it were running on the actual device.

In order to build resources, the Palm OS development environment includes a resource compiler that runs as an MPW tool. The developer creates a text file description of the resource which converts into a resource file to import into the emulation environment along with the application code.

To create applications that run on a Palm OS device, the developer links the application with the actual Palm OS libraries to create a device executable.

The Palm OS also enables applications to override the functions of the standard hardware buttons. For example, an application can re-map the function of the memo pad button to run a personal finance application instead. This facilitates the creation of customized devices, with personality-specific functionality.

Device applications can reside and be executed from the device's ROM or RAM. A launcher application running on the device scans for all available applications in ROM and RAM and displays an icon by which the user can launch the application. The user installs new applications distributed on diskette into the device by synchronizing.

Now let's examine the PC side of the Palm OS.

Efficient Synchronization

A synchronization manager application runs in the background on the Windows or Macintosh PC to enable one-button synchronization, or "HotSync," between the hand-held and the PC. This application monitors the serial port of the PC for a wake-up packet from the hand-held

device to begin synchronization. Once it receives a wake-up packet, it runs in turn each conduit installed in the system, systematically synchronizing each hand-held database with the associated database on the PC (see Figure 3). Conduits need not, however, create a mirror-image of data on the PC and hand-held. For example, an install conduit can be used to install applications from the PC to the hand-held. A finance conduit might upload new transaction data from the hand-held's's check register and, after they are integrated into the PC finance database, download a new balance

Each conduit uses the synchronization manager API to make calls during the synchronization process to do whatever is required during synchronization. This can include opening databases on the hand-held, retrieving and writing records, closing databases, and so on.

Extendability

The Palm OS enables synchronization between a wide range of desktop and hand-held applications in a single step. Other systems require developers to create their own connectivity solution. As a result, end users must use multiple connectivity solutions for the various applications on their hand-held devices.

In order to add a type of synchronization to the HotSync process, a developer simply creates a conduit, a PC-based application that manages data exchange between a database on the hand-held and a file on the PC. Conduits can perform a vast range of functions. One conduit might install applications from a diskette on the PC to the RAM on the hand-held device. Another conduit might synchronize the Address Book database on the hand-held with a file on the PC so that they are mirror-images of each other.

To add a conduit, the developer registers it with the synchronization manager. The synchronization manager runs all registered conduits in sequence to synchronize the data on the two devices during HotSync.

Conduits can synchronize any PC application's data with any of the databases residing on the hand-held. For example, the address book conduit retrieves all new, modified, deleted and archived records from the hand-held's's address book. It then updates the PC's address book with the new records and any modified records that have not changed on the PC. It then synchronizes any records changed on both the hand-held and the PC and deletes or archives any records deleted or archived on the hand-held. Next, it updates the hand-held with the set of records changed during the process. Finally, it readies both devices for the next synchronization by clearing the status flags for each database and record on both sides.

Communication Independence

Although other hand-held operating systems facilitate some level of connectivity with the PC, the burden of integrating communications is frequently the job of the developer.

Palm OS conduits are communication independent. The conduit developer need not worry about low-level communications protocols. The synchronization manager manages communications and runs transparently across a variety of communication media, whether a wired serial connection, or a wired or wireless modem. Therefore the conduit developer need only interface to the synchronization manager to access whatever communication media are supported by the device.

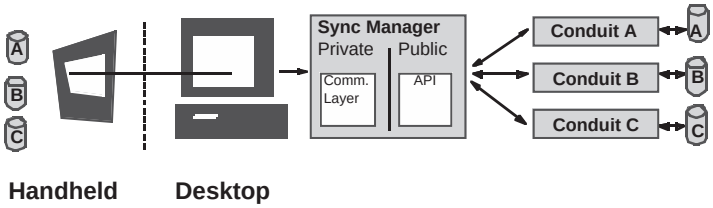

Figure 3

Standard Conduit Development

To facilitate easy development, conduits are developed using standard Windows or Macintosh development tools, depending on the target platform. Under Windows, developers create conduits using Visual C++ and Microsoft's MFC. Under Macintosh, the developer creates conduits using Apple MPW.

Implications for the Future

The connectivity focus of the Palm OS facilitates more targeted devices that can harness the power of the PC to extend functionality. For example, smart phones can download address book information entered using the full-sized PC keyboard, rather than forcing users to program phone numbers with the limited phone keypad.

The operating system extends the functionality of the PC by providing a means to link to and carry desktop data. It also extends the power of the hand-held device by leveraging the functionality of the PC applications to perform more powerful functions. Palm OS conduits will be developed for a wide range of desktop applications, including group scheduling, e-mail, personal finance and database.

As the Palm OS becomes more widely accepted, a new breed of devices will emerge. It will serve as the foundation for a wide range of increasingly smaller devices, including smart phones and graphical pagers. The Palm OS expands the opportunity for hand-held computing and extends the usefulness of the desktop computer. It will also enable network devices that can connect to the Internet for information access. ✔

Palm OS Specifications

Operating System size - 300K
Written in C
Graphical user-interface
Pre-emptive multi-tasking
User interface toolbox
Display and serial drivers
French and German localized versions
Graffiti® Power Writing technology

Palm OS Hardware Requirements

Microprocessor - Motorola 68328
Physical buttons for navigation
System Memory - 32K system memory plus data storage
Graphics display with digitizer (160 x 160 typical)
System ROM - 512K typical

System services

Resource manager
Event manager
Alarm manager
Power manager
Data manager
Memory manager
Timerr manager
Communication manager
Graffiti manager
Execution monitorr (kernal)
Sound manager

Palm OS Software Development Kit

Developers create Palm OS applications using standard development tools on a Macintosh computer. The development environment uses Apple's Macintosh Programmers Workshop (MPW) and Metrowerks' CodeWarrior. The developer writes the applications using the C language, although assembly language and C++ are also available, then debugs the application on the Macintosh using the source level debugger provided with CodeWarrior. Applications are tested on the Macintosh using the Pilot Simulator, a Macintosh application that simulates the functionality of a Pilot Organizer.

Device applications can reside and be executed from the device's ROM or RAM. The launcher application running on the device scans for all available applications in ROM and RAM and displays an icon by which the user can launch the application. New applications are distributed on diskette and loaded onto the devices by synchronizing the device with the PC.

The Palm OS SDK will be available in April, 1996. It consists of the following:

- System software libraries and headers: Code libraries and associated header files for all the Palm OS system software.
- System resources: ResEdit template files, fonts and bitmaps.
- Pilot Simulator: A Macintosh application that "hosts" a Palm OS application for execution on a Macintosh to simulate a Pilot device. The simulator also provides test and debug capabilities.
- Debug and console application: A Macintosh-based debugger for debugging remote applications on the Pilot device.
- Documentation: Printed and HTML-based on-line documentation, including a tutorial and style guidelines (available on the Palm Web Site: http://www.usr.com/palm)
- MPW and Code Warrior extensions: Scripts and utilities that aid in the development process.
- Install Conduit: A conduit that installs applications from the Macintosh to the RAM of the Pilot device during a synchronization.
- Other Utilities: Project files, application shell code and makefiles.
- Sample Code: Well-documented source code to illustrate how to develop an application. Referenced in the Tutorial.

Palm Conduit Software Development Kit

To facilitate easy development, conduits are developed using standard Windows or Macintosh development tools, depending on the target platform. Under Windows, developers create conduits using Visual C++ and Microsoft's MFC. Under Macintosh, the developer creates conduits using Macintosh's MPW.

The Palm Conduit SDK will be available in April, 1996. It consists of the following:

- Libraries: Libraries that provide API calls to exchange data between the PC and the handheld device via the Synchronization Manager, an application that runs in the background on a Windows or Macintosh PC.
- Documentation: Printed and HTML-based on-line documentation (available on the Palm Web Site http://www.usr.com/palm
- C and C++ Header Files: Header files with definitions of structures and prototypes of functions.
- Sample Code: Well-documented sample conduits, referenced in the documentation, makefile templates and a basic conduit template.
- Install Utilities: Programs to set up the Conduit SDK onto a host computer, and a utility to register new conduits with the Synchronization Manager.