Intro to the Palm OS® and application programming

Palm Platform Hardware

- Total memory (RAM/ROM)
  - originals had only 128 Kb
  - currently average is 4 Mb (max. 8Mb)
  - 32 bit addresses
  - 8, 16 & 32 bit data types
  - OS consumes 86 Kb in latest variation
  - cost of memory read or write 5 - 8 CPU cycles

Palm Platform Hardware (cont.)

- Processor
  - Motorola Dragonball chip (MC68328 / MC68EZ328 / MC68VZ328)
  - Processor Speed 16 - 33 Mhz (Avg 20 Mhz)
  - 16 bit bus

Palm Platform Hardware (cont.)

- Display
  - refresh rate 85 Hz avg.
  - Originally supported 1 bit , then 2 bit monochrome, 8 monochrome and 16 bit color (with separate display controller).
  - 160 x 160 pixel size
- Additional support HW
  - TCP
  - IR

Issues with OS / Application development

- Quick Turnaround Expected
  - Each time user must start application (no multiple applications running at same time)
  - Accessed multiple times a day as opposed to a PC which may be start a left running
- PC Connectivity
  - Integral part of use of the Palm (Data backup on PC due to memory volatility)

Issues with OS / Application development

- Power
  - Limited processing power due to battery
  - System is actually always running in a reduced power consumption mode.

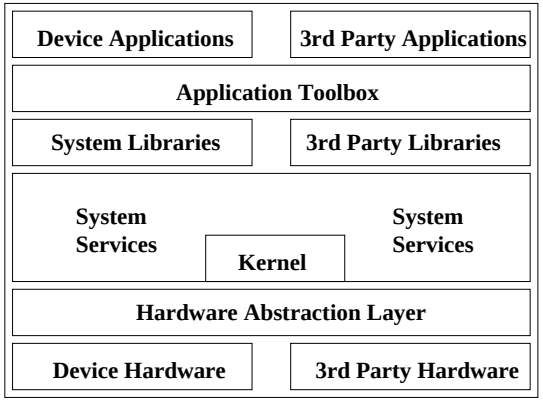
Issues with OS / Application development (cont)

- Memory
  - All memory on cards (ROM and RAM)
  - Memory is always powered
  - RAM
    - **Dynamic Heap**
      - originals 32 Kb / average 128 Kb / max 256 Kb
      - depends on total memory
      - 40 Kb for OS variables / 32 Kb for TCP/IP stack
      - 184 Kb for call stack, local, global and static variables
    - **Remaining is for Program and Data Storage**
  - ROM
    - Built in application and OS stored in ROM

Issues with OS / Application development (cont)

- File System
  - non traditional, uses records as part of a database
  - Stored in RAM with edits taking place in RAM
- Backward compatibility
  - more so an issue with application development
  - many versions of Palm platform and OS
- Screen size / input digitizer
  - more so an issue with application development
  - mapped to memory

Palm OS Platform Components



Palm OS Platform Components (cont.)

- Device Applications & 3rd Party Applications
  - User applications
    - PIMS apps
    - Mail
    - iMessanger™ App
    - Games
    - etc.

Palm OS Platform Components (cont.)

- Application Toolbox
  - Provides interface to system utilities and libraries
  - Examples:
    - CodeWarrior Interactive Development Environment (IDE) from 3Com ®
    - Palm OS ® Software Development Kit.
      - Palm OS Constructor to create UI resources.

Palm OS Platform Components (cont.)

- System Libraries
  - TCP/IP
  - Floating Point
- 3rd Party Libraries
  - Java
  - Communications

Palm OS Platform Components (cont.)

- System Services
  - Graffiti® Manager
    - Allows input character via screen OCR
  - Key Manager
    - interfaces to the HW keys on
  - Pen Manager
    - allows input similar to the mouse on a PC

Palm OS Platform Components (cont.)

- System Services
  - Memory Manager
    - Maintains the memory allocations for the system
  - Data Manager
    - Stores data in databases (like files)
  - Resource Manager
    - Stores data like Data Manager except a allows a tag of resource type and id

Palm OS Platform Components (cont.)

- System Services (cont)
  - Sound Manager
    - Allows the reproduction of 1 channel midi sound
  - Serial / Modem / SLP Managers
    - Allows serial type interface control for simple serial / modem / Palm serial specific communication

Palm OS Platform Components (cont.)

- System Services (cont)
  - Feature Manager
    - Provides a means to determine if a feature exists
      - Wireless support
      - OS version
    - Save data between applications launches
    - Develop user defined features

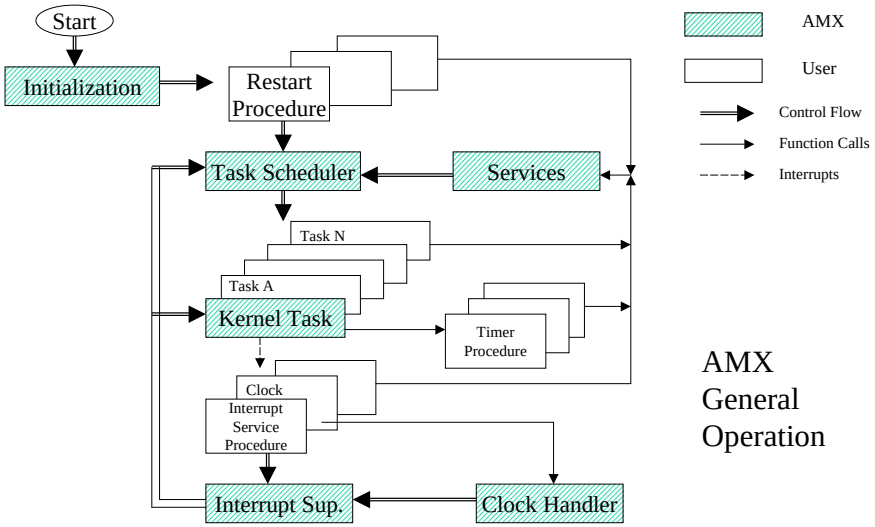
Palm OS Platform Components (cont.)

- System Services (cont)
  - Event Manager
    - Handles the interface between the application and events generated by other managers
  - Text , International and Overlay Managers
    - Allow the developer to produce one program and provide multiple language support
  - Exchange Manager
    - Allow synchronization of data across platforms (i.e. PCs)

Palm OS Platform Components (cont.)

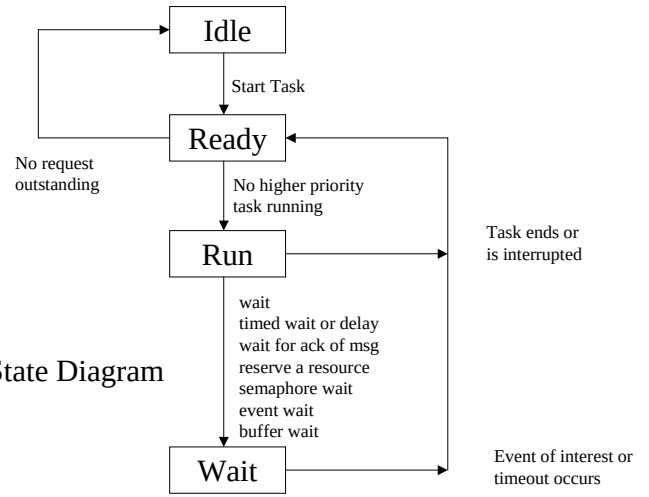
- Kernel
  - Based upon the Kadak AMX kernel
    - The underlying kernel provides multi-tasking support, however Palm OS does not support it for user level interaction
    - Provides basic microkernel type functions
      - ISR
      - Semaphores
      - timers
    - Task Control Block (TCB)

Palm OS Platform Components (cont.)



AMX General Operation

Palm OS Platform Components (cont.)



AMX Task State Diagram

Palm OS Platform Components (cont.)

- Hardware Abstraction Layer
  - Allows both kernel and application SW to be stable with changes in the HW interface
    - Memory Cards
    - GPS Cards
    - Wireless communication
  - Different makers of Palm Platform PDA's
    - Palm
    - Handsprings Visor
    - Sony CLIE

CodeWarrior for PalmOS

- CW runs on Macintosh,Windows 95/98 or Windows NT/2K/XP
- CodeWarrior for PalmOS version 8.0
  - Includes Symbol's SDK
- CW Lite
  - Demo version available at the CodeWarrior web site

CW 8.0 Components

- C/C++ compiler that generates code for your PDA
- An **I**ntegrated **D**evelopment **E**nvironment
- A Linker
- A Post Linker that bundles resources and applications
- A source and assembly debugger
- A Constructor for easy resource creation
- PalmOS SDK 3.0 or 4.0
- Sample applications

CodeWarrior Directory Structure

..\Program Files\Metrowerks
- CodeWarrior
  - Bin
  - Palm OS Emulator
  - Scanner_SDK
  - PalmOS Support
    - Docs
    - Examples
    - Incs
    - Libraries
    - Tutorial

Resource Types

- Forms
- Menu Bars
- Menus
- Strings
- String lists
- AppInfo string lists
- Alerts
- Icons
- Bitmaps

Constructor

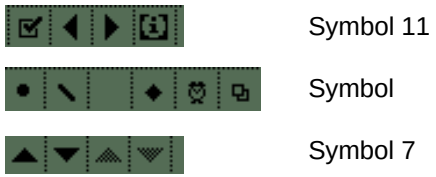
- Visual Resource Editor
- Forms are created and outlined
- Form resources are dragged and dropped onto the form similar to Visual Basic
  - Form resources are available in the “Catalog” option menu
- Each resource has a ResourceID generated automatically in the header file
  - **Do not edit the resource header file**

Catalog Resources or Form Objects

- Button,Push Button or Repeating button
- Checkbox
- Text Fields
- Form Bitmap
- Gadget
- Graffiti Shift Indicator
- Label
- List boxes
- Popup Trigger
- Tables
- Scrollbar

Built-in Fonts

- Three built in fonts can be used with CW
  - Standard,Bold and Large
- Other built-in fonts available with Constructor
- In each font an ASCII code identifies a specific non standard character
  - I.e.



PalmOS Variable Types

- VoidHand,VoidPtr,CharPtr
- UInt,Int or UIntPtr
- Boolean
- Ulong,Long
- FrmPtr,FieldPtr,ListPtr,ControlPtr,FormTitlePtr
- Err
- ...

Creating a project

- A project consist of several folders
  - "src" folder contains the source code and header file
  - Resource.frk
    - Heritage from Macintosh
    - Contains data from resource files
- A project has an "mcp" extension
  - i.e. starter.mcp
- A project is always created from a minimum project
  - Contains the bare minimum functions
  - Starter.mcp is the default (starting) project
    - Wizard Generator

CW Debugger

- Debugs an application as it runs on the PDA
- Uses same serial connection as HotSync
  - Requires to quit HotSync
- Application must contain debugging information
  - Select menu "Project" followed by "Enable Debugging"
- Launch debugger from within the IDE
- Set PDA in console mode
  - Shortcut Graffiti character followed by two taps and number 2
- Debugger can be used with POSE

PalmOS Emulator or POSE

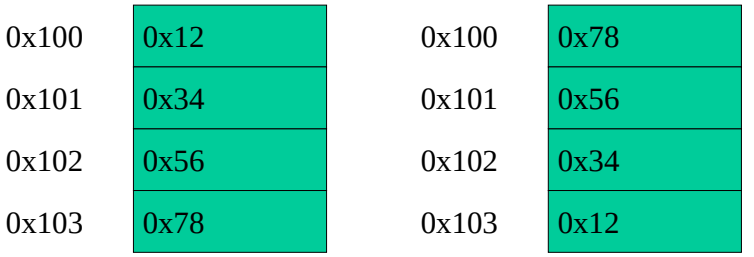
- Pose is a Palm III emulator running on the PC
  - Pose requires a ROM to be loaded first
- Options (when you right click the mouse)
  - Reset
  - HotSync
  - Load Application or PDB file
  - Upload a copy of the ROM from a device
  - Save a copy of the screen
  - Gremlins
    - Random sequence of events to fully test for bugs
    - Gremlins are available for the Palm III or specific app

Portability

- Palm uses a Motorola chip
- PC uses an Intel processor
- Bytes on both platforms are stored differently
  - On Palm 0x2056 is stored in memory with 0x20 first
  - On PC 0x2056 is stored in memory with 0x56 first
- When writing conduits a swapping function needs to be created that handles this issue
  - Strings do not have this problem since they end with '\0' character on both systems

Host-dependent Data Representation

- Big Endian and Little Endian
  - How do we store the integer 0x12345678 ?



Big Endian

Little Endian

Big and Little Endian Users*

- Big Endian
  - PowerPc
  - Sun Sparc
  - HP Workstation
- Little Endian
  - Dec Alpha
  - Intel Pentium

Some systems (MIPS 2000 and Intel i860) can use either big endian or little endian. The Intel i860 can even change modes while a program is running!

*Source: Unix network programming, W. Richard Stevens

Network Protocols Must Adopt One of the Two Byte Orders

- Network Byte Order
  - The protocol byte order
- Host Byte Order
  - The native machine byte order
- Conversion Functions
  - htons : convert 16-bit value from host byte order to network byte order. (ntohs provides the inverse)
  - htonl: convert 32-bit value from host byte order to network byte order. (ntohl provides the inverse)

About the PalmOS

- PalmOS is Trap-based
- API calls don't access function directly
- API call trips Trap
- OS looks up Trap in table, calls function

PalmOS Guts

- From Window.h:

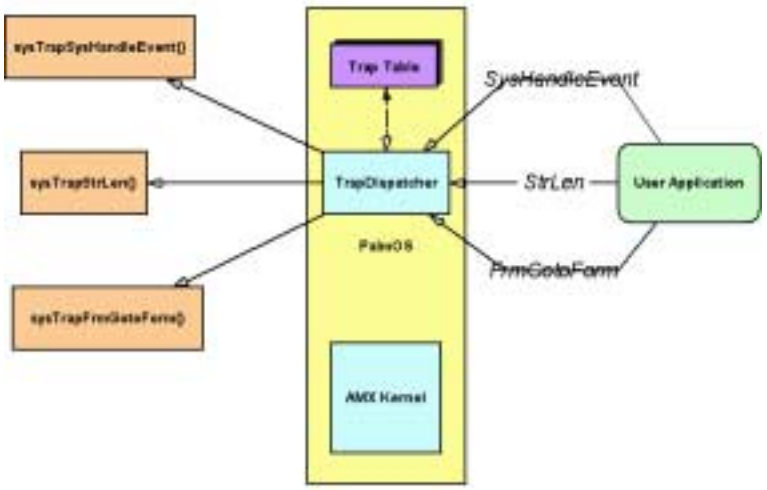
```
extern void WinDrawLine (Coord x1, Coord y1, Coord x2, Coord y2)
                                    SYS_TRAP(sysTrapWinDrawLine);
```

- From PalmTypes.h:

```
#define SYS_TRAP(trapNum)  _SYSTEM_API(_CALL)(_SYSTEM_TABLE, trapNum)
```

- See CoreTraps.h for a full list of Traps (more traps than exposed functions)

PalmOS Traps



The Trap Table

- The Trap table can be accessed directly!
- SysGetTrapAddress()
- SysSetTrapAddress()
- Trap table can be edited at *any* time (so be careful)

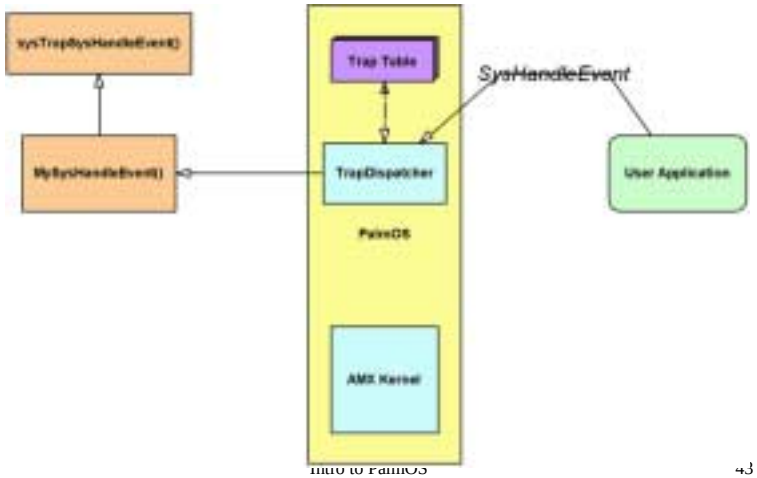
The Trap Table

- SysGetTrapAddress() returns Void* to code block
  - Use to get direct address of function for efficiency
  - Call right before tight loop

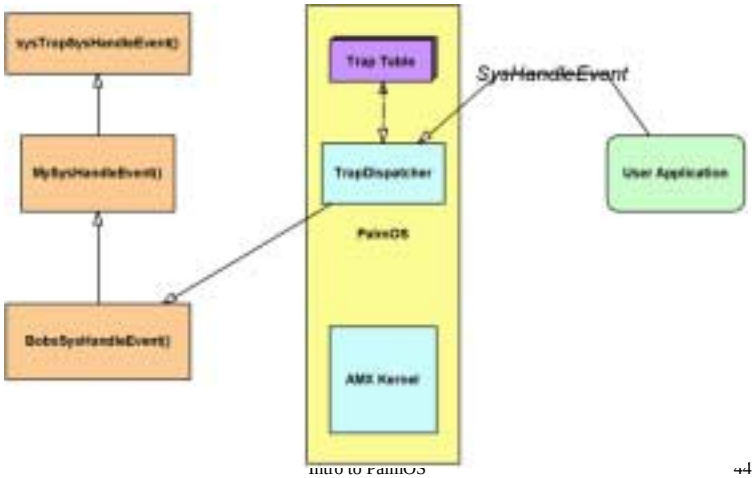
The Trap Table

- SysSetTrapAddress() sets new code block for Trap
- Do not do this yourself!
- If you forget to change it back, all kinds of Bad Stuff happens
- If another process patches the same Trap, Bad Stuff happens

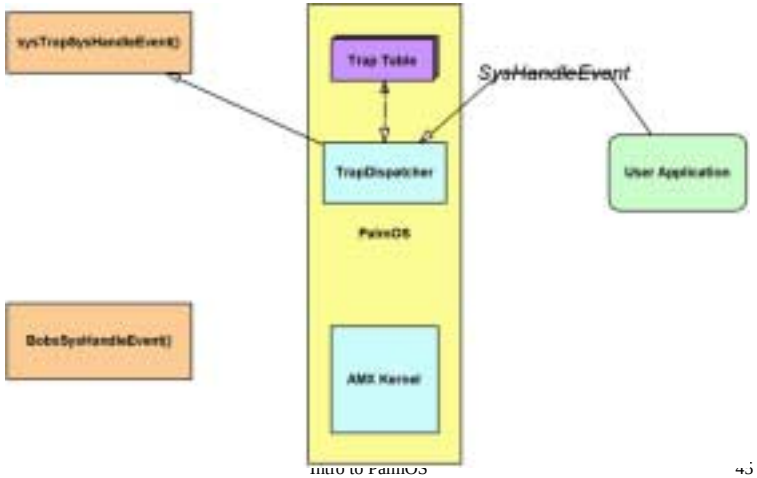
Bad Stuff Explained



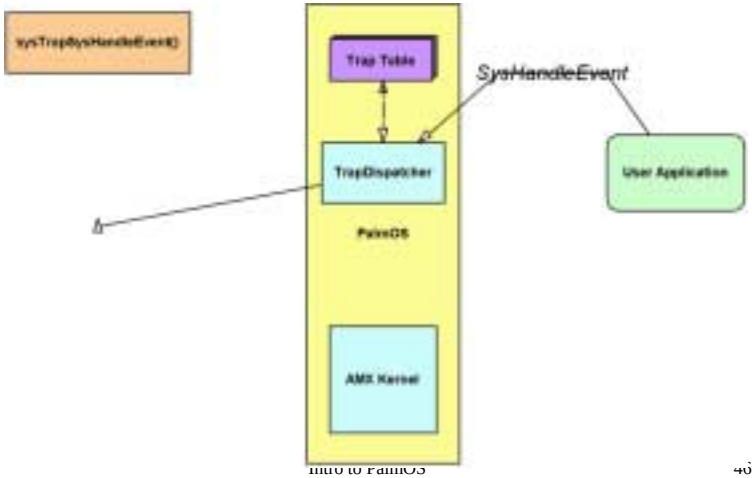
Bad Stuff Explained



Bad Stuff Explained



Bad Stuff Explained



Avoiding Bad Stuff

- Hack Managers handle tracking what has patched what for you.
- Hack Master was original Hack Manager, several more since
- Teal Master (shareware), X-Master (freeware), EVPlugBase, etc.

Hack Master “API”

- Set of guidelines for how to structure a code block for Hack Manager.
- “Free floating snippets of code”
- … No globals
- Stores Trap address info in Features
- Most hacks written with gcc, not CodeWarrior. CW doesn’t like PilRC and hacks