Power management from Linux kernel to Android

Liran Sinay
Yedidel Elhayany
Power Management
from Linux Kernel to Android
Introduction to Linux Power Management
Concepts behind Android Power Management
Design and Implementation
Room for Improvements
Introduction to Linux Power Management

- Goal of Power Management
- APM vs. ACPI
- APM emulation
- No silver bullet
- Manage power in different power state
- Lighthouse in the sea

Concepts behind Android Power Management

- Design and Implementation
- Room for Improvements
Power Management Basics

Prolong Life
Advanced Power Management

Applications
- APM-Aware Application
- APM-Aware Application
- APM-Aware Device Driver
- APM-Aware Device Driver

Operating System
- APM Driver
- APM Interface

BIOS
- APM BIOS
- APM BIOS Controlled Hardware

Add-In Device

OS dependent

OS independent
Device Responsiveness Decreases

APM power state

- Full On
- Off
- APM Enabled
- APM Suspend
- APM Standby

- APM Enable
- Enable Call
- Off Switch
- Off Call
- Short Inactivity
- Standby Call

- APM Disable
- Disable Call
- Resume Event

- On Switch
- Wakeup Event
- Suspend Interrupt
- Suspend Call

- Off Switch
- Off Call

- Power Usage Increases

POWER MANAGED
## Power management events

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Standby Request Notification</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>System Suspend Request Notification</td>
<td>0x0002</td>
<td></td>
</tr>
<tr>
<td>Normal Resume System Notification</td>
<td>0x0003</td>
<td></td>
</tr>
<tr>
<td>Critical Resume System Notification</td>
<td>0x0004</td>
<td></td>
</tr>
<tr>
<td>Battery Low Notification</td>
<td>0x0005</td>
<td></td>
</tr>
<tr>
<td>Power Status Change Notification</td>
<td>0x0006</td>
<td></td>
</tr>
<tr>
<td>Update Time Notification</td>
<td>0x0007</td>
<td></td>
</tr>
<tr>
<td>Critical System Suspend Notification</td>
<td>0x0008</td>
<td></td>
</tr>
<tr>
<td>User System Standby Request Notification</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>User System Suspend Request Notification</td>
<td>0x000A</td>
<td></td>
</tr>
<tr>
<td>System Standby Resume Notification</td>
<td>0x000B</td>
<td></td>
</tr>
<tr>
<td>Capabilities Change Notification</td>
<td>0x000C</td>
<td>Due to setup or device insertion/removal</td>
</tr>
</tbody>
</table>
## Power management functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>APM Installation Check</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>APM Real Mode Interface Connect</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>APM Protected Mode 16-bit Interface Connect</td>
<td>0x02</td>
<td>Avoids real or virtual86 mode.</td>
</tr>
<tr>
<td>APM Protected Mode 32-bit Interface Connect</td>
<td>0x03</td>
<td>Avoids real or virtual86 mode.</td>
</tr>
<tr>
<td>APM Interface Disconnect</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>CPU Idle</td>
<td>0x05</td>
<td>Requests system suspend.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0) Clock halted until timer tick interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1) Slow clock[^1]</td>
</tr>
<tr>
<td>CPU Busy</td>
<td>0x06</td>
<td>Driver tells system APM to restore clock speed of the CPU.</td>
</tr>
<tr>
<td><strong>Set Power State</strong></td>
<td>0x07</td>
<td></td>
</tr>
<tr>
<td>Enable/Disable Power Management</td>
<td>0x08</td>
<td>Set system or device into Suspend/Standby/Off state.</td>
</tr>
<tr>
<td>Restore APM BIOS Power-On Defaults</td>
<td>0x09</td>
<td></td>
</tr>
<tr>
<td>Get Power Status</td>
<td>0x0A</td>
<td>Supports AC status &quot;On backup power&quot;. And battery status.</td>
</tr>
<tr>
<td>Get PM Event</td>
<td>0x0B</td>
<td>Checks for APM events. Shall be called once per second.</td>
</tr>
<tr>
<td>Get Power State</td>
<td>0x0C</td>
<td></td>
</tr>
<tr>
<td>Enable/Disable Device Power Management</td>
<td>0x0D</td>
<td></td>
</tr>
<tr>
<td>APM Driver Version</td>
<td>0x0E</td>
<td></td>
</tr>
<tr>
<td>Engage/Disengage Power Management</td>
<td>0x0F</td>
<td>APM management for a specific device.</td>
</tr>
<tr>
<td>Get Capabilities</td>
<td>0x10</td>
<td></td>
</tr>
<tr>
<td>Get/Set/Disable Resume Timer</td>
<td>0x11</td>
<td></td>
</tr>
<tr>
<td>Enable/Disable Resume on Ring Indicator</td>
<td>0x12</td>
<td></td>
</tr>
<tr>
<td>Enable/Disable Timer Based Requests</td>
<td>0x13</td>
<td></td>
</tr>
<tr>
<td>OEM APM Installation Check</td>
<td>0x80</td>
<td>Tells if APM BIOS supports OEM hardware dependent functions.</td>
</tr>
<tr>
<td>OEM APM Function</td>
<td>0x80</td>
<td>Access to OEM specific functions.</td>
</tr>
</tbody>
</table>
# Power states

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0/Working</td>
<td>System is on. The CPU is fully up and running; power conservation operates on a per-device basis.</td>
</tr>
<tr>
<td>S1 Sleep</td>
<td>System appears off. The CPU is stopped; RAM is refreshed; the system runs in a low power mode.</td>
</tr>
<tr>
<td>S2 Sleep</td>
<td>System appears off. The CPU has no power; RAM is refreshed; the system uses a lower power mode than S1.</td>
</tr>
<tr>
<td>S3 Sleep</td>
<td>System appears off. The CPU has no power; RAM is in slow refresh; the power supply is in a reduced power mode. This mode is also referred to as 'Save To RAM'.</td>
</tr>
<tr>
<td>S4 Hibernate</td>
<td>System appears off. The hardware is completely off, but system memory has been saved as a temporary file onto the harddisk. This mode is also referred to as 'Save To Disk'.</td>
</tr>
<tr>
<td>S5/Off</td>
<td>System is off. The hardware is completely off, the operating system has shut down; nothing has been saved. Requires a complete reboot to return to the Working state.</td>
</tr>
</tbody>
</table>

Source: [http://www.lifsoft.com/power/faq.htm](http://www.lifsoft.com/power/faq.htm)
APM vs. ACPI

**APM**
- Control resides in BIOS
- Uses activity timeouts to determine when to power down a device
- BIOS rarely used in embedded systems
- Makes power-management decisions without informing OS or individual applications
- No knowledge of add-in cards or new devices (e.g. USB, IEEE 1394)

**ACPI**
- Control divided between BIOS and OS
- Decisions managed through the OS
- Enables sophisticated power policies for general-purpose computers with standard usage patterns and hardware
- No knowledge of device-specific scenarios (e.g. need to provide predictable response times or to respond to critical events over extended period)
APM emulation

```
/dev/apm_bois

kapmd

user_pmd

battery_drv

pmu_drv

pm_prepare

pm_enter

pm_finish

get_pm_status_drv

mach_pm_ops
```
Activity

Shut down

User press Power Button (System Reboots)

Run

CPU Off
LCD Off
Screen Off
Keys Off
Audio Off
DRAM Off
RTC Off
...

Standby

CPU On
LCD Off
Screen Off
Keys On
Audio On
DRAM On
RTC On
...

Activity

Sleep

CPU Sleep
LCD Off
Screen Off
Keys Off
Audio Off
DRAM Off
RTC Self-Refresh
RTC On
...

Inactivity

Sleep

CPU Sleep
LCD Off
Screen Off
Keys Off
Audio Off
DRAM Off
RTC Self-Refresh
RTC On
...

Inactivity

Standby

CPU On
LCD Off
Screen Off
Keys On
Audio On
DRAM On
RTC On
...

Activity

Reboot

CPU On
LCD On
Screen On
Keys On
Audio On
DRAM On
RTC on
...

Run

CPU Off
LCD Off
Screen Off
Keys Off
Audio Off
DRAM Off
RTC Off
...

Shut Down

CPU Off
LCD Off
Screen Off
Keys Off
Audio Off
DRAM Off
RTC Off
...

Inactivity

Reboot
Application-driven power management

- Micro manage your device
  - WiFi -enable PS poll mode
  - Switch on/off device by demand (Android's concept)
  - Gating off unused device clock

- Keep the flexibility of CPU
  - Gating CPU freq dynamically
  - Tickless idle
  - Using DMA instead PIO
Suspend/Sleep Mode

STR vs STD
- Suspend to RAM
- Suspend to Disk/Hibernate

STR: A technique by which systems preserve state in RAM during suspend and restore system state from RAM upon resume

STR is suitable for mobile device if hardware supports this function
# echo mem >/sys/power/state

Stop tasks

pm_registered_dev
  |-> bus_suspend
  |  |→ dev_suspend
  |  |→ bus_suspend
  ..........

- keep GPIO state
- Set SDRAM as self-refresh mode
- Request CPU to sleep

Restarting tasks

pm_registered_dev
  |-> bus_resume
  |  |→ dev_resume
  |  |→ bus_resume
  ..........

- restore GPIO state
Practical Example - S3C244x

Diagram:

- **IDLE**: 
  - IDLE_BIT=1
  - Transitions to IDLE from Interrupts, EINT[0:23], RTC alarm

- **RESET**: 
  - Transitions to NORMAL (SLOW_BIT=0)

- **NORMAL (SLOW_BIT=0)**: 
  - Transitions to SLOW (SLOW_BIT=1)
  - Transitions to SLEEP BIT=1

- **SLOW (SLOW_BIT=1)**: 
  - Transitions to NORMAL (SLOW_BIT=0)

- **SLEEP**: 
  - Transitions to SLEEP BIT=1 from EINT[15:0], RTC alarm

Legend:
- Arrows indicate transitions based on conditions.
- States are represented by ellipses.
Practical Example - S3C244x (2)
Provide a kernel module to dump and check GPIOs
Provide power source as independent as possible
Don't ignore the poweroff state
To most difficult part of debugging STR is resuming
Introduction to Linux Power Management

Concepts behind Android Power Management

- Basically, it is the slim wrapper for Linux Power Management

Design and Implementation

Room for Improvements
Where is Power Manager?
In fact, power ghost exists everywhere

- Let “*grep -ri power*” tell you about the details!
- It is layered into several components, but implementation involves in some hacks.
- Check: CONFIG_PM, sysfs, device drivers, libhardware_legacy, libril, init.rc, powerd, alarm, vold, JNI, PowerManager, BatteryManager, ...
Android does rely on Linux Kernel 2.6 for core system services

- Memory/Process Management
- Device Driver Model
- sysfs, kobject/uevent, netlink

Android Kernel extensions

- Binder
- android_power

Key Idea: Android attempts to provide an abstraction layer between hardware and the related software stack.
Android's PM Concepts

- Android PM is built on top of standard Linux Power Management.
- It can support more aggressive PM, but looks fairly simple now.
- Components make requests to keep the power on through "Wake Locks".
  - PM does support several types of "Wake Locks".

- If there are no active wake locks, CPU will be turned off.
- If there are partial wake locks, screen and keyboard will be turned off.
Types of Wake Locks

- **ACQUIRE_CAUSES_WAKEUP**
  - Normally wake locks don't actually wake the device, they just cause it to remain on once it's already on. Think of the video player app as the normal behavior.

- **FULL_WAKE_LOCK**
  - Wake lock that ensures that the screen and keyboard are on at full brightness.

- **ON_AFTER_RELEASE**
  - When this wake lock is released, poke the user activity timer so the screen stays on for a little longer.

- **PARTIAL_WAKE_LOCK**
  - Wake lock that ensures that the CPU is running. The screen might not be on.

- **SCREEN_BRIGHT_WAKE_LOCK**
  - Wake lock that ensures that the screen is on at full brightness; the keyboard backlight will be allowed to go off.

- **SCREEN_DIM_WAKE_LOCK**
  - Wake lock that ensures that the screen is on, but the keyboard backlight will be allowed to go off, and the screen backlight will be allowed to go dim.
PM State Machine

Touchscreen or keyboard user activity event or full wake locks acquired.

Partial wake locks acquired

All partial wake locks released

Applications

- Application A
  - new wake lock

Applications Framework

- Power Manager
  - create wake lock

Linux Kernel

- Power Management
  - control
  - CPU
  - LCD
  - Keyboard

Partial Wake Lock

Full Wake Lock

Applications

- Application A
- Application B
- Application C

Applications Framework

- PowerManager
  - Android.os.PowerManager
- Power
  - android.os.Power
- PowerManagerService
  - Android.service.PowerManagerService

Libraries (user space)

- Core Libraries

- JNI

Linux Kernel

- Linux Drivers
- Android Power Management/drivers/android/power.c
- Android_register_early_suspend()
- Android_register_early_resume()
Introduction to Linux Power Management

Concepts behind Android Power Management
  - Basically, it is the slim wrapper for Linux Power Management

Design and Implementation

Room for Improvements
Design and Implementation
frameworks/base/core/jni/android_os_Power.cpp

... static JNINativeMethod method_table[] = {
    { "acquireWakeLock", "(ILjava/lang/String;)V", (void*)acquireWakeLock },
    { "releaseWakeLock", "(Ljava/lang/String;)V", (void*)releaseWakeLock },
    { "setLastUserActivityTimeout", "(J)I", (void*)setLastUserActivityTimeout },
    { "setLightBrightness", "(II)I", (void*)setLightBrightness },
    { "setScreenState", "(Z)I", (void*)setScreenState },
    { "shutdown", "()V", (void*)android_os_Power_shutdown },
    { "reboot", "(Ljava/lang/String;)V", (void*)android_os_Power_reboot },
};

int register_android_os_Power(JNIEnv *env)
{
    return AndroidRuntime::registerNativeMethods(
        env, "android/os/Power",
        method_table, NELEM(method_table));
}

static void acquireWakeLock(JNIEnv *env, jobject clazz,
    jint lock, jstring idObj)
{
    if (idObj == NULL) {
        throw_NullPointerException(env, "id is null");
        return ;
    }
    const char *id = env->GetStringUTFChars(idObj, NULL);
    acquire_wake_lock(lock, id);
    env->ReleaseStringUTFChars(idObj, id);
}

static void acquireWakeLock(JNIEnv *env, jobject clazz,
    jint lock, jstring idObj)
{
    if (idObj == NULL) {
        throw_NullPointerException(env, "id is null");
        return ;
    }
    const char *id = env->GetStringUTFChars(idObj, NULL);
    acquire_wake_lock(lock, id);
    env->ReleaseStringUTFChars(idObj, id);
}
hardware/libhardware_legacy/power/power.c

... int acquire_wake_lock(int lock, const char* id) {
    initialize_fds();
    if (g_error) return g_error;
    int fd;
    if (lock == PARTIAL_WAKE_LOCK) {
        fd = g_fds[ACQUIRE_PARTIAL_WAKE_LOCK];
    } else {
        return EINVAL;
    }
    return write(fd, id, strlen(id));
}

static inline void initialize_fds(void) {
    if (g_initialized == 0) {
        if(open_file_descriptors(NEW_PATHS) < 0) {
            open_file_descriptors(OLD_PATHS);
            on_state = "wake";
            off_state = "standby";
        }
        g_initialized = 1;
    }
}

const char * const OLD_PATHS[] = {
    "/sys/android_power/acquire_partial_wake_lock",
    "/sys/android_power/release_wake_lock",
    "/sys/android_power/request_state"
};
const char * const NEW_PATHS[] = {
    "/sys/power/wake_lock",
    "/sys/power/wake_unlock",
    "/sys/power/state"
};

(Kernel interface changes in Android Cupcake)
Source code (Cupcake, linux-2.6.27)

- **kernel/kernel/power/userwake.c**
- **kernel/kernel/power/wakelock.c**

```c
static int power_suspend_late(
    struct platform_device *pdev,
    pm_message_t state)
{
    int ret =
        has_wake_lock(WAKE_LOCK_SUSPEND) ?
        -EAGAIN : 0;
    return ret;
}
```

```c
static struct platform_driver power_driver = {
    .driver.name = "power",
    .suspend_late = power_suspend_late,
};
static struct platform_device power_device = {
    .name = "power",
};
```

```c
static long has_wake_lock_locked(int type)
{
    struct wake_lock *lock, *n;
    long max_timeout = 0;
    BUG_ON(type >= WAKE_LOCK_TYPE_COUNT);
    list_for_each_entry_safe(lock, n,
        &active_wake_locks[type], link) {
        if (lock->flags & WAKE_LOCK_AUTO_EXPIRE) {
            long timeout = lock->expires - jiffies;
            if (timeout <= 0)
                expire_wake_lock(lock);
            else if (timeout > max_timeout)
                max_timeout = timeout;
        } else
            return -1;
    }
    return max_timeout;
}
```

```c
long has_wake_lock(int type)
{
    long ret;
    unsigned long irqflags;
    spin_lock_irqsave(&list_lock, irqflags);
    ret = has_wake_lock_locked(type);
    spin_unlock_irqrestore(&list_lock, irqflags);
    return ret;
}
```
static int __init wakelocks_init(void)
{
    int ret;
    int i;

    for (i = 0; i < ARRAY_SIZE(active_wake_locks); i++)
        INIT_LIST_HEAD(&active_wake_locks[i]);

    wake_lock_init(&main_wake_lock, WAKE_LOCK_SUSPEND, "main");
    wake_lock(&main_wake_lock);
    wake_lock_init(&unknown_wakeup, WAKE_LOCK_SUSPEND, "unknown_wakeups");

    ret = platform_device_register(&power_device);
    if (ret) {
        pr_err("wakelocks_init: platform_device_register failed\n");
        goto err_platform_device_register;
    }

    ret = platform_driver_register(&power_driver);
    if (ret) {
        pr_err("wakelocks_init: platform_driver_register failed\n");
        goto err_platform_driver_register;
    }

    suspend_work_queue = create_singlethread_workqueue("suspend");
    if (suspend_work_queue == NULL) {
        ret = -ENOMEM;
        goto err_suspend_work_queue;
    }

}
Use "`grep -r acquire_wake_lock`" to discover.

- frameworks/base/libs/ui/EventHub.cpp
  - EventHub::EventHub(), EventHub::~EventHub(), EventHub::getEvent()
- hardware/ril/libril/ril.cpp
  - RIL_onUnsolicitedResponse(),
- system/wlan/ti/sta_dk_4_0_4_32/CUDK/tiwlan_loader/tiwlan_loader.c
  - start_cli()
Use "grep -ri power" to discover.

- base/core/jni/android_net_wifi_Wifi.cpp
  - android.net.wifi.WifiNative.setPowerModeCommand
    (android_net_wifi_setPowerModeCommand)
- base/core/java/com/android/internal/os/BatteryStats.java
- base/core/java/com/android/internal/os/BatteryStatsImpl.java
- base/core/java/com/android/os/PowerManager.java
- base/core/java/com/android/os/LocalPowerManager.java
- base/core/java/com/android/os/Power.java
- base/core/java/com/android/app/ApplicationContext.java
- base/core/java/android/bluetooth/ScoSocket.java
- base/core/java/android/bluetooth/HeadsetBase.java
- base/core/media/java/android/media/MediaPlayer.java
- base/core/media/java/android/media/AsyncPlayer.java
Use "grep -ri power" to discover.

- base/telephony/java/com/android/internal/telephony/gsm/GSMConnection.java
- base/telephony/java/com/android/internal/telephony/gsm/RIL.java
- base/telephony/java/com/android/internal/telephony/gsm/GSMPhone.java
- base/services/jni/com_android_server_BatteryService.cpp
- base/services/java/com/android/server/am/ActivityManagerService.java
- base/services/java/com/android/server/SystemServer.java
- base/services/java/com/android/server/BatteryService.java
- base/services/java/com/android/server/AlarmManagerService.java
- base/services/java/com/android/server/LocationManagerService.java
- base/services/java/com/android/server/HardwareService.java
- base/services/java/com/android/server/PowerManagerService.java
Use "grep -ri power" to discover.

- system/core/toolbox/powerd.c
- system/core/toolbox/alarm.c
- system/core/vold/mmc.c
- system/core/vold/uevent.c
Introduction to Linux Power Management

Concepts behind Android Power Management
  - Basically, it is the slim wrapper for Linux Power Management

Design and Implementation

Room for Improvements
So, Android PM is simple

- Key concept is “Wake Locks”, which is simple and portable.
- The implementation should be introspected.

- If there are no active wake locks, CPU will be turned off.
- If there is are partial wake locks, screen and keyboard will be turned off.

- Nowadays, CPU can enter into more states for power saving and usability purpose! → applied in modern SoC
Interface to Physical World

The Real World
- Temperature
- Pressure
- Position
- Speed
- Flow
- Humidity
- Sound
- Light

Signal Conditioning

Analog Signal Conversion to Digital

Power Management

Digital Signal Processor

Interface
- Clocks & Timers
Power management aims to improve battery life of equipment by minimizing power consumption while guaranteeing expected system performance

**Active power consumption** occurs while some processing is on-going
- Dynamic power consumption (transistor switching) + Leakage consumption

**Static (also Standby or Idle) power consumption** occurs when limited or no processing is on-going and the system is waiting for a wakeup event
- Very limited dynamic power consumption + Leakage consumption
- Managed by
  - Dynamic Voltage & Frequency Scaling (DVFS)
  - Adaptive Voltage Scaling (AVS)
  - Dynamic Power Switching (DPS)

On OMAP35xx, power management is handled by the Power, Reset and Clock Management (PRCM) module
Adaptive Voltage Scaling

- Silicon manufacturing process yields a distribution of performance capability
- For a given frequency requirement:
  - Devices on hot/strong/fast end of distribution can meet this at a lower voltage
  - Devices on cold/weak/slow end of distribution need higher voltage
- Simple system will set the higher voltage for operating all devices
- Smarter system will adapt operating voltage per device.
Linux PM Mechanisms

- cpuidle
  - Generic framework for supporting software-controlled idle processor power management
  - Hardware specific drivers
  - Various governing for the state transition decisions

- Latency and power management
  - Framework for expressing latency constraints, and make sure that they are taken into account for power management decisions

- Needs to be integrated into Android Partial Wake Locks
Integration is the key to the most power-saving system.
Reference

- PDK :: Android – Power Management
  - http://www.netmite.com/android/mydroid/development/pdk/docs/power_management.html

- Android Cupcake source code

- BeagleBoard and its Linux support
  - http://elinux.org/BeagleBoard

- class PowerManager

- Free-Electrons
  - http://free-electrons.com/training

- CELF's power management specification
  - http://elinux.org/Power_Management