POWER MANAGEMENT SYSTEM FOR EMBEDDED RTOS

- Anuja Kumar

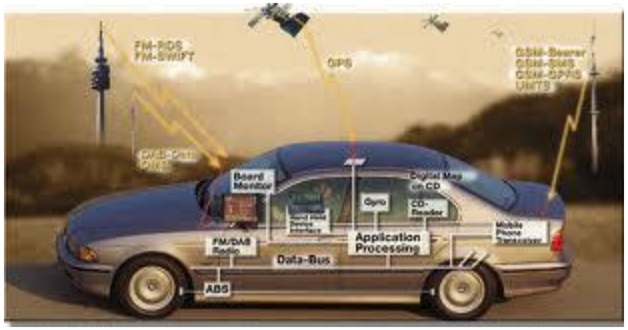
Introduction

• Devices such as household appliances, cell phones and car engines all contain embedded computers.

• Nature of usages and blend of computation extensive applications makes power consumption one of the major concerns in developing these devices

• Need for longer battery life

• Power management is widely employed to contain the energy consumption in power constrained devices.



Power Management

- Problems with PM at lowest level
  - Transistor Level- Dynamic voltage scaling (DVS) and dynamic frequency scaling
  - Moore's law : number of transistors on a chip doubles in every eighteen months - exponential increase in the complexity of embedded systems.
  - Even though more accurate it is unfeasibly complex
  - compounded by time-to-market pressures

- Designs are now being described at higher level - system level design, system-on-chip and networks-on-chip

- Therefore need for system level power management

Power management at OS level ?

• Power management systems for embedded devices can be developed in operating system or in applications

•Advantages of PM at OS level over application level

- developers can concentrate only on application development.
- since OS contains accurate information about the various tasks being executed, it is logical to place algorithms that place components not being used into lower power states.
- can significantly reduces the energy consumption by the system.
- RTOS has a comprehensive set of power management APIs for both device drivers and applications within a power management component

Objectives of paper

- Motivation - need for applications to provide well-defined interfaces between RTOS and devices to manage the overall power of the system.

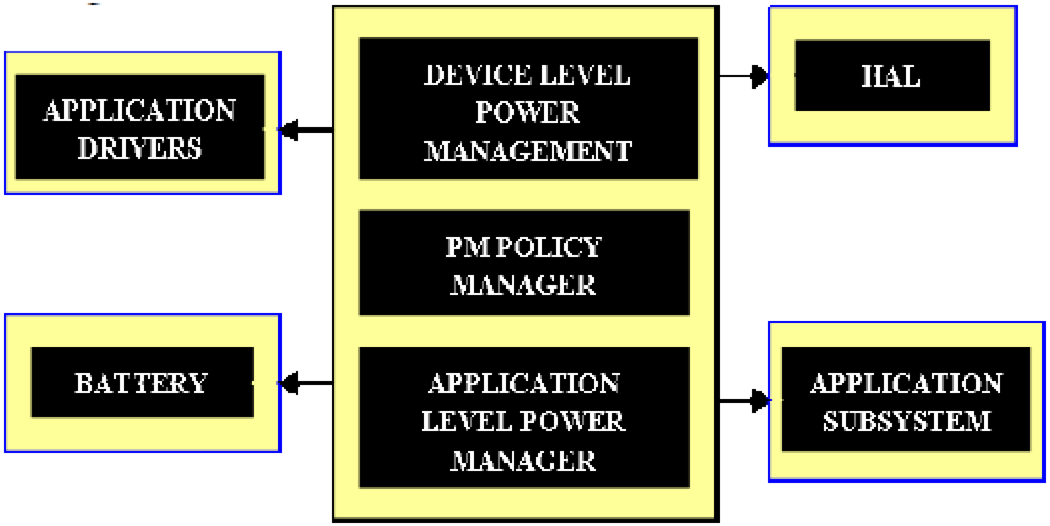
- provide a simple programming interface for the application developers to inform RTOS about application's power and device requirements.

- need for RTOS to inform the application about the current battery status so that the application can keep user informed

- once RTOS is aware of the power requirements, it should be able to bring the complete system into a lowest possible power state

- object-oriented representation for power manager components that are embedded in the RTOS, device drivers and applications.

Power Manager



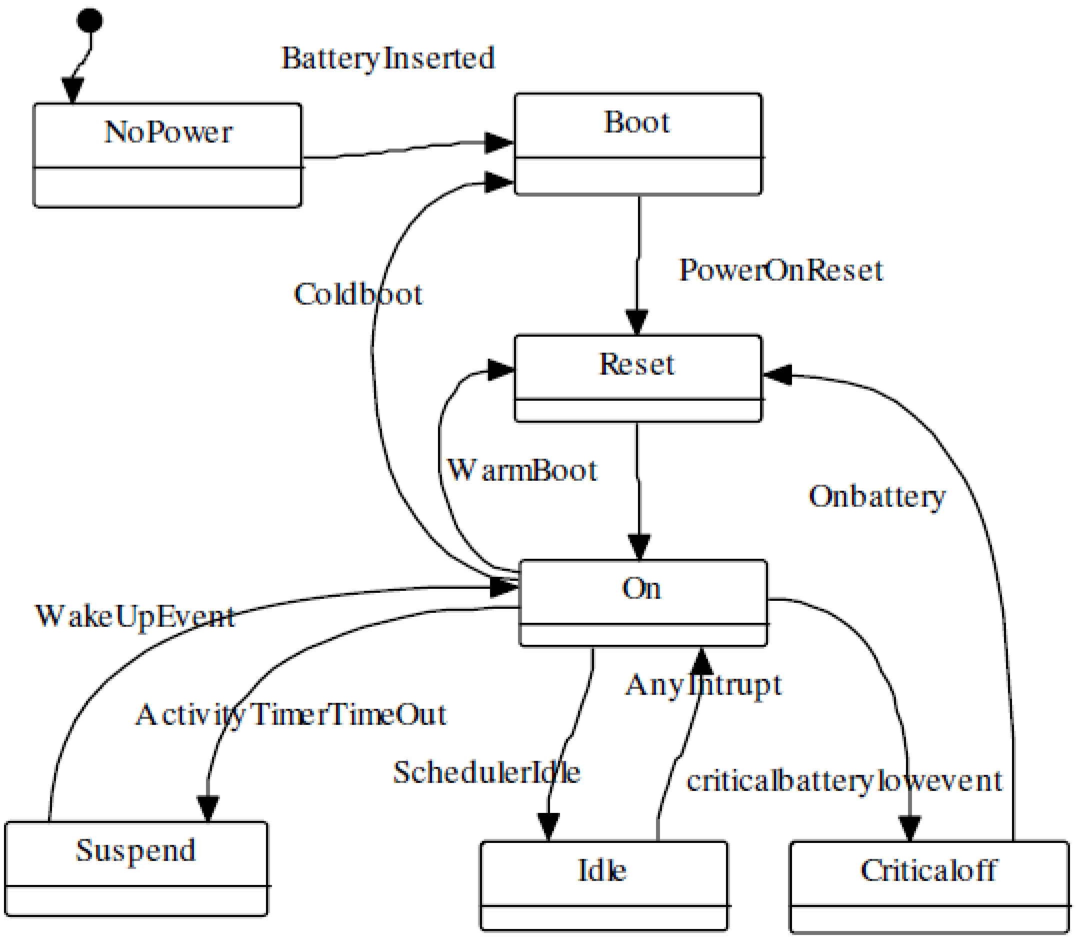
- mediator between devices, applications and processor
- monitors processor utilization to ensure its operation at lowest power
- provides means for drivers to intercommunicate their power states
- allows whole system to work together

Power States

- Seven predefined power states in a system.

1) No Power state (S0) : when system has no power
2) Boot state (S1) : Upon the insertion of the battery
3) On state (S2) : used for the normal operation - system dispatches user mode (application) threads
4) Idle state (S3) : system context is maintained by the hardware, no loss of system context in the CPU or peripheral devices.
5) Suspend state (S4) : where all system contexts are lost except system memory.
6) Critical Off State (S5) : system context is saved and restored when needed.
7) Reset state (S6) : system contexts are properly saved

State Transitions

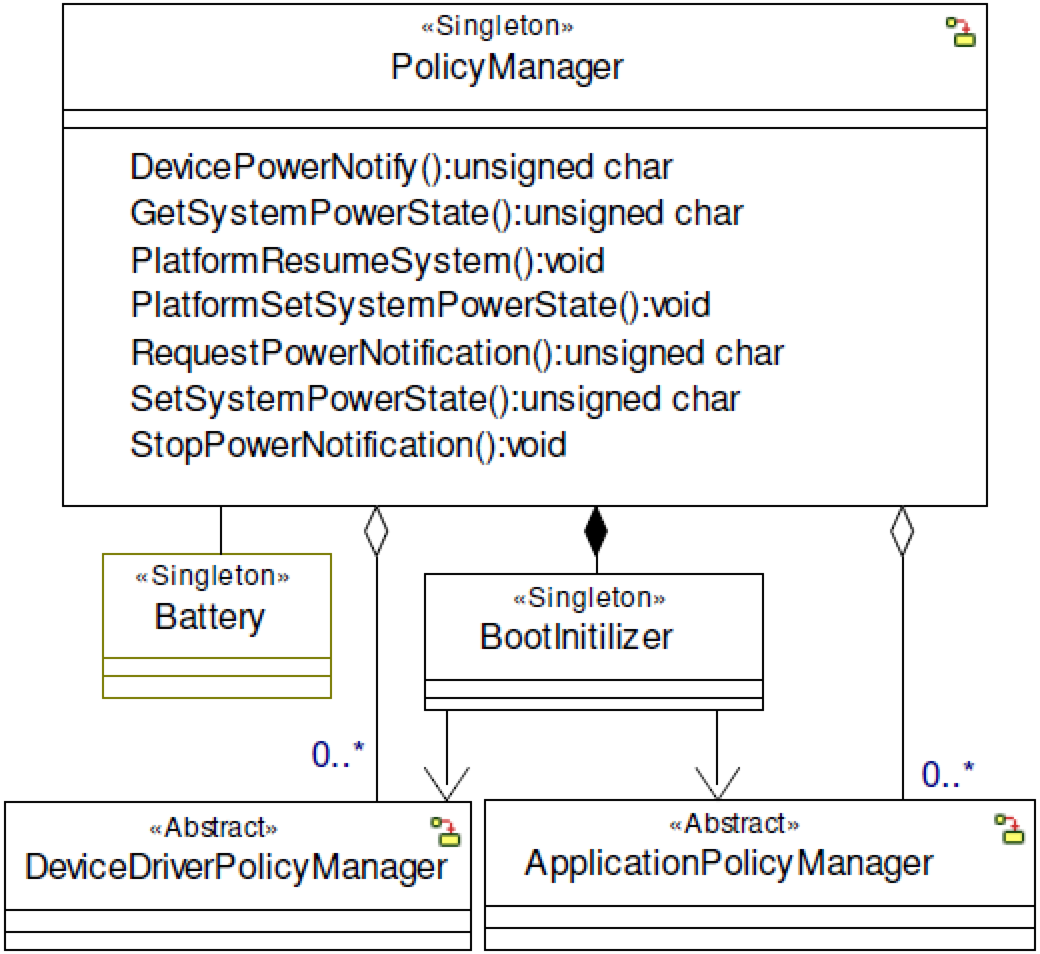


Power Management Classes

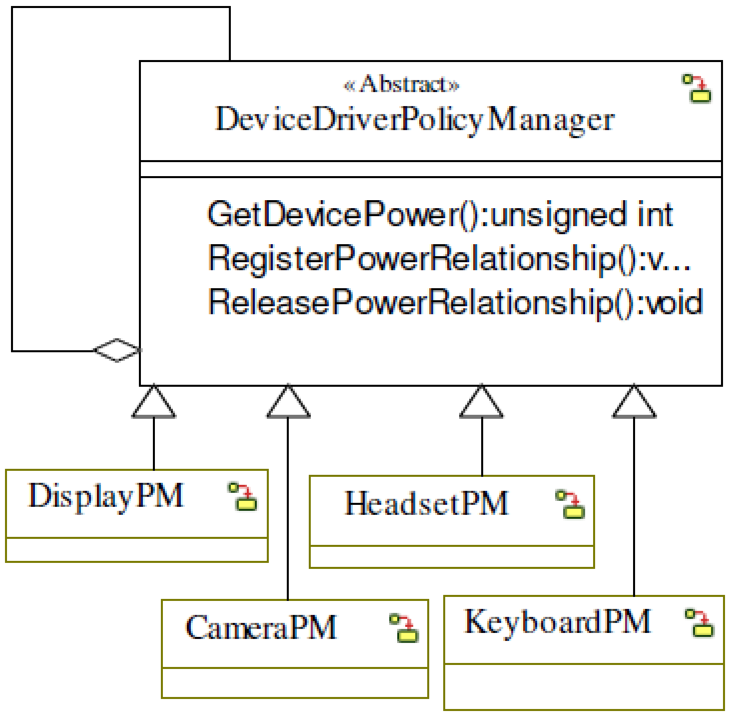
• Power management features abstracted to three different types of classes:

- PolicyManager
- DeviceDriverPolicyManager
- ApplicationPolicyManager

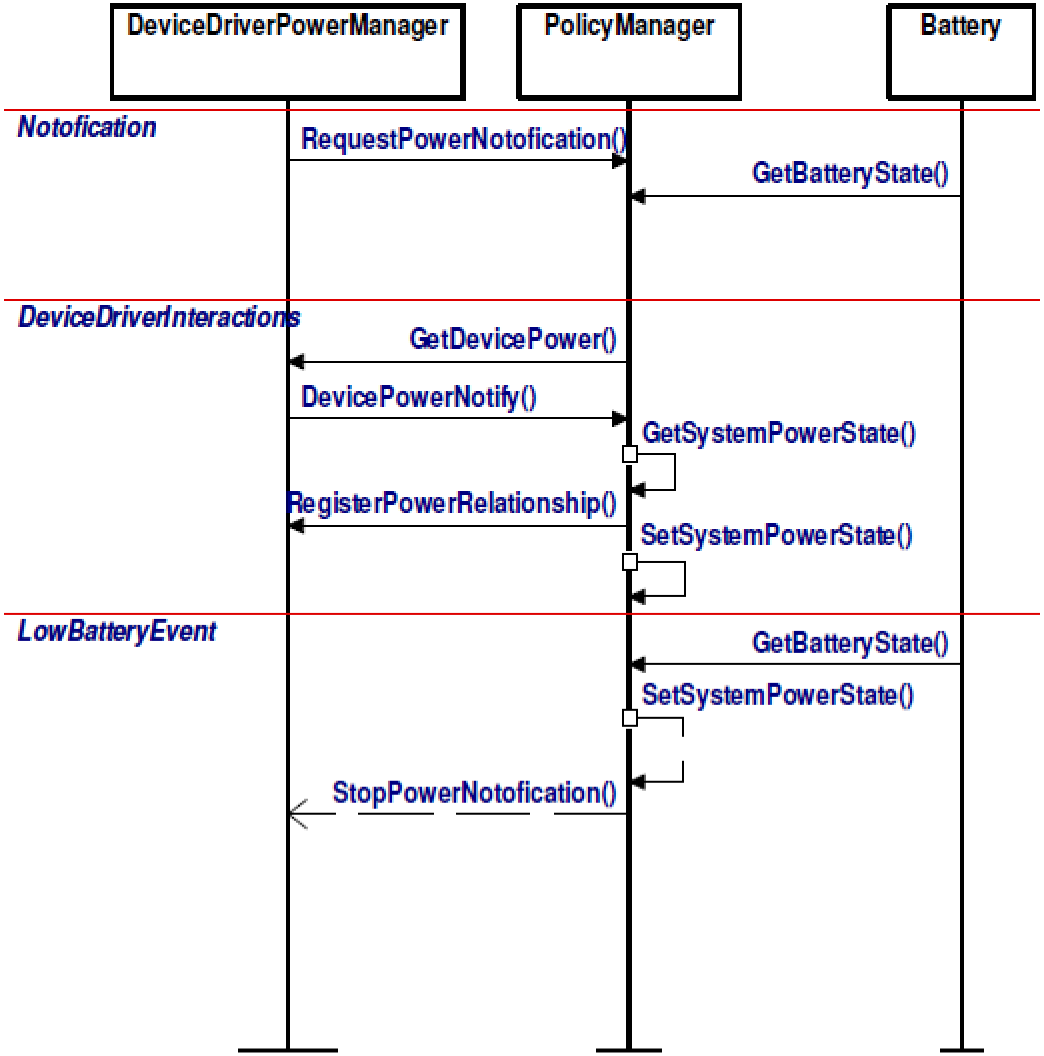
Policy Manager



Device Driver Policy Manager



Device Driver Interaction with Power Manager



Steps for communication

• All the device drivers register themselves with Power Manager through RequestPowerNotification()

•Devices receive an acknowledgement from PM

• The PM reads a list of device classes from registry and uses RequestPowerNotifications() to determine when devices of that class are loaded

• For a device to get activated in the system, the device finds out its current power state by GetDevicePower()

• The device then notifies the policy manager to change its state

• DevicePowerNotifiy() informs the device about the change in its power state.

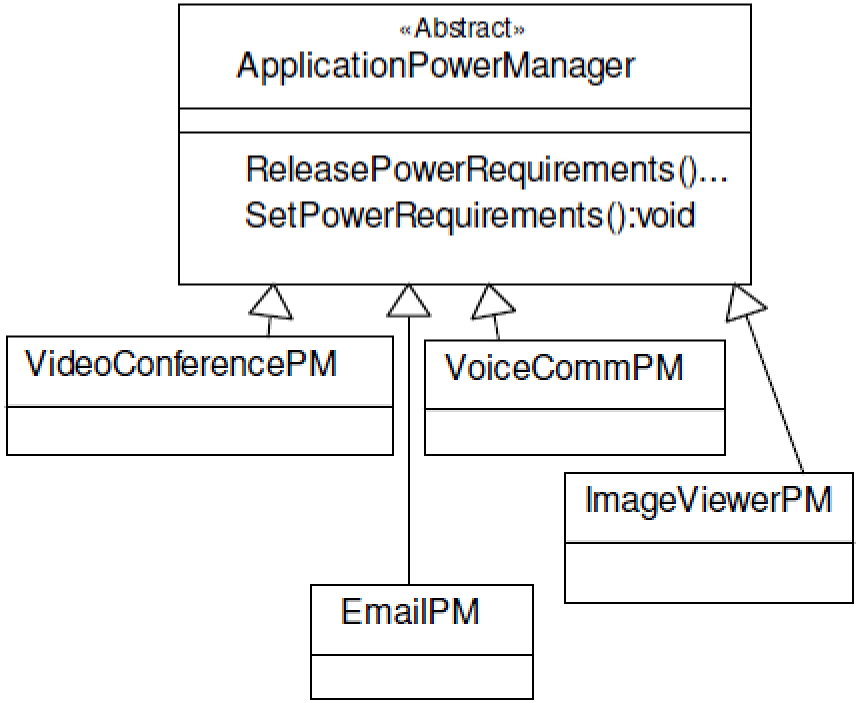
• Once the power state of the device changes the system changes its previous power state to the new power state

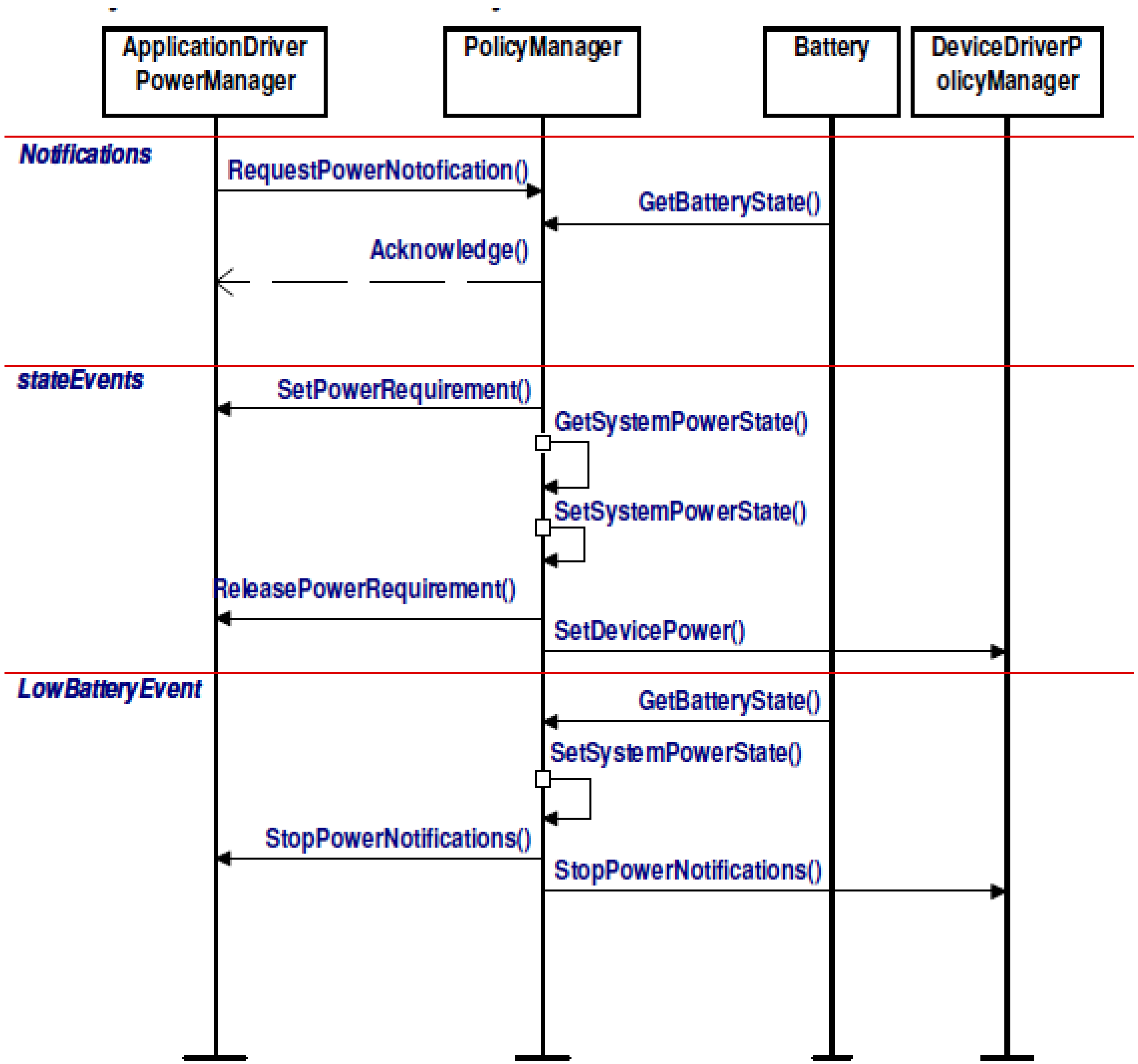
• The Policy Manager constantly monitors battery status.

• If low battery state detected, it notifies DeviceDriverPolicyManager to change the device state of all devices to idle.

•As the device state of the device transits to idle state, an acknowledgement is sent to Policy Manager, which puts the system state to idle.

Application Power Manager




ApplicationDriver PowerManager
PolicyManager
Battery
DeviceDriverP olicyManager
Notifications
RequestPowerNotofication()
GetBatteryState()
Acknowledge()
stateEvents
SetPowerRequirement()
GetSystemPowerState()
SetSystemPowerState()
ReleasePowerRequirement()
SetDevicePower()
LowBatteryEvent
GetBatteryState()
SetSystemPowerState()
StopPowerNotifications()
StopPowerNotifications()

Steps for communication

• The applications register themselves RequestPowerNotification() with PM and receive an acknowledgement

• APM notifies the Policy Manager that an application has a specific device power requirement and sets it using SetPowerRequirement()

• The application also requests the power notification for the specific device drivers it needs in order to execute

• The system responds to its request by changing the power state of those device drivers using SetDevicePower()

• Once the application power requirements are fulfilled the PolicyManager updates the system power state using GetSystemPowerState(), SetSystemPowerStstate()

Conclusion

- Paper explains the operation of the power manager in conjunction with the applications, devices and the processors from the developers point of view.

- Power management within OS : longer battery life

- Well defined interface : lowering overall power needs of system

- Power management interface : simplifies testing for power management scenarios

- The proposed framework will foster development of embedded systems that are more power efficient, easy to maintain, and faster to develop.


QUESTIONS