Power Management from Linux Kernel to Android

Matt Hsu & Jim Huang (jserv)
from 0xlab

June 19, 2009

Rights to copy

contact@0xlab.org

Corrections, suggestions, contributions and translations are welcome!

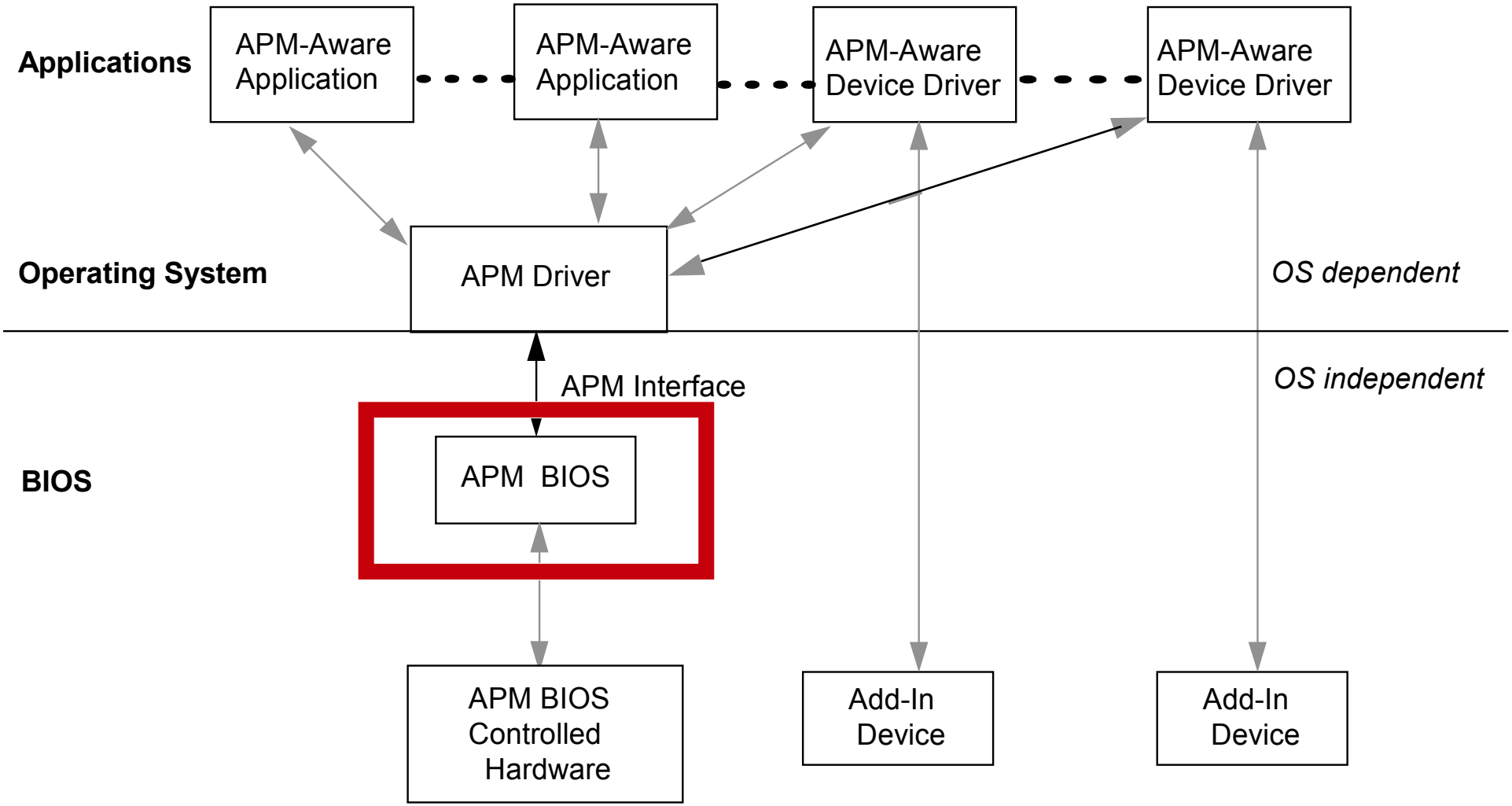
Agenda

- Introduction to Linux Power Management
- Concepts behind Android Power Management
- Design and Implementation
- Room for Improvements

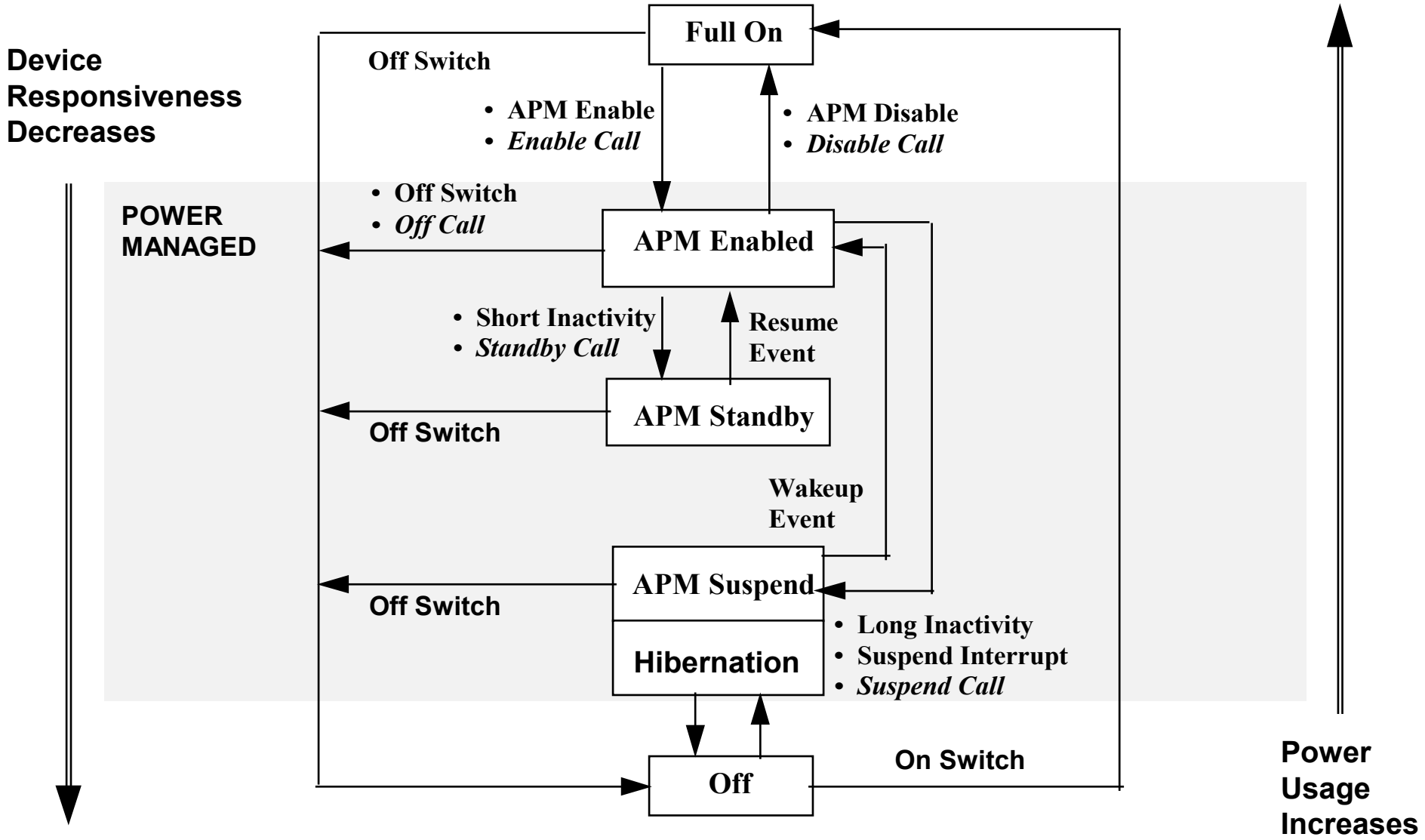
- **Introduction to Linux Power Management**
  - **Goal of Power Management**
  - **APM vs. ACPI**
  - **APM emulation**
  - **No silver bullet**
  - **Manage power in different power state**
  - **Lighthouse in the sea**
- Concepts behind Android Power Management
- Design and Implementation
- Room for Improvements

Prolong  Life

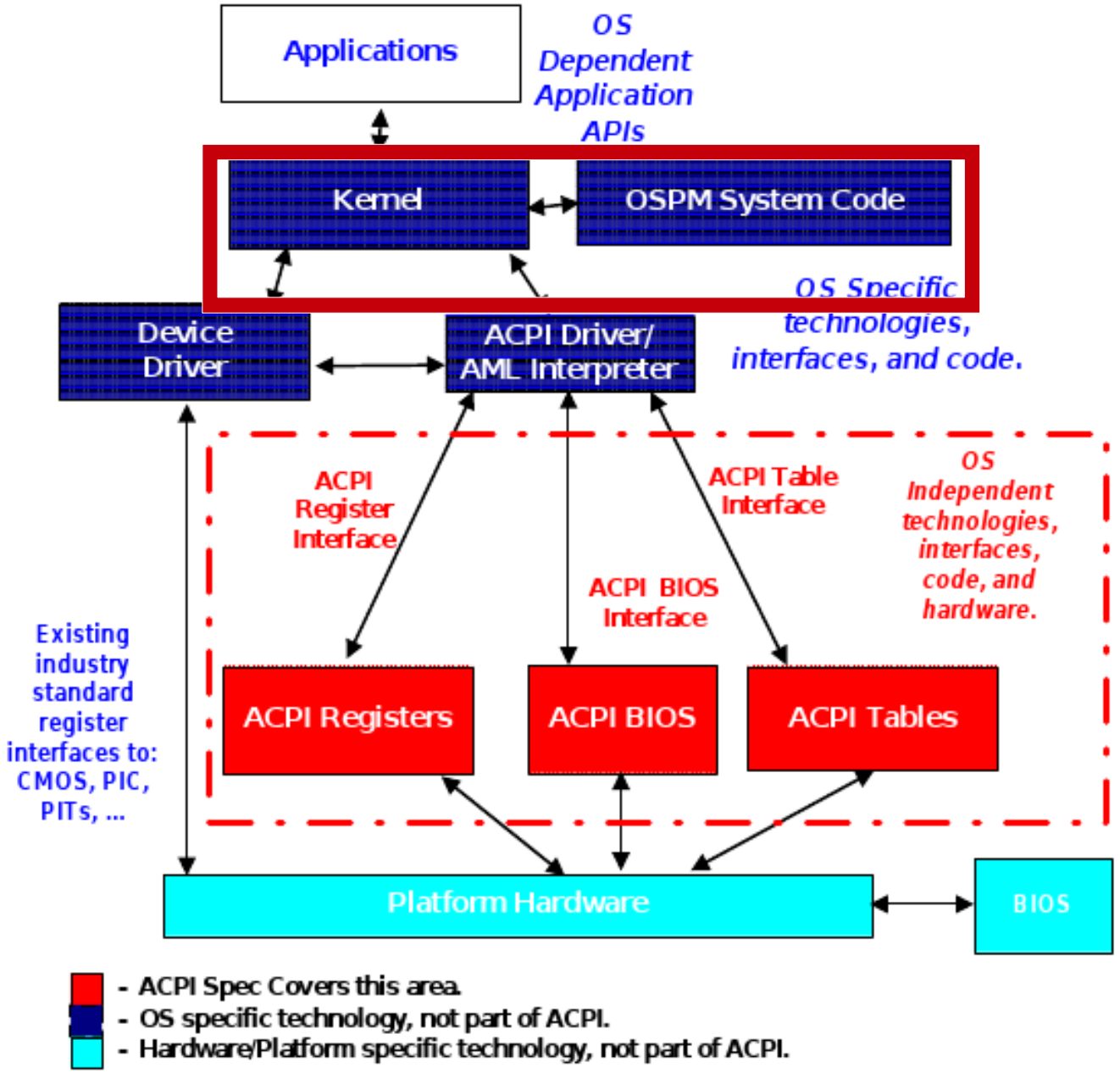
Advanced Power Management



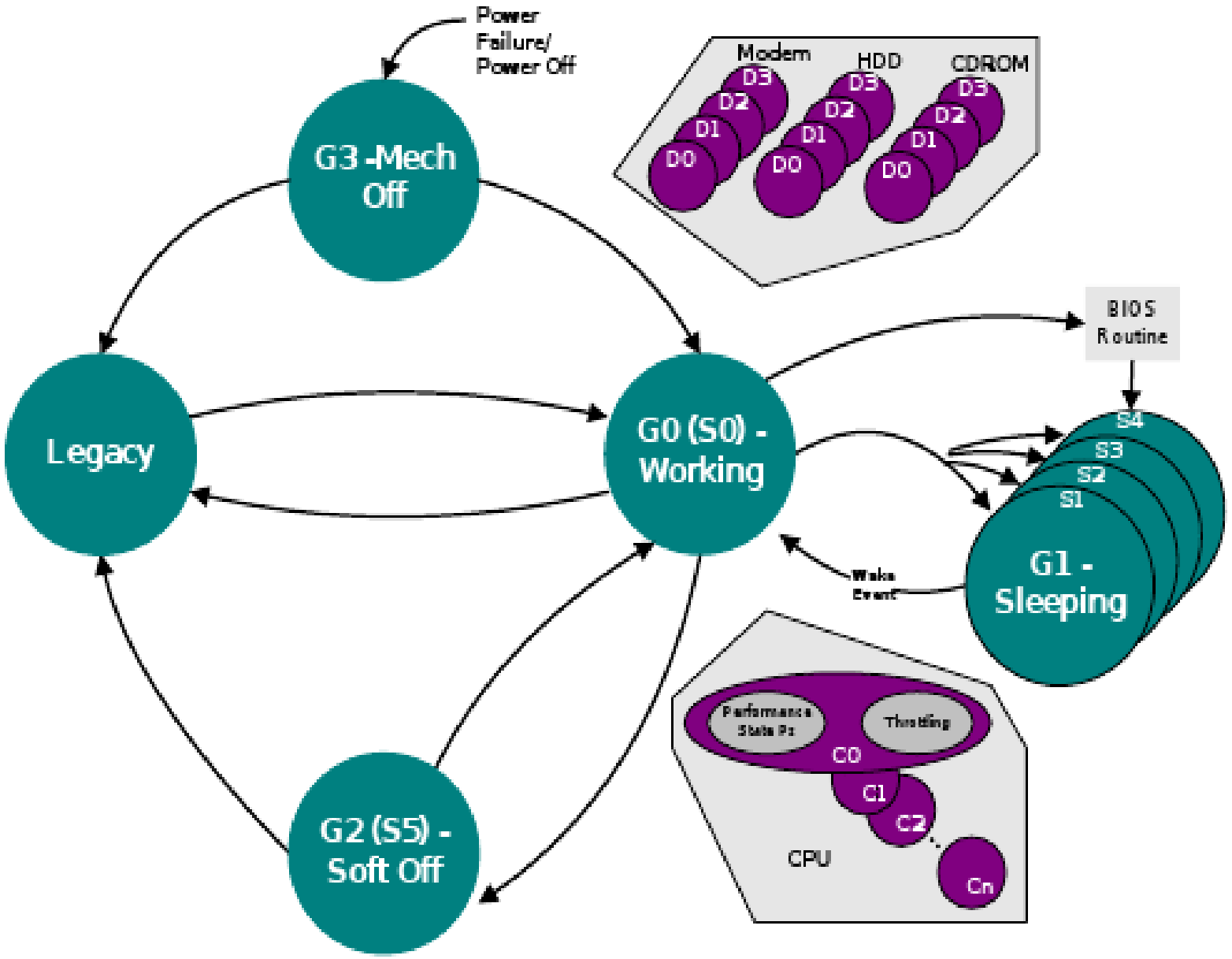
APM power state



Advanced Configuration and Power Interface



ACPI Power State Transition



APM vs. ACPI

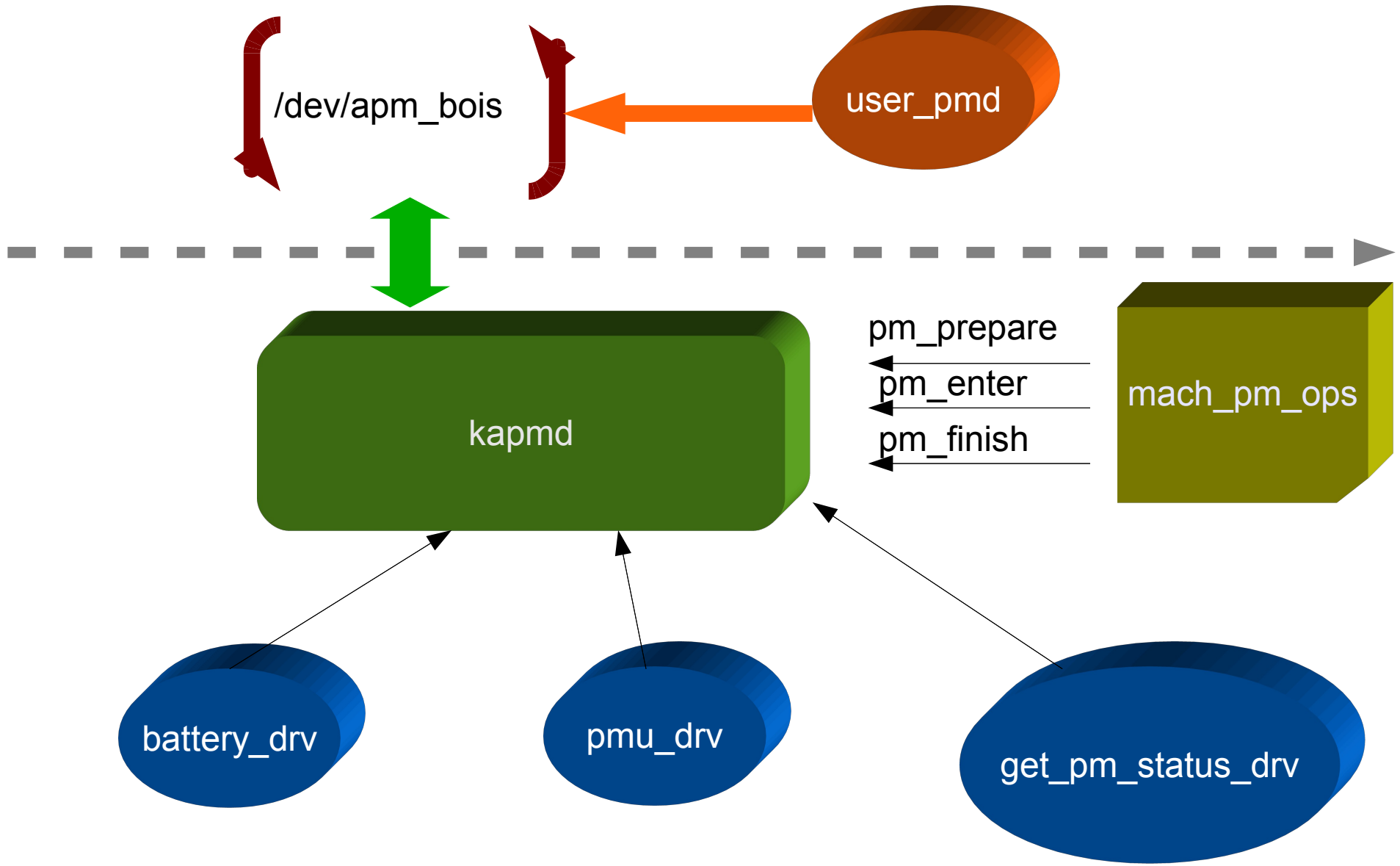
APM

- 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

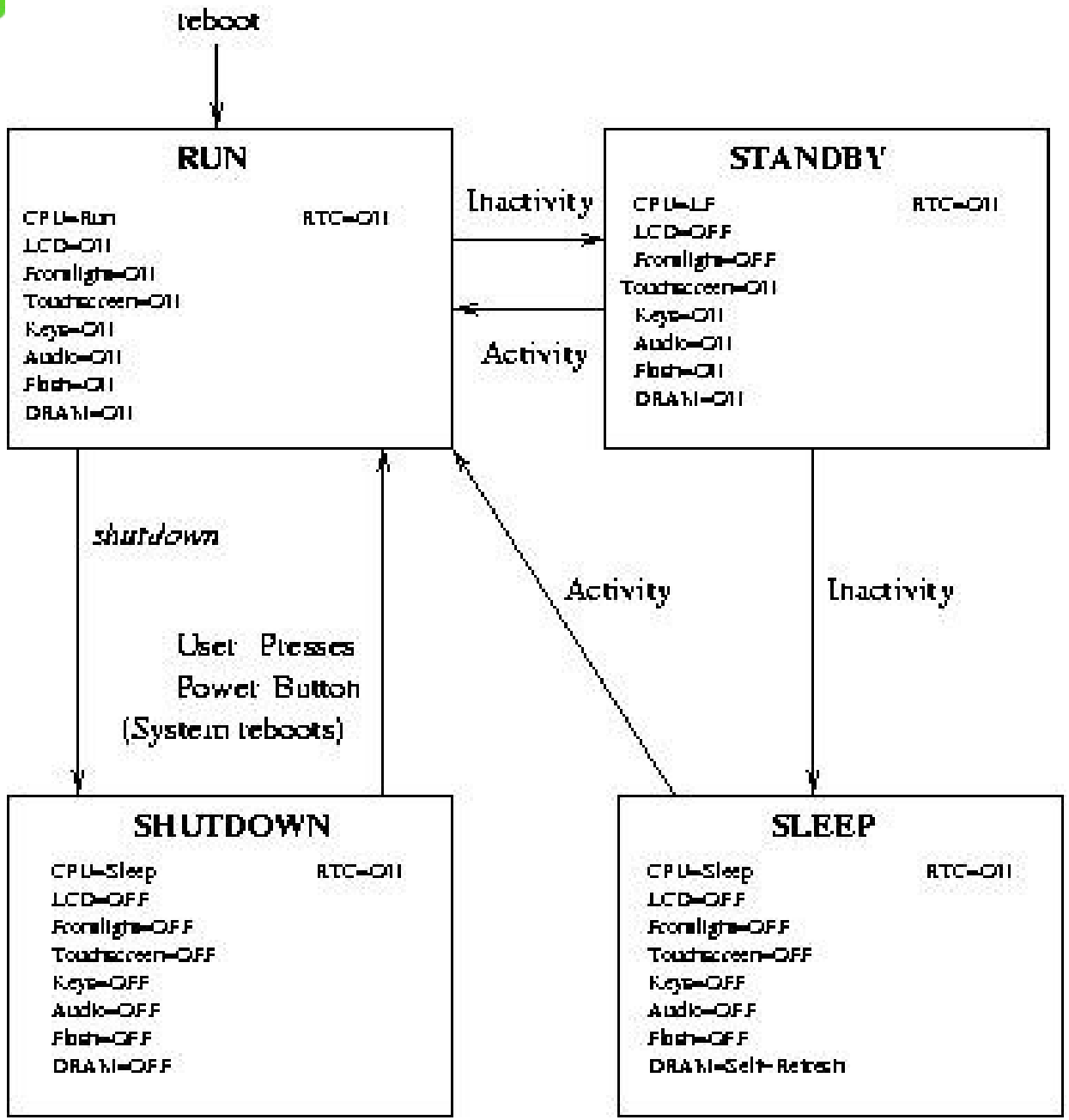
- Control divided between BIOS and OS
- Decisions managed through the OS
- Enables sophisticated power policies for general-purpose computers with standard usage patterns and hardware
- No knowledge of device-specific scenarios (e.g. need to provide predictable response times or to respond to critical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

APM emulation



No silver bullet
for building most power-saving
mobile system.

Power State Transition



Runtime / Standby

- 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

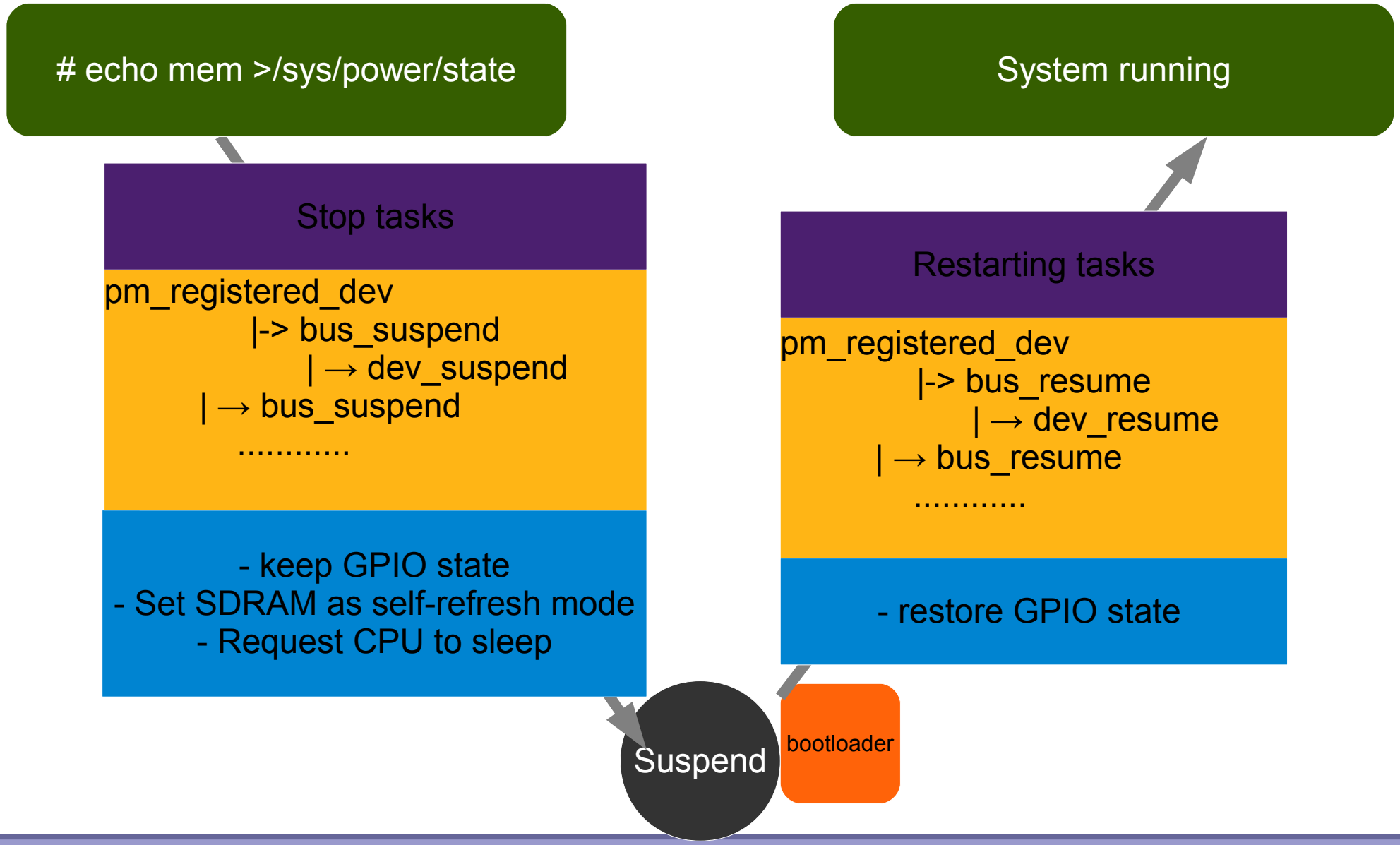
Suspend/Sleep Mode

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

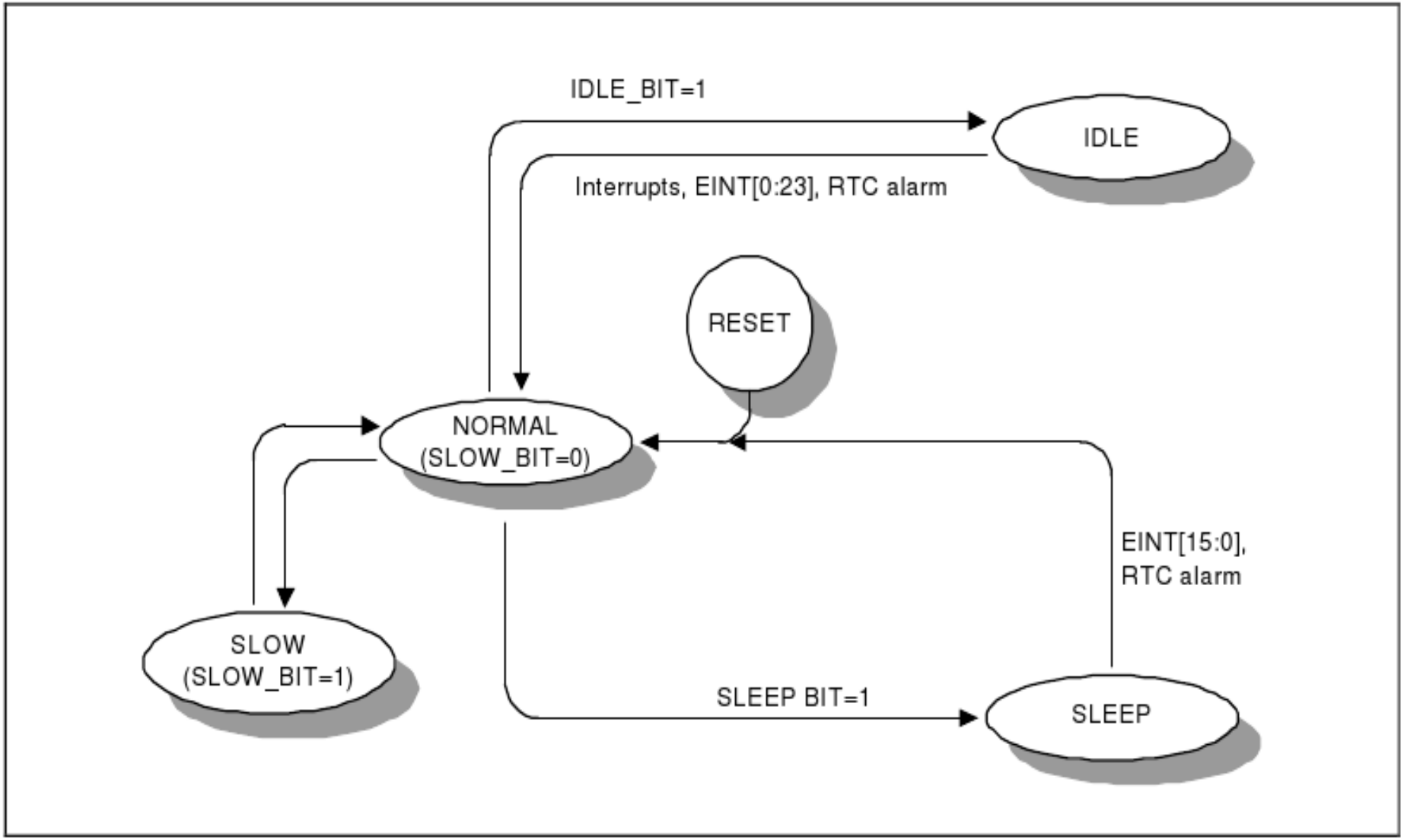
- STR: A technique by which systems preserve state in RAM during suspend and restore system state from RAM upon resume

- STR is suitable for mobile device if hardware supports this function

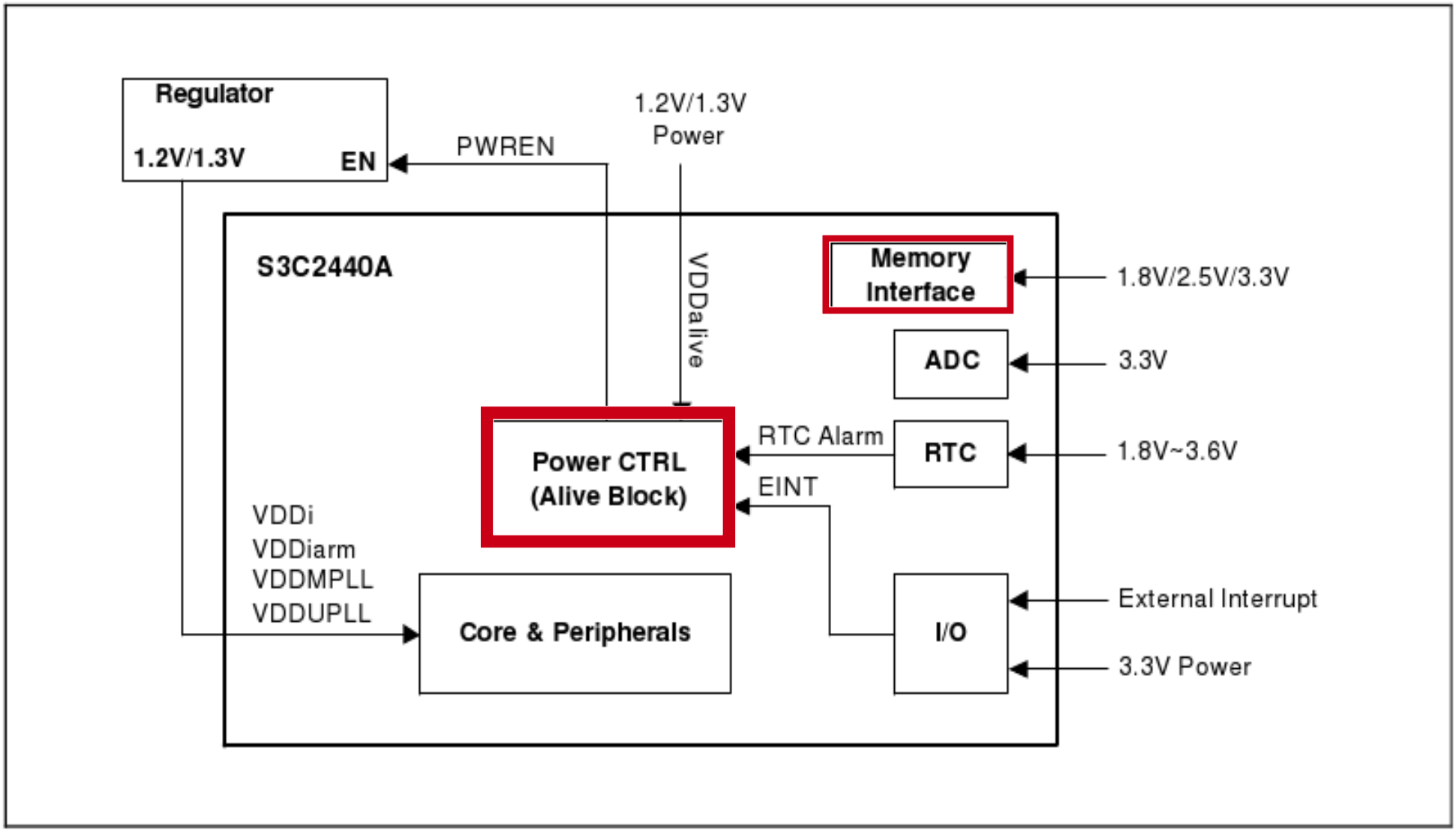
Suspend/Resume Control Path



Practical Example -S3C244x (1)



Practical Example -S3C244x (2)

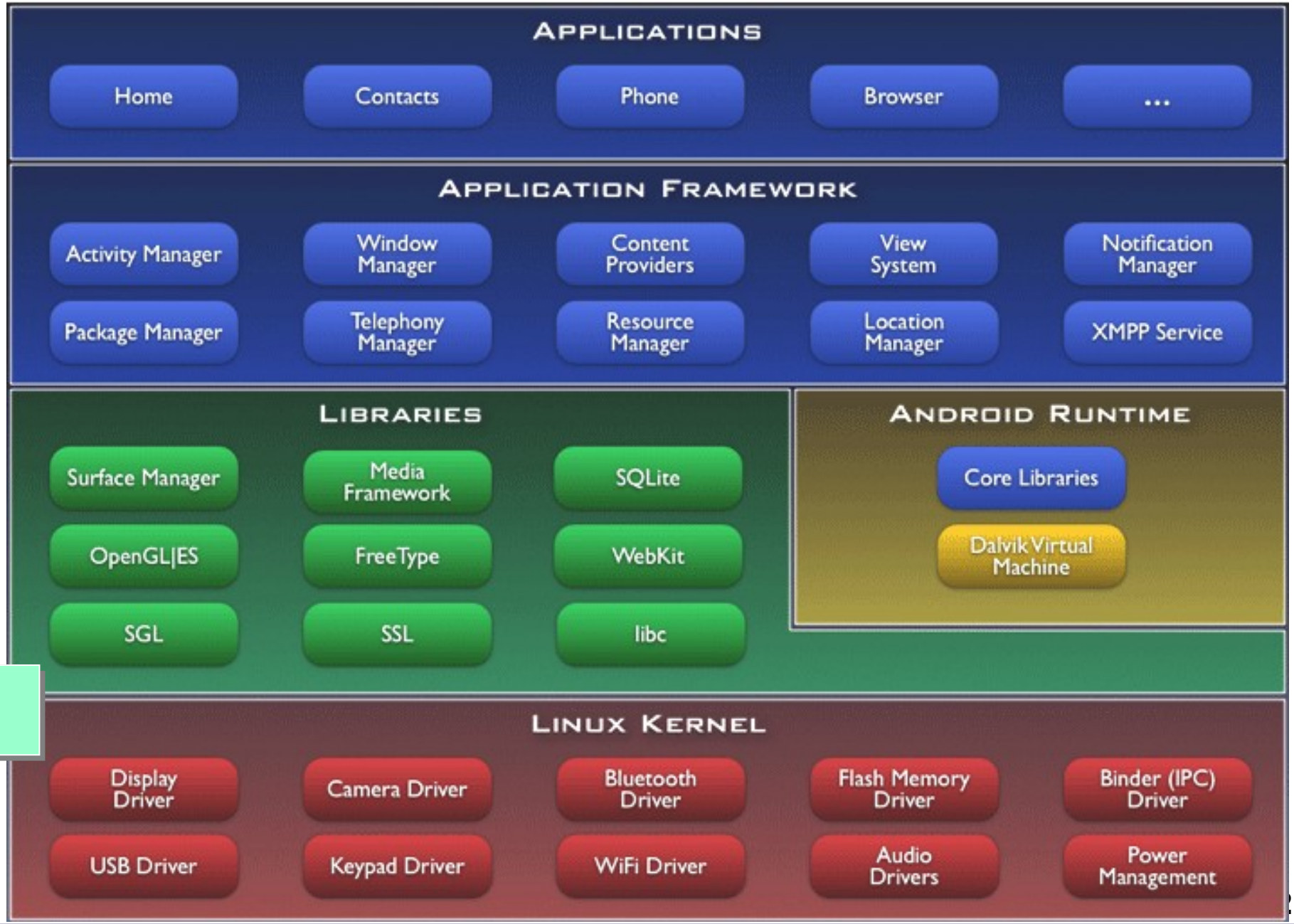


A ship on the beach is a lighthouse to the sea

- 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

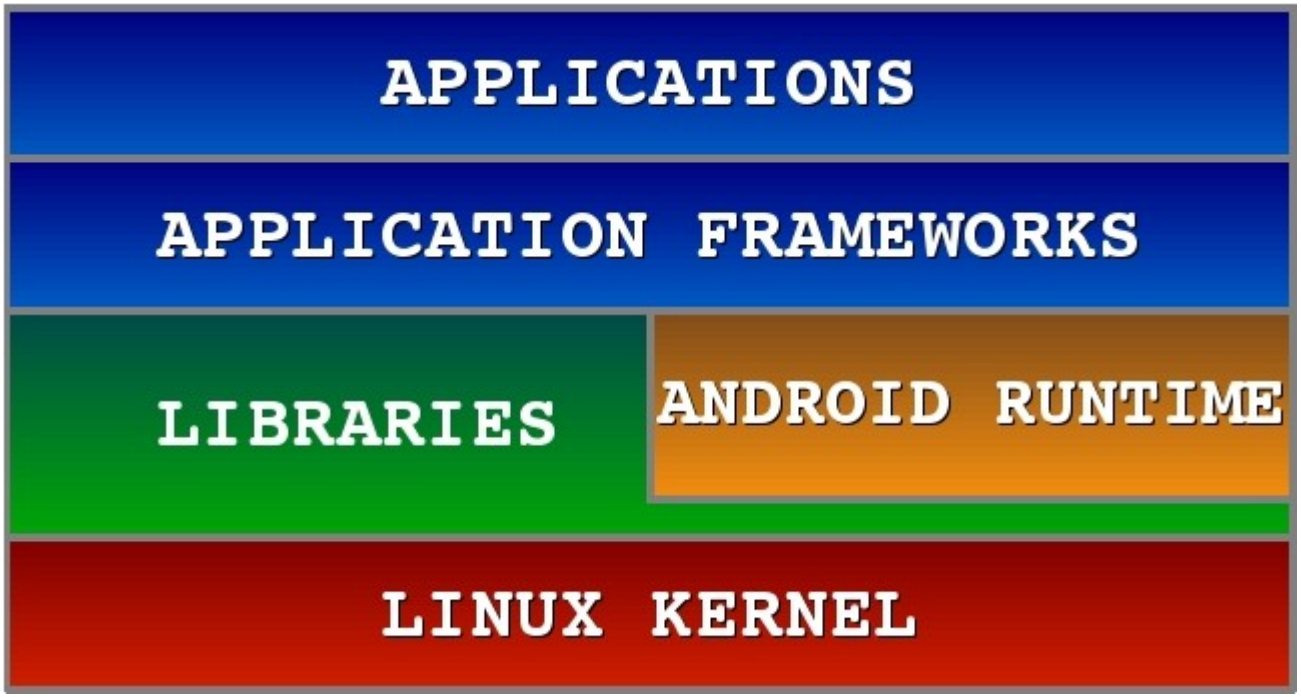
Where is Power Manager?



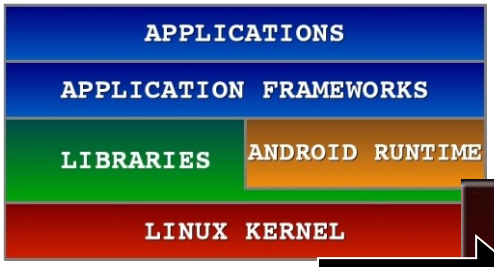
PM?

In fact, power ghost exists everywhere

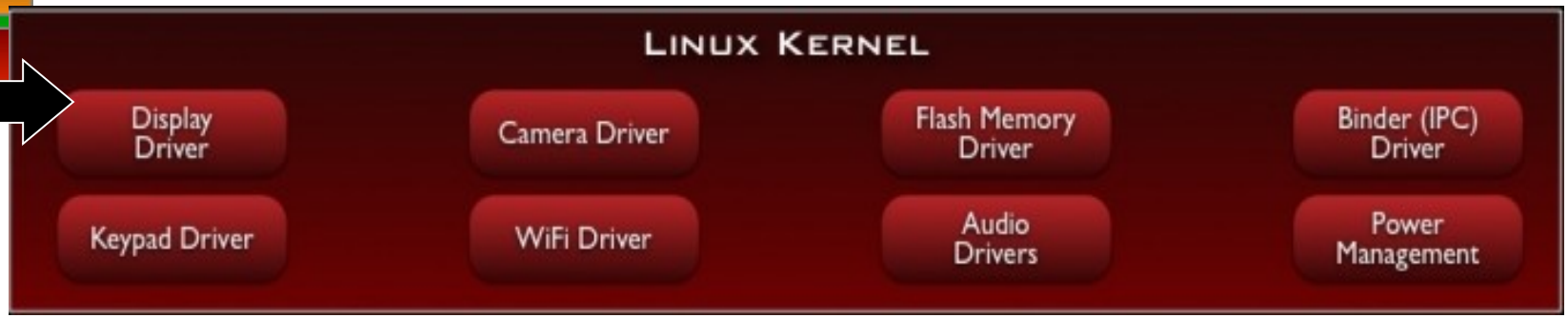
- Let "**`grep -ri power`**" tell you about the details!
- It is layered into several components, but implementation involves in some hacks.
- Check: CONFIG_PM, sysfs, device drivers, libhardware_legacy, libril, init.rc, powerd, alarm, vold, JNI, PowerManager, BatteryManager, ...



Base: Linux Kernel



Directory: arch/arm/mach-omap2/
board-omap3beagle.c
gpmc.c
pm.[ch]

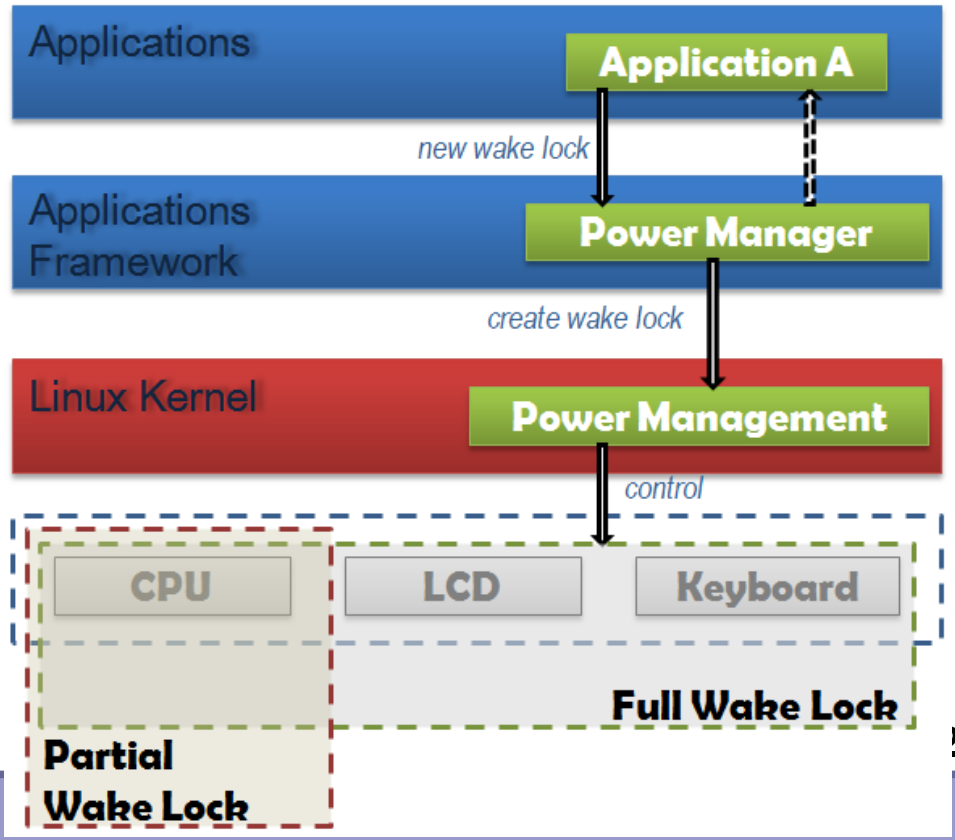


- Android does rely on Linux Kernel 2.6 for core system services
  - Memory/Process Management
  - Device Driver Model
  - sysfs, kobject/uevent, netlink
- Android Kernel extensions
  - Binder
  - android_power
    - /sys/android_power/, /sys/power/

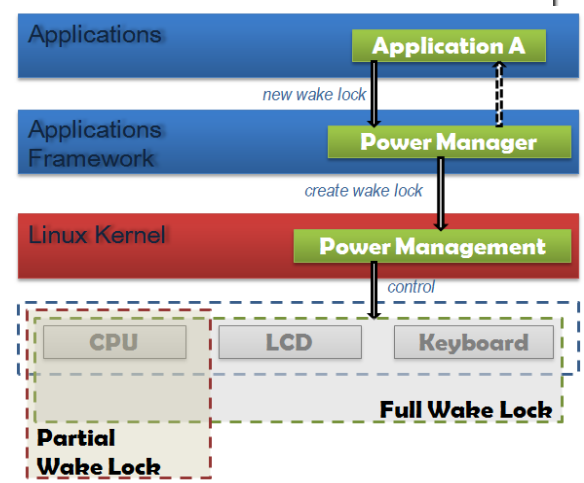
Key Idea: Android attempts to provide an abstraction layer between hardware and the related software stack.

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".



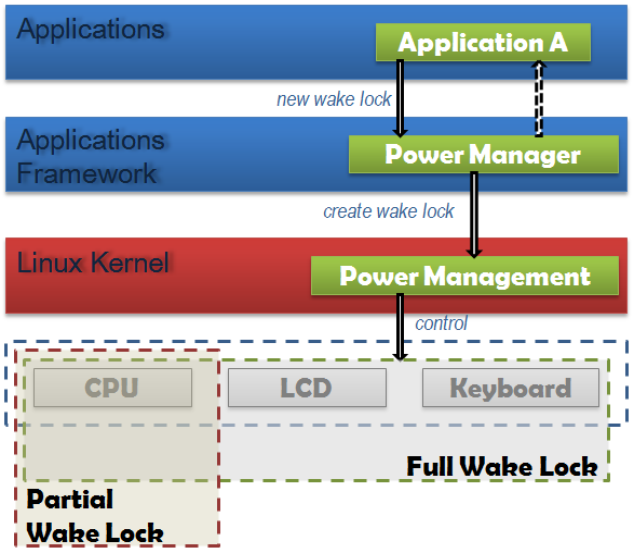
- If there are no active wake locks, CPU will be turned off.
- If there is are partial wake locks, screen and keyboard will be turned off.


Applications
Application A
new wake lock
Applications Framework
Power Manager
create wake lock
Linux Kernel
Power Management
control
CPU
LCD
Keyboard
Full Wake Lock
Partial Wake Lock

Applications
Application A
Application B
Application C
WI = newWakeLock(…);
WI.acquire();
WI.release();
Applications Framework
PowerManager
Android.os.PowerManager
Power
Android.os.Power
PowerManagerService
Andorid.server.PowerManagerService
JNI
Libraries (user space)
Core Libraries
X
Power
/lib/hardware/power.c
Linux Kernel
Linux Drivers
Android Power Management
/drivers/android/power.c
Android_register_early_suspend()
Android_register_early_resume()
Linux Power Management

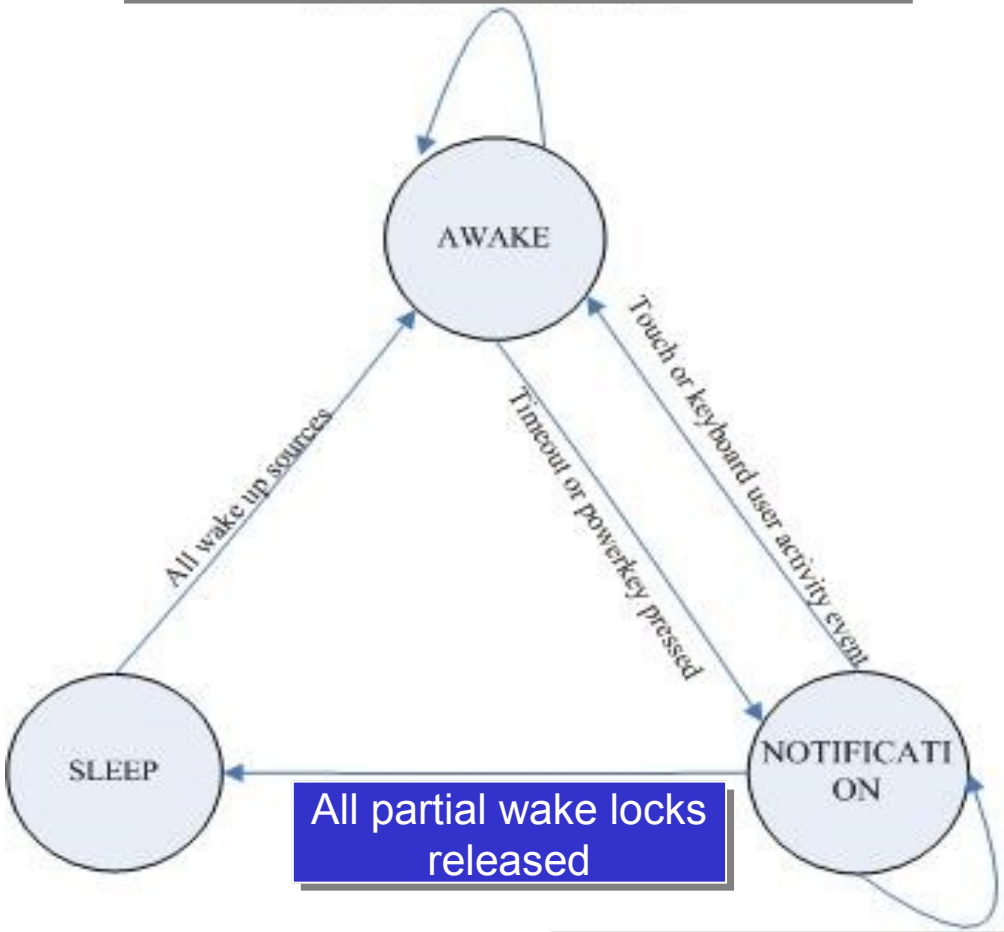
Types of Wake Locks

- 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.



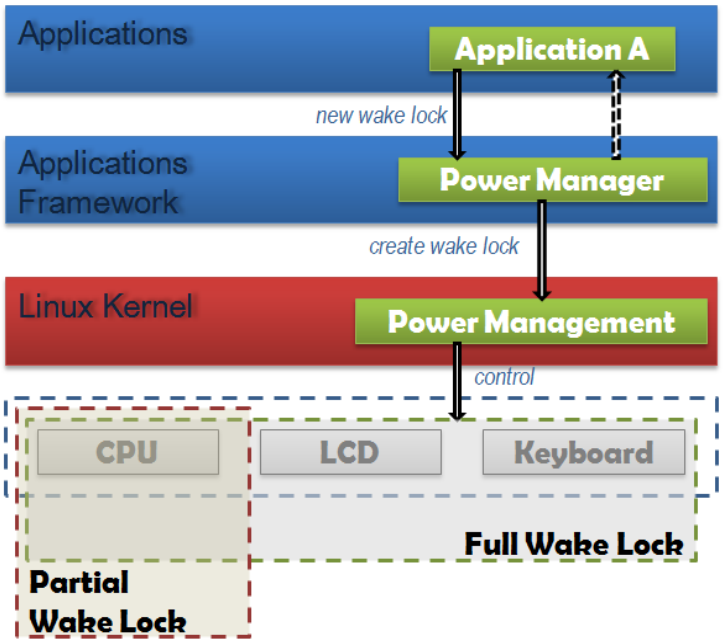
PM State Machine

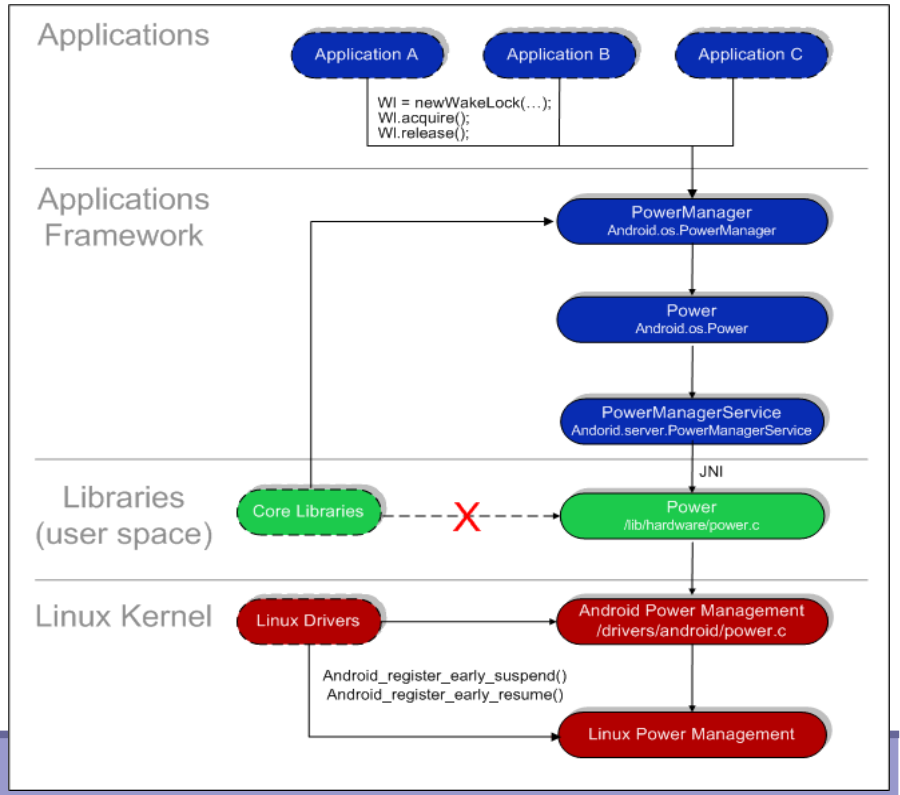
Touchscreen or keyboard user activity event or full wake locks acquired.



All partial wake locks released

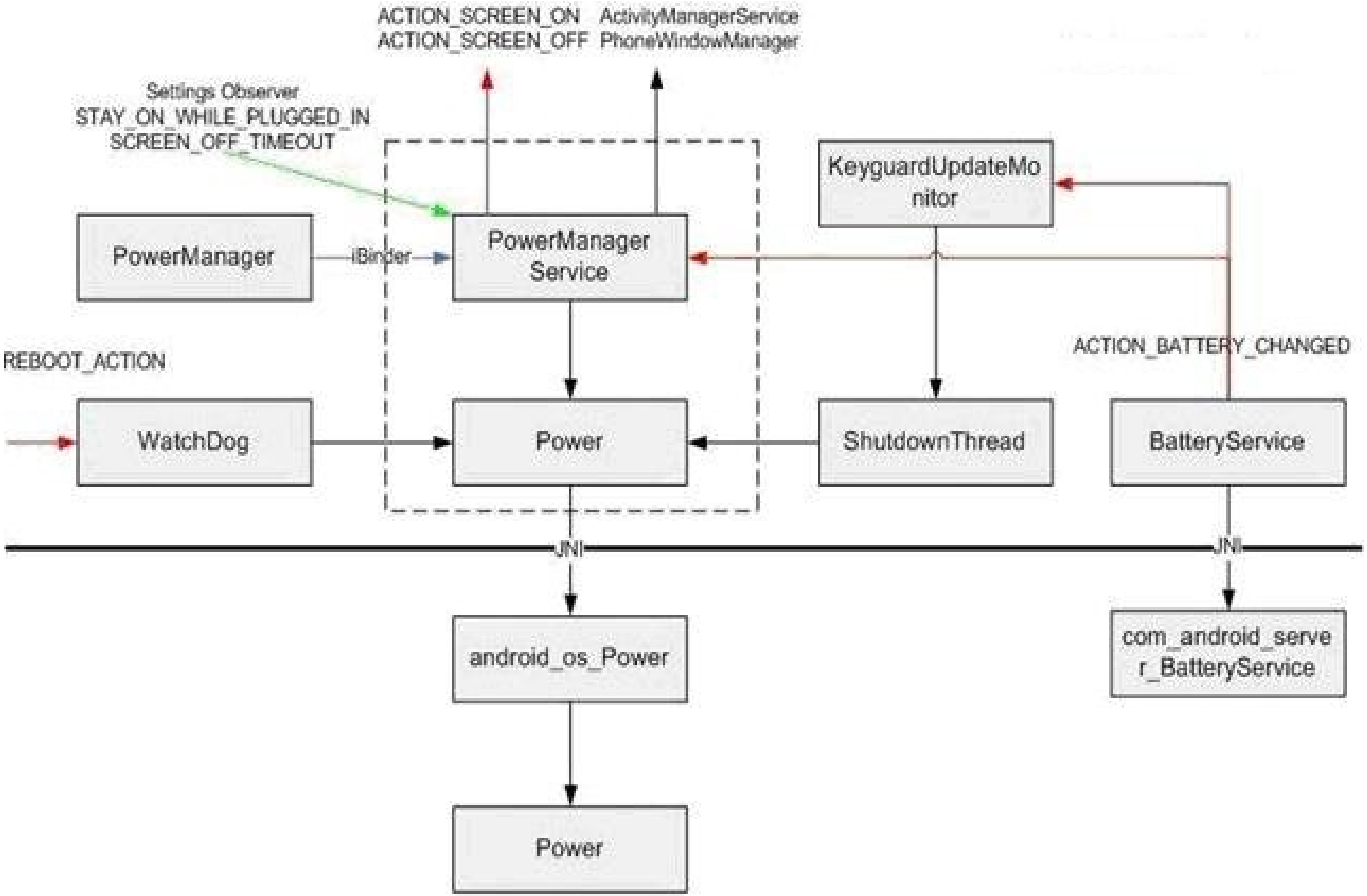
Partial wake locks acquired





- 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

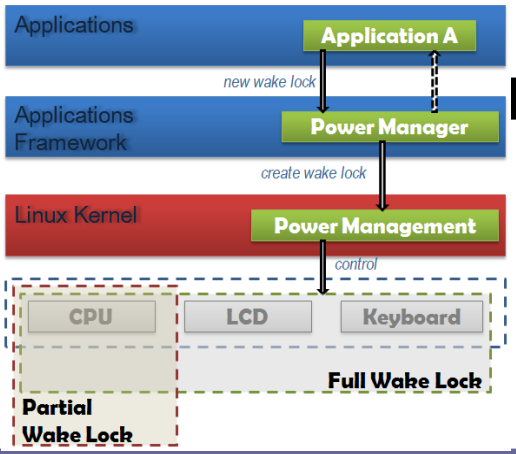
Design and Implementation

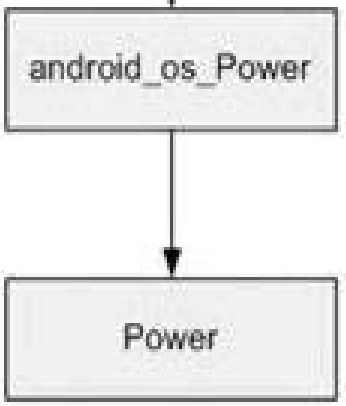


android_os_Power

frameworks/base/core/jni/android_os_Power.cpp

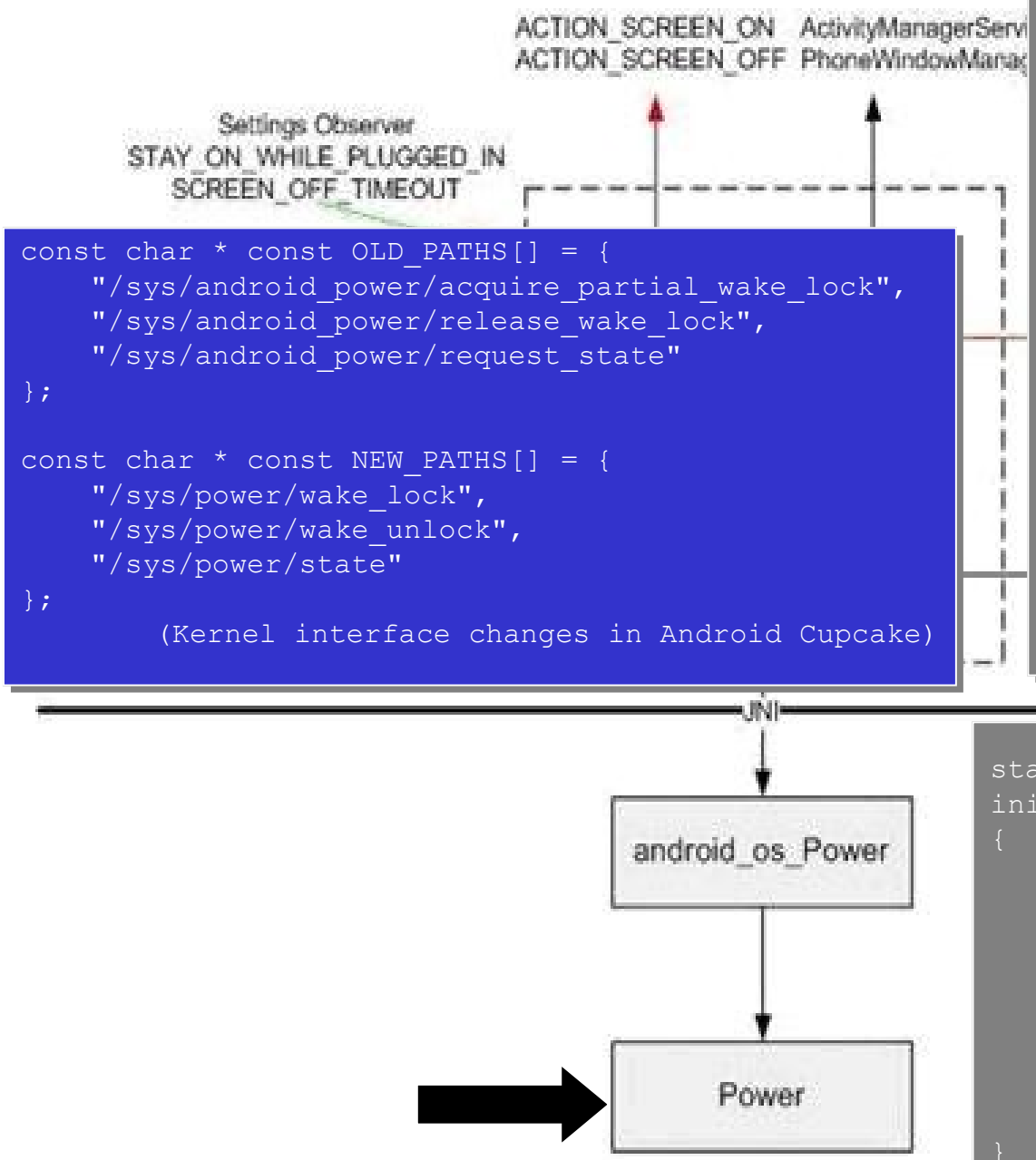
```
...
static JNINativeMethod method_table[] = {
    { "acquireWakeLock", "(ILjava/lang/String;)V", (void*)acquireWakeLock },
    { "releaseWakeLock", "(Ljava/lang/String;)V", (void*)releaseWakeLock },
    { "setLastUserActivityTimeout", "(J)I", (void*)setLastUserActivityTimeout },
    { "setLightBrightness", "(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);
}
```

Power




```
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)

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;
    }
}
```

Android PM Kernel APIs

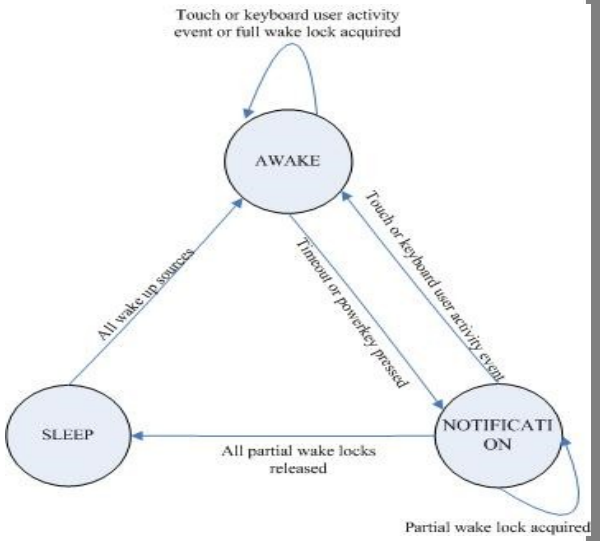
- Source code (Cupcake, linux-2.6.27)
  - kernel/kernel/power/userwake.c
  - kernel/kernel/power/wakelock.c

```
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;
}

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

```
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;
}

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;
}
```

Android PM Kernel APIs

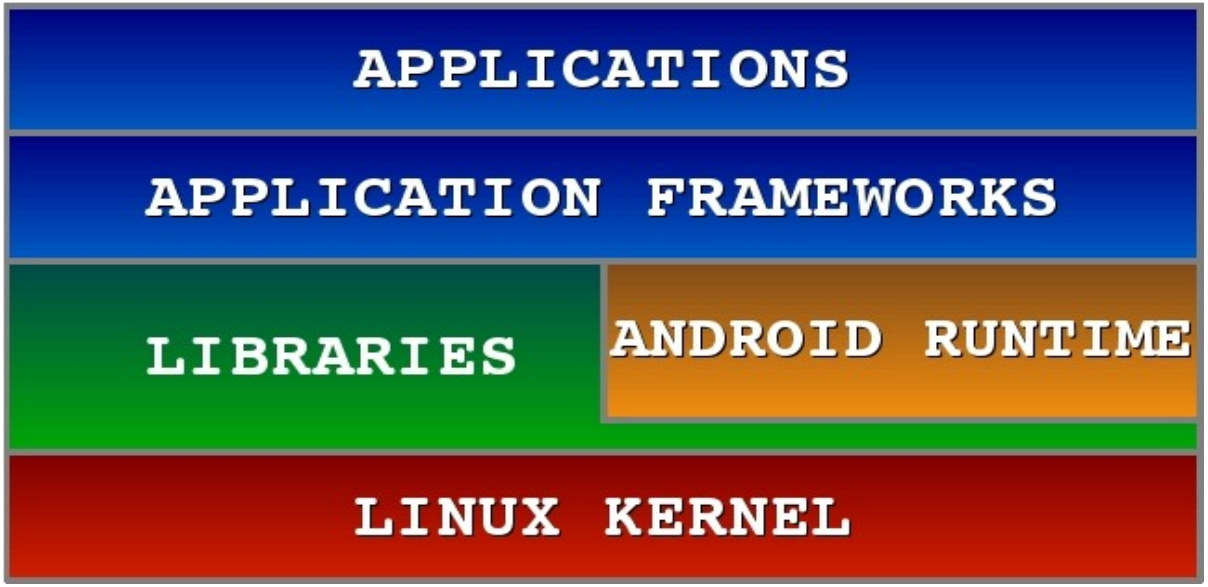
- kernel/kernel/power/wakelock.c




```
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;
	}
```

Back to userspace

- 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()



Back to userspace

- 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

Back to userspace

- 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

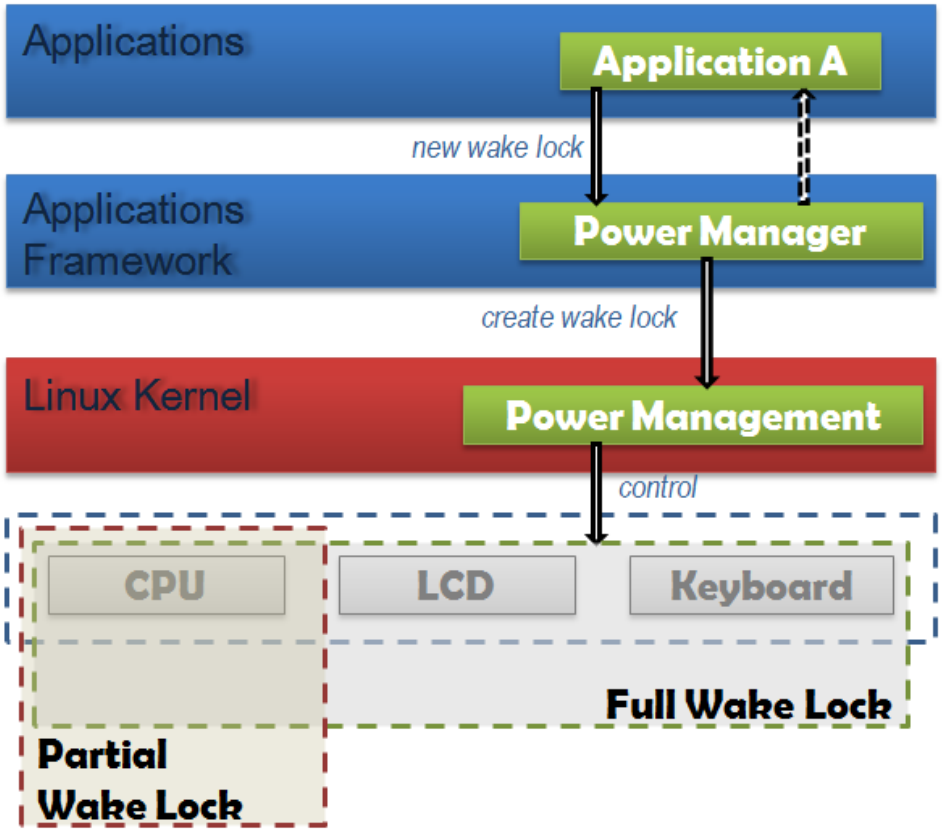
Back to userspace

- 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

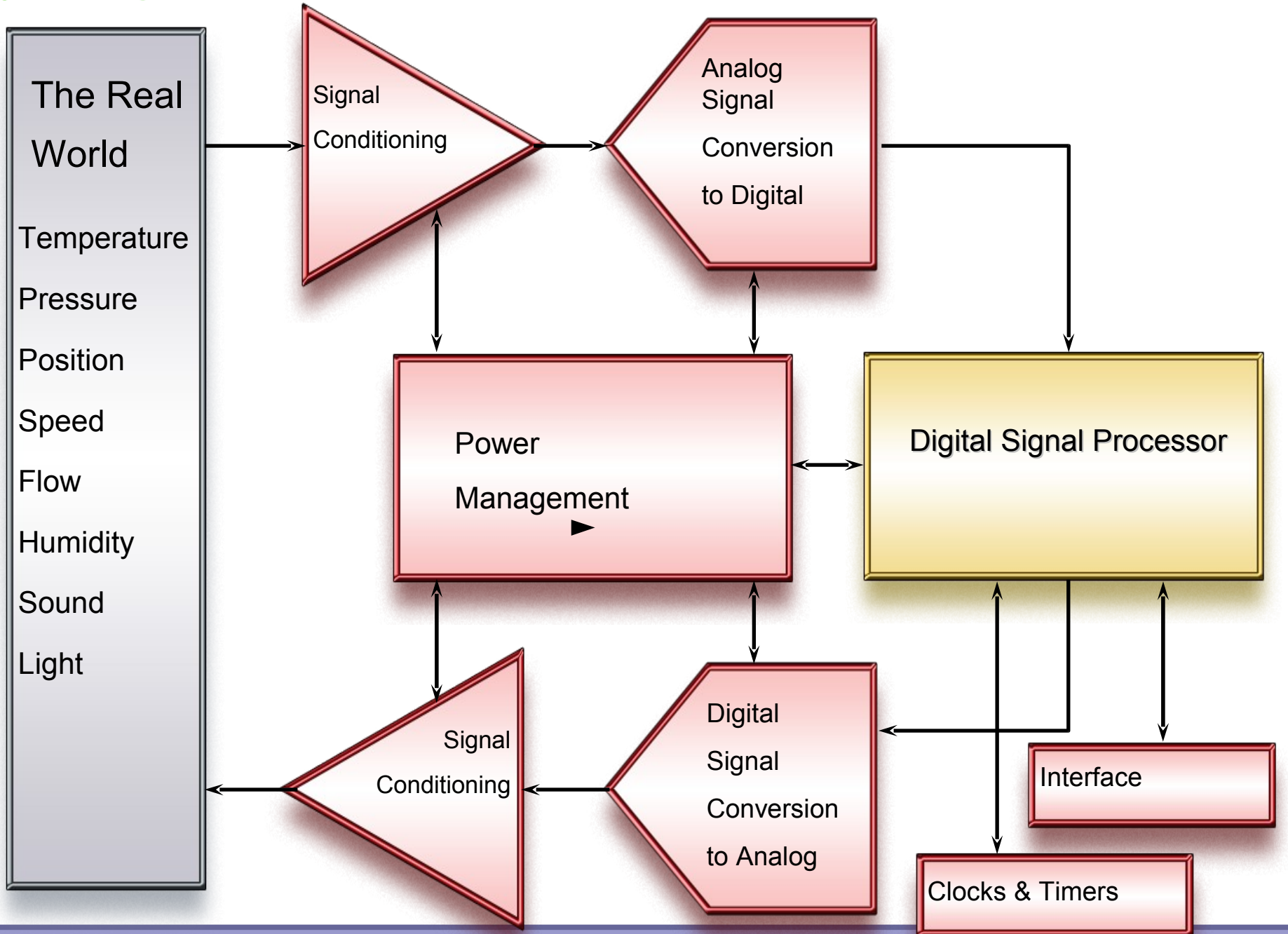
- Key concept is “Wake Locks”, which is simple and portable.
- The implementation should be introspected.



- 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



Advanced Power Management

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

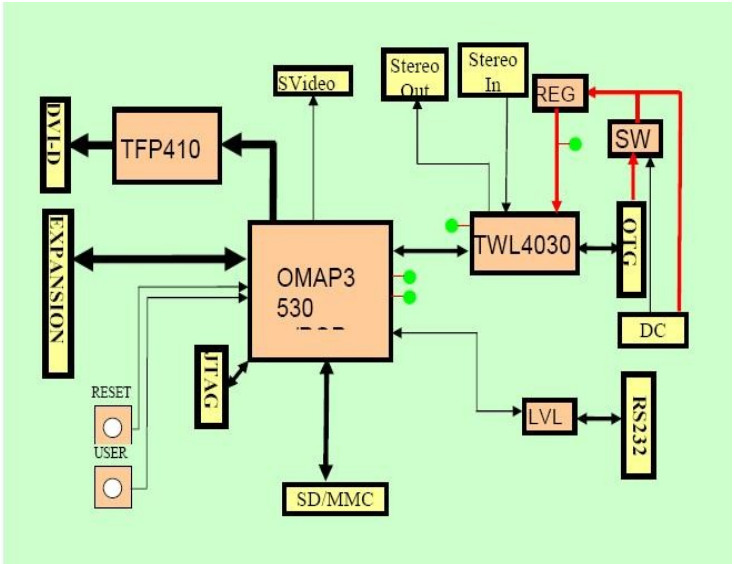
Static (also Standby or Idle) power consumption occurs when limited or no processing is on-going and the system is waiting for a wakeup event

- Very limited dynamic power consumption + Leakage consumption
- Managed by
  - Dynamic Voltage & Frequency Scaling (DVFS)
  - Adaptive Voltage Scaling (AVS)
  - Dynamic Power Switching (DPS)

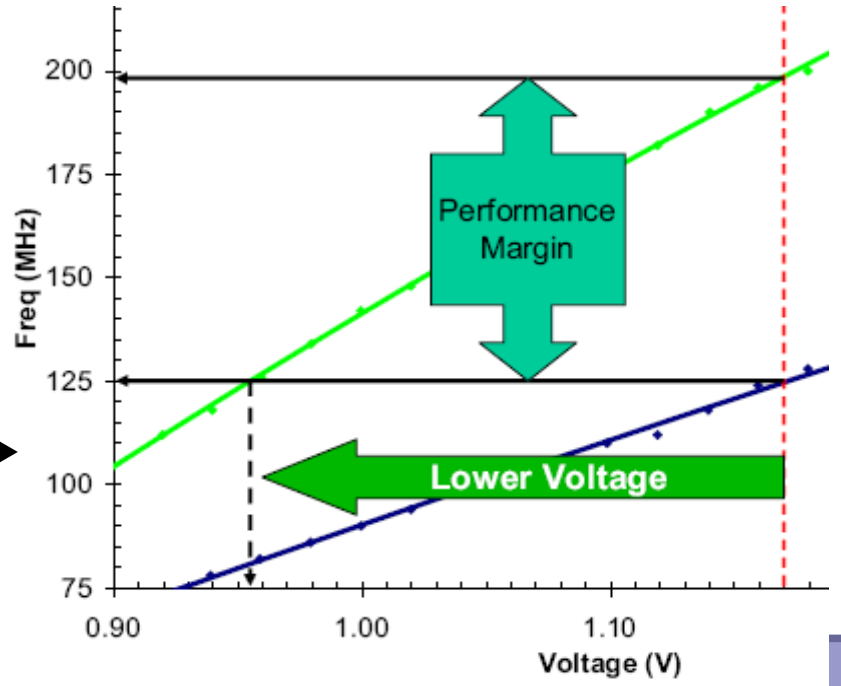
On OMAP35xx, power management is handled by the Power, Reset and Clock Management (PRCM) module

Adaptive Voltage Scaling

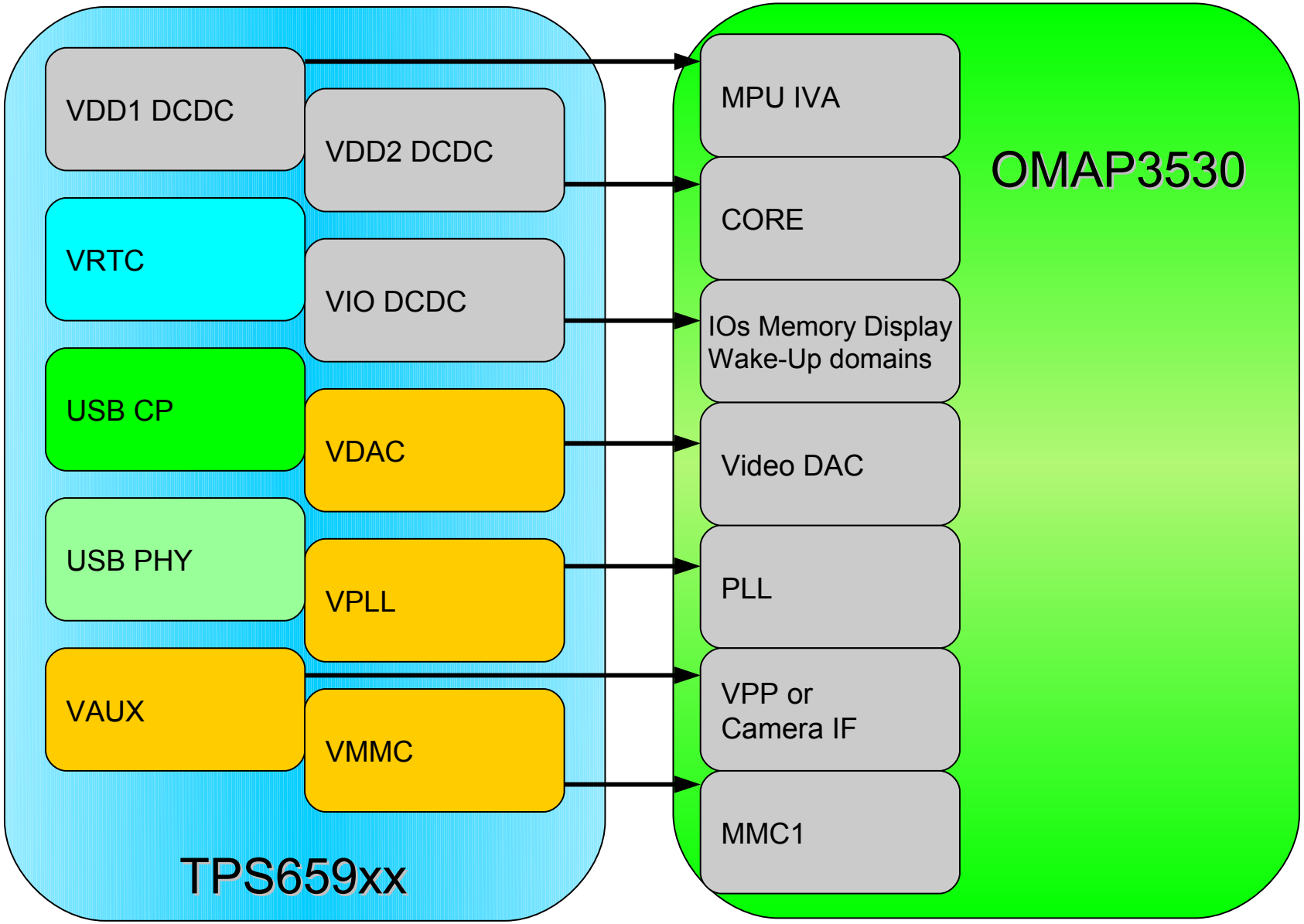
- Silicon manufacturing process yields a distribution of performance capability
- For a given frequency requirement:
  - Devices on hot/strong/fast end of distribution can meet this at a lower voltage
  - Devices on cold/weak/slow end of distribution need higher voltage
- Simple system will set the higher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



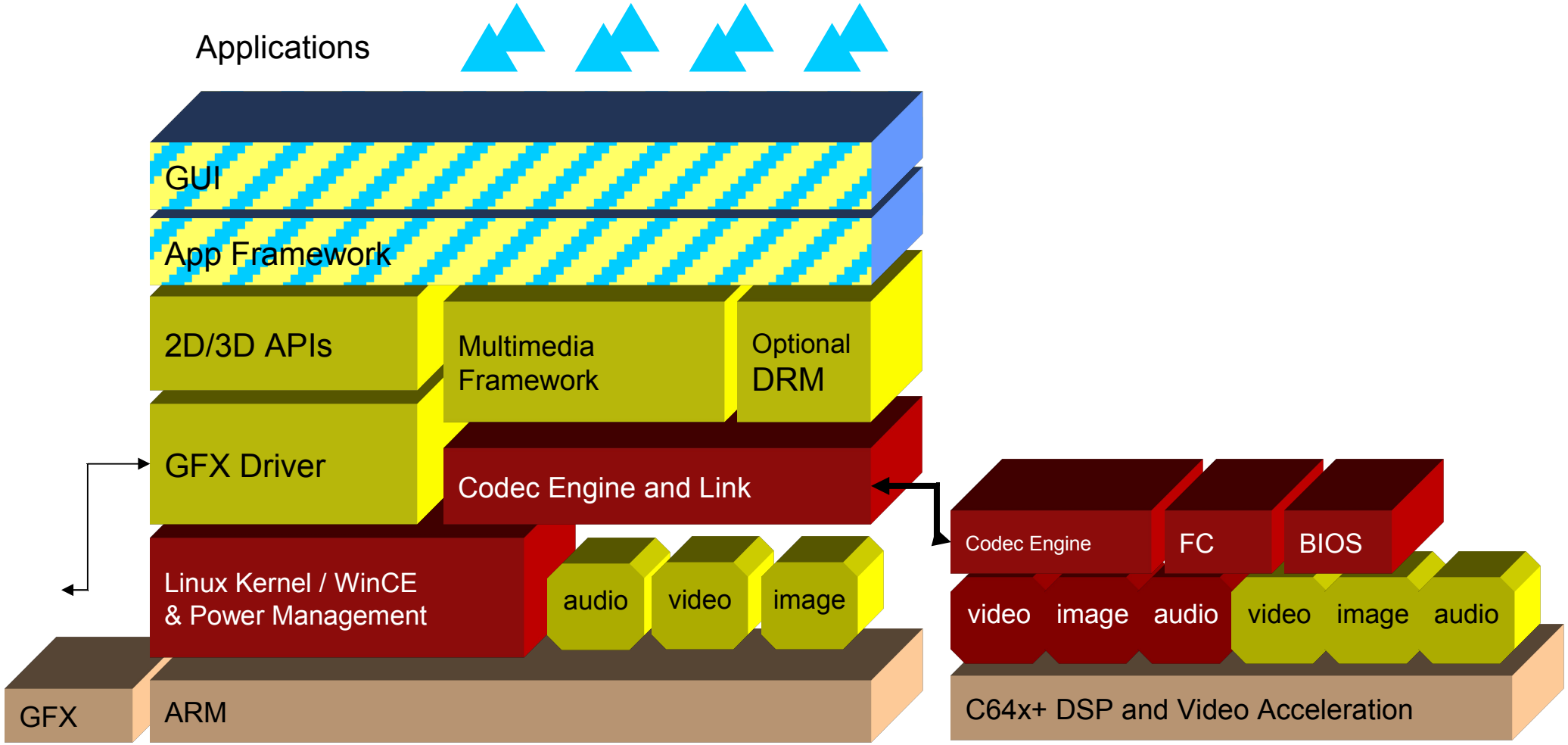
hotter device



Power Block Diagram



Reference Software Architecture



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

Summary

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
  - http://www.netmite.com/android/mydroid/cupcake/
- BeagleBoard and its Linux support
  - http://elinux.org/BeagleBoard
- class PowerManager
  - http://developer.android.com/reference/android/os/PowerManager.html
- Free-Electrons
  - http://free-electrons.com/training
- CELF's power management specification
  - http://elinux.org/Power_Management