Charger-Manager:

Aggregating Chargers, Fuel-Gauges, and Batteries

MyungJoo Ham
myungjoo.ham@samsung.com
Samsung Electronics Co.

Topics

- Motivation
  - The Why Questions 1 to 4
- Design
- General Issues
- Appendix
  - Related Framework
  - Interface Design in Detail
  - Usage Example (how to use CM)
  - Related Work (History)
  - References / Images

What are we doing with Charger Manager?

- Monitor the charger/battery health
  - Both suspended and running state
    - Provide suspend_again callback for platform.
- Represent a battery and provide sysfs information
  - Support multiple batteries at a single device
  - Aggregate and merge
    - Multiple chargers
    - A fuel gauge
  - Then, provide combined info to userland. (w/ power-supply-class sysfs)
- Let’s stop adding kernel hacks for mobile devices
  - For battery health monitoring
  - Without self-monitoring/interrupting hardware support.

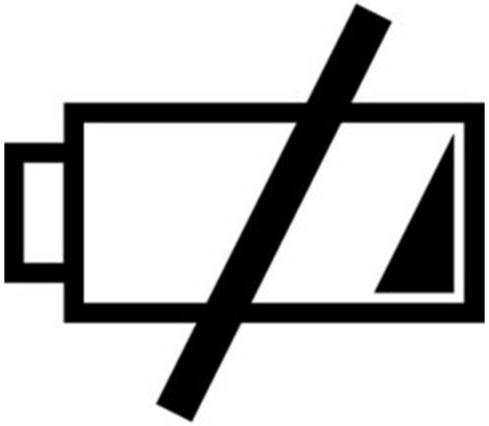
Q1: Why Polling Batteries While Suspended & Charging At Kernel

(thus, creating Charger Manager)

Polling Batteries? Why?

Do **NOT** charge if it’s too HOT or too COLD.

Or, get some explosions or shorter battery life.



The products in the images are not related with the presented content.

Polling Batteries? Why?

Do **NOT** charge if it’s too HOT or too COLD.

However, we often don’t have the **luxury** :
interrupts from temperature sensors

➔ Poll the temperature (or battery health)

Why Suspend while Charging?

"Let's prohibit suspend while charging!"

- We cannot poll things in suspend-to-RAM.
- Forget about power consumption.
  - We have an external power source anyway while charging.

Why Suspend while Charging?

"~~Let's prohibit suspend while charging!~~"

- ~~We cannot poll things in suspend-to-RAM.~~
  - Poll by waking up and suspending again
- ~~Forget about power consumption.~~
  - ~~We have an external power source anyway while charging.~~
  - Power consumption still matters!
    - Charger production < Device consumption, sometimes. really.
    - Battery health

Polling at Kernel’s “Charger Manager”? Why?

What if user land processes poll the battery health?

- It is possible. (including in-suspend polling: wake-up and suspend-again)
- We don’t need to modify kernel.

Polling at Kernel's "Charger Manager"? Why?

What if user land processes poll the battery health?

It is possible. (including in-suspend polling: wake-up and suspend-again)
We don't need to modify kernel.

Userland in-suspend polling incurs a FULL WAKEUP from suspend.

- Every device wakes up and suspends again
  - May take too much time
  - May use too much power

  4~8J vs < 0.004J per poll in a tested device.
  USB 2.0 (5V 500mA), 10s polling ➔ 32% of energy lost
- Every user process wakes up and suspends again
  - Not transparent to other user processes.
  - May start something and block suspending again.
  - User didn't mean to wake the system up.

Q2: Why Support Multiple Chargers with Charger Manager?

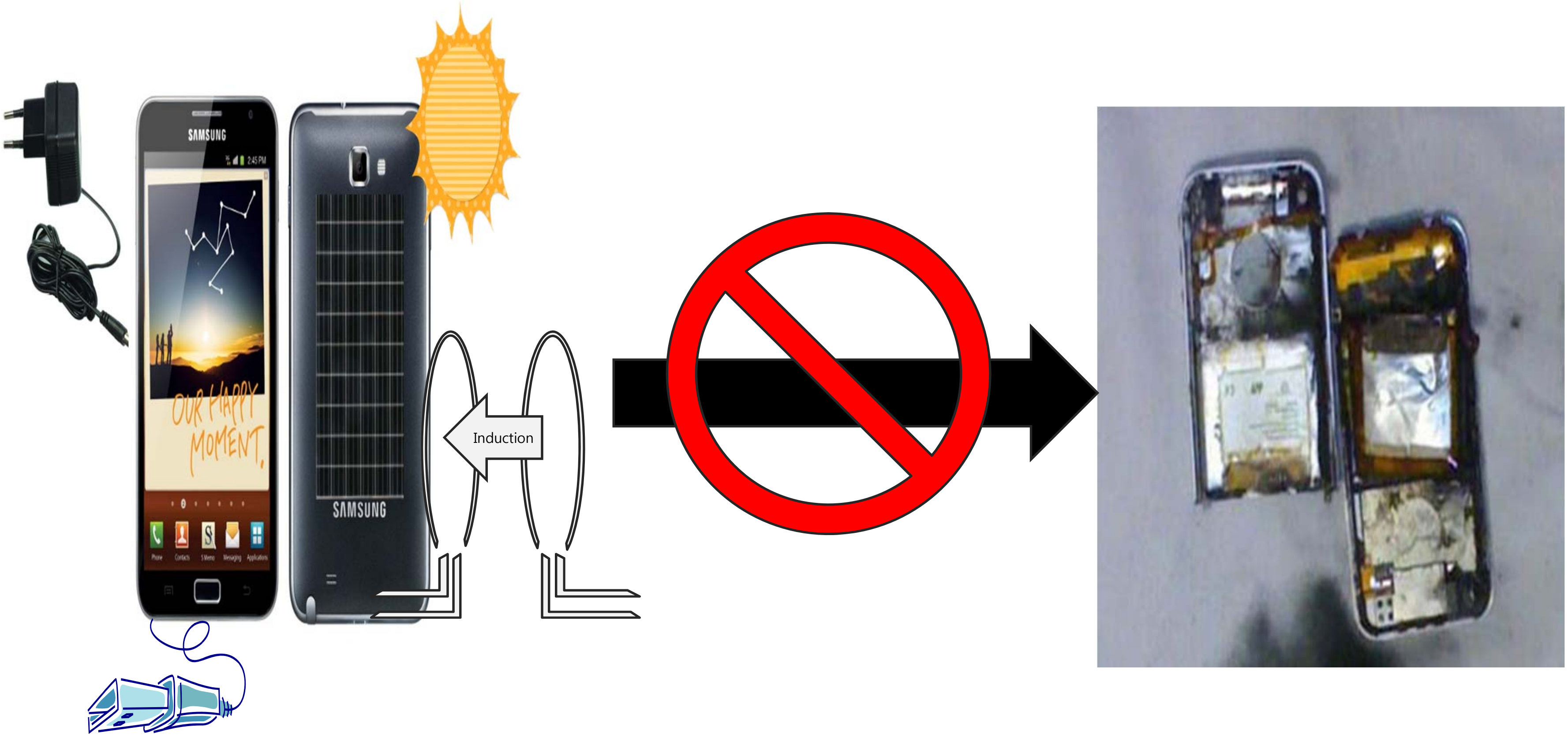
Do We Have Multiple Chargers?

Yes, we do have. And, will have more in the devices to-be-released soon.

SAMSUNG
2:45 PM
OUR HAPPY MOMENT.
Phone
Contacts
S Memo
Messaging
Applications
SAMSUNG
Induction

The products in the images are not related with the presented content.

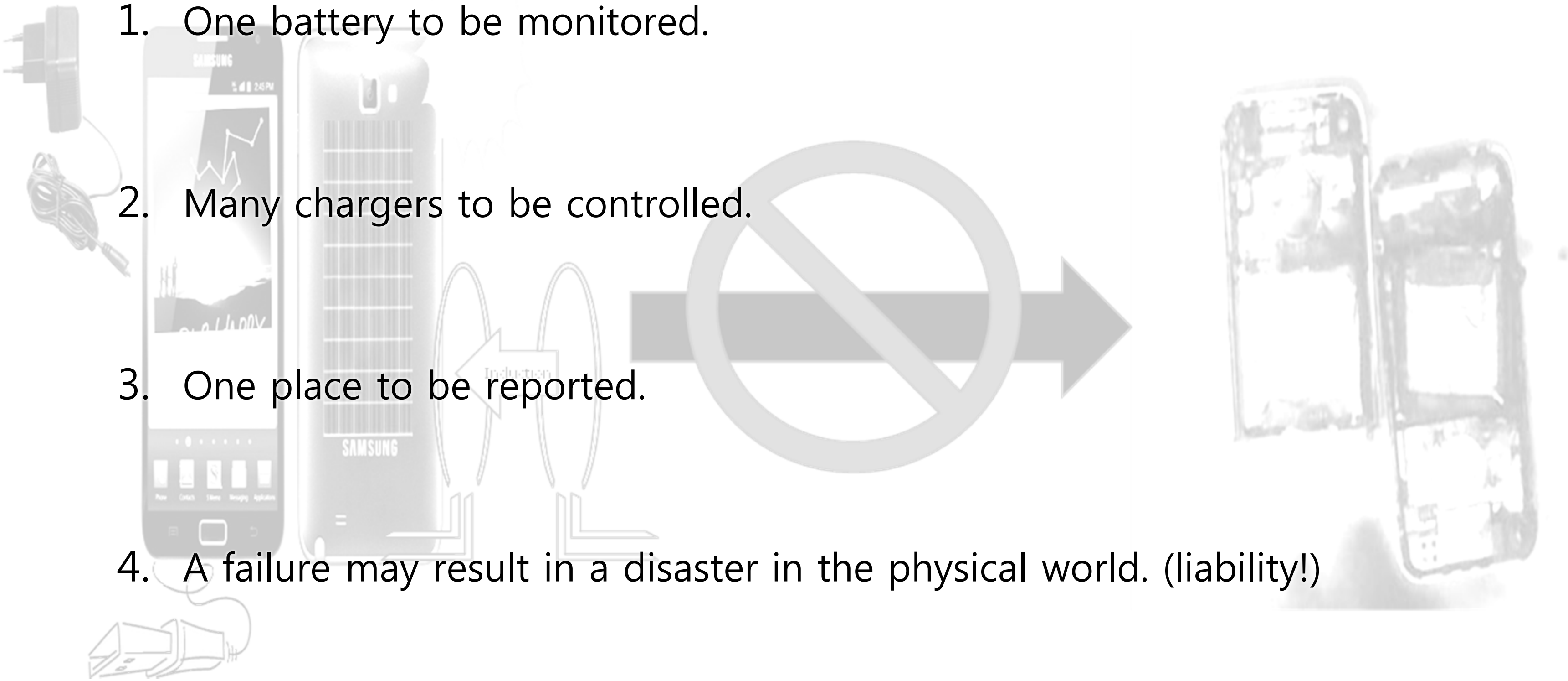
Why Support It At Kernel's Charger Manager?



The products in the images are not related with the presented content.

Why Support It At Kernel’s Charger Manager?

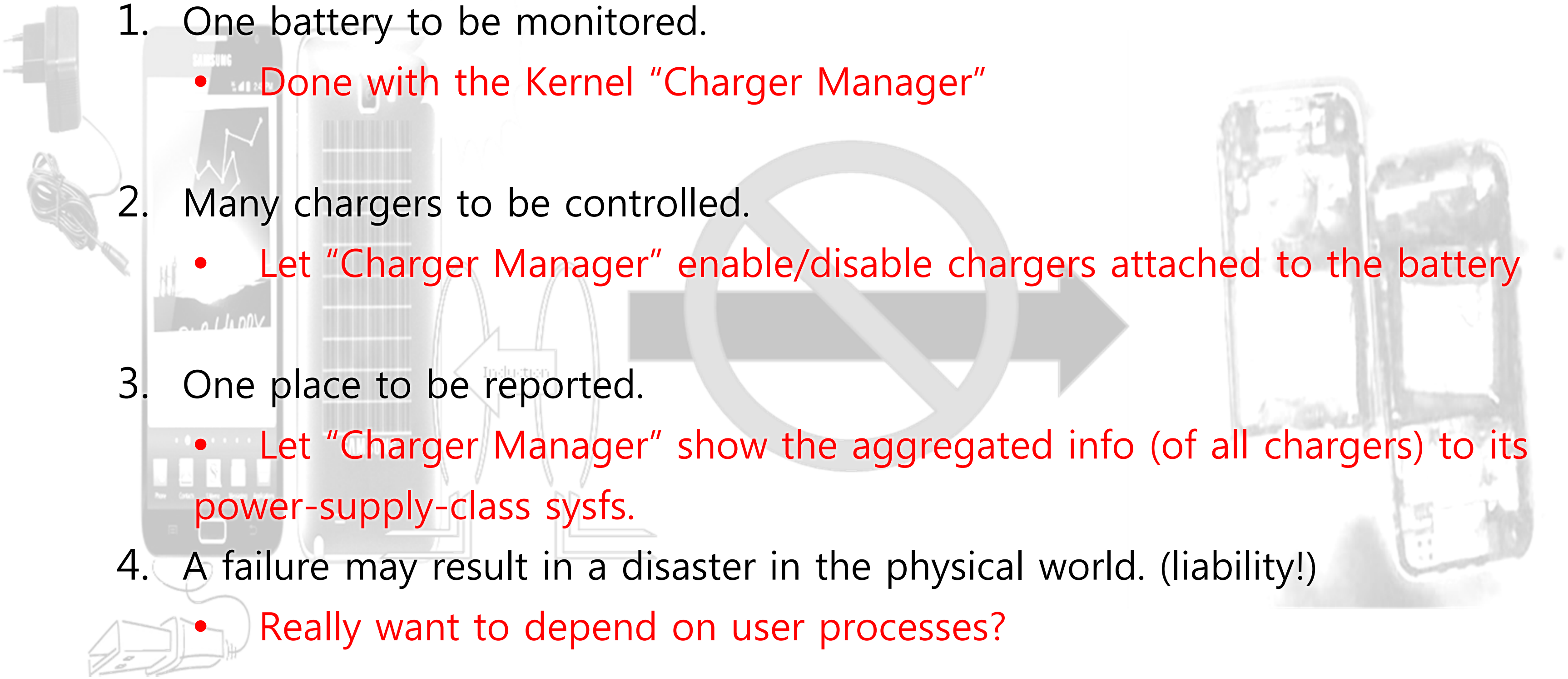
- One battery. Multiple chargers. One device. Critical task.
  1. One battery to be monitored.
  2. Many chargers to be controlled.
  3. One place to be reported.
  4. A failure may result in a disaster in the physical world. (liability!)



The products in the images are not related with the presented content.

Why Support It At Kernel's Charger Manager?

- One battery. Multiple chargers. One device. Critical task.
  1. One battery to be monitored.
     - Done with the Kernel "Charger Manager"
  2. Many chargers to be controlled.
     - Let "Charger Manager" enable/disable chargers attached to the battery
  3. One place to be reported.
     - Let "Charger Manager" show the aggregated info (of all chargers) to its power-supply-class sysfs.
  4. A failure may result in a disaster in the physical world. (liability!)
     - Really want to depend on user processes?



The products in the images are not related with the presented content.

Q3: Why Support Multiple Batteries at a Device with Charger Manager

Multiple Batteries at a Device

WHY NOT?

Although it is rare...... yet...

- A backup battery and a main battery?

May the device be powered during sudden battery-out.

Main Batteries

Backup Battery

I don’t think you can run today’s Linux kernel on this device anyway... (0.000027 BogoMips!*)

The products in the images are not related with the presented content.

* http://tldp.org/HOWTO/BogoMips/bogo-list.html

Multiple Batteries at a Device

WHY NOT?

Although it is rare…… yet…

- Easy to support:
  Simply allow to have multiple instances of Charger Manager.

- One instance of Charger Manager for one battery

- Let them share the polling loop.
  Especially for the suspended state.

Q4: Why Aggregate Information from Multiple Chargers?

Information Spread All Over the SYSFS Places

For a battery X, we have chargers A, B, and C and a fuel gauge K.

For a battery Y, we have chargers D and E and a fuel gauge L.

Then, we may have:

/sys/class/power_supply/A/charging = 50000	status = CHARGING
/sys/class/power_supply/B/charging = 10000	status = CHARGING
/sys/class/power_supply/C/charging = 0	status = NOT CHARGING
/sys/class/power_supply/D/charging = 0	status = UNKNOWN
/sys/class/power_supply/E/charging = 0	status = UNKNOWN
/sys/class/power_supply/K/charging: not exist	status = UNKNOWN
/sys/class/power_supply/L/charging = 0	status = DISCHARGING

How should we interpret this at userland?

- Difficult to “decode”.
- May get inconsistent information for the same battery.
- Relation between components are not visible.

Information Spread All Over the SYSFS Places

For a battery X, we have chargers A, B, and C and a fuel gauge K.

For a battery Y, we have chargers D and E and a fuel gauge L.

Then, we may have:

/sys/class/power_supply/A/charging = 50000	status = CHARGING
/sys/class/power_supply/B/charging = 10000	status = CHARGING
/sys/class/power_supply/C/charging = 0	status = NOT CHARGING
/sys/class/power_supply/D/charging = 0	status = UNKNOWN
/sys/class/power_supply/E/charging = 0	status = UNKNOWN
/sys/class/power_supply/K/charging: not exist	status = UNKNOWN
/sys/class/power_supply/L/charging = 0	status = DISCHARGING

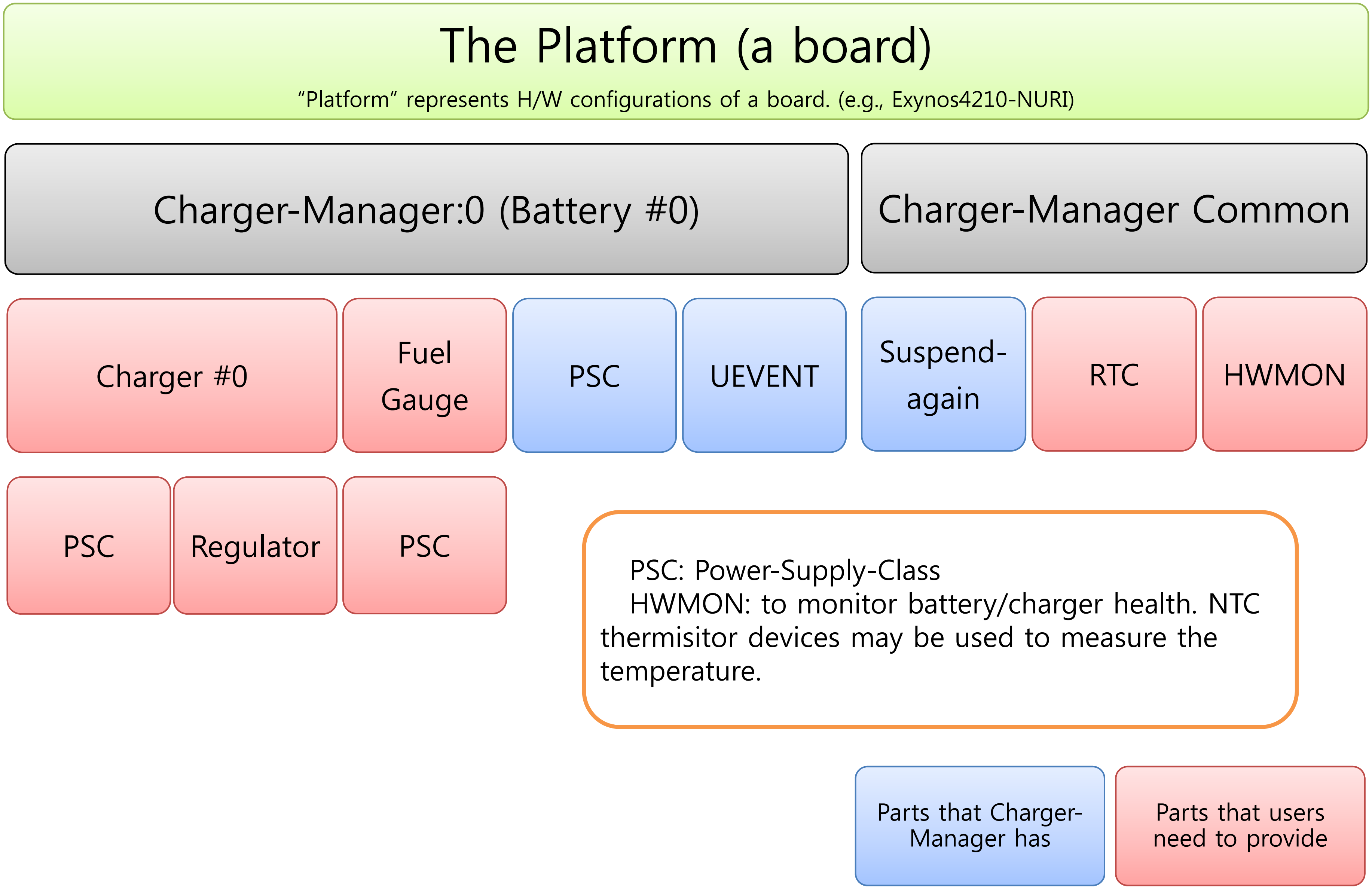
How should we interpret this at userland?

Let's add /sys/class/power_supply/X and Y to clarify this…

/sys/class/power_supply/X/charging = 60000	status = CHARGING
/sys/class/power_supply/Y/charging = 0	status = DISCHARGING

Charger-Manager Design

Design: System Layout



The Platform (a board)

"Platform" represents H/W configurations of a board. (e.g., Exynos4210-NURI)

Charger-Manager:0 (Battery #0)

Charger-Manager Common

Charger #0

Fuel Gauge

PSC

UEVENT

Suspend-again

RTC

HWMON

PSC

Regulator

PSC

PSC: Power-Supply-Class

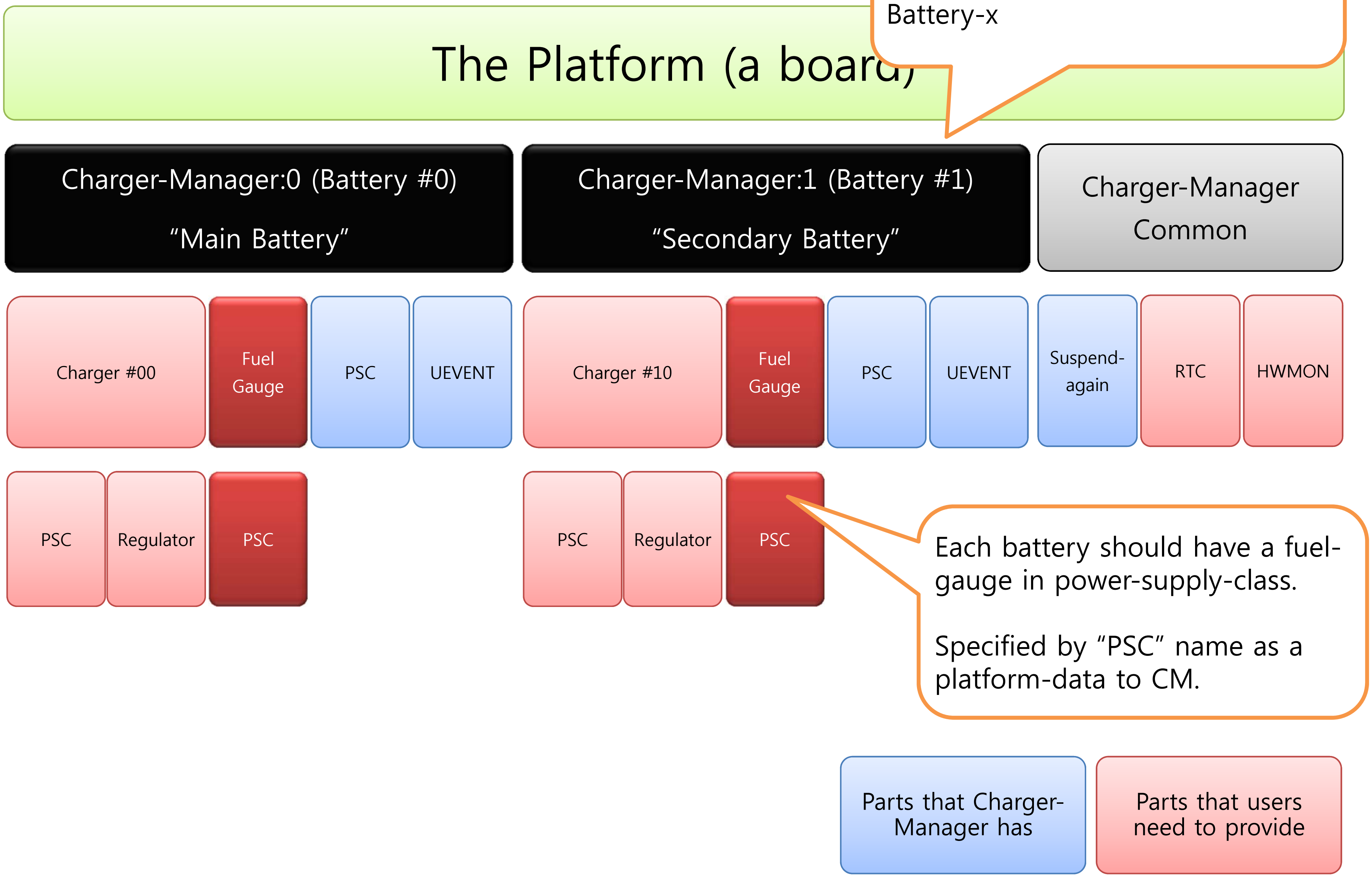
HWMON: to monitor battery/charger health. NTC thermisitor devices may be used to measure the temperature.

Parts that Charger-Manager has

Parts that users need to provide

Design: Allow multiple batteries.

Charger-Manager-x
Represents
Battery-x



The Platform (a board)

Charger-Manager:0 (Battery #0)

"Main Battery"

Charger-Manager:1 (Battery #1)

"Secondary Battery"

Charger-Manager Common

Charger #00

Fuel Gauge

PSC

UEVENT

Charger #10

Fuel Gauge

PSC

UEVENT

Suspend-again

RTC

HWMON

PSC

Regulator

PSC

PSC

Regulator

PSC

Each battery should have a fuel-gauge in power-supply-class.

Specified by "PSC" name as a platform-data to CM.

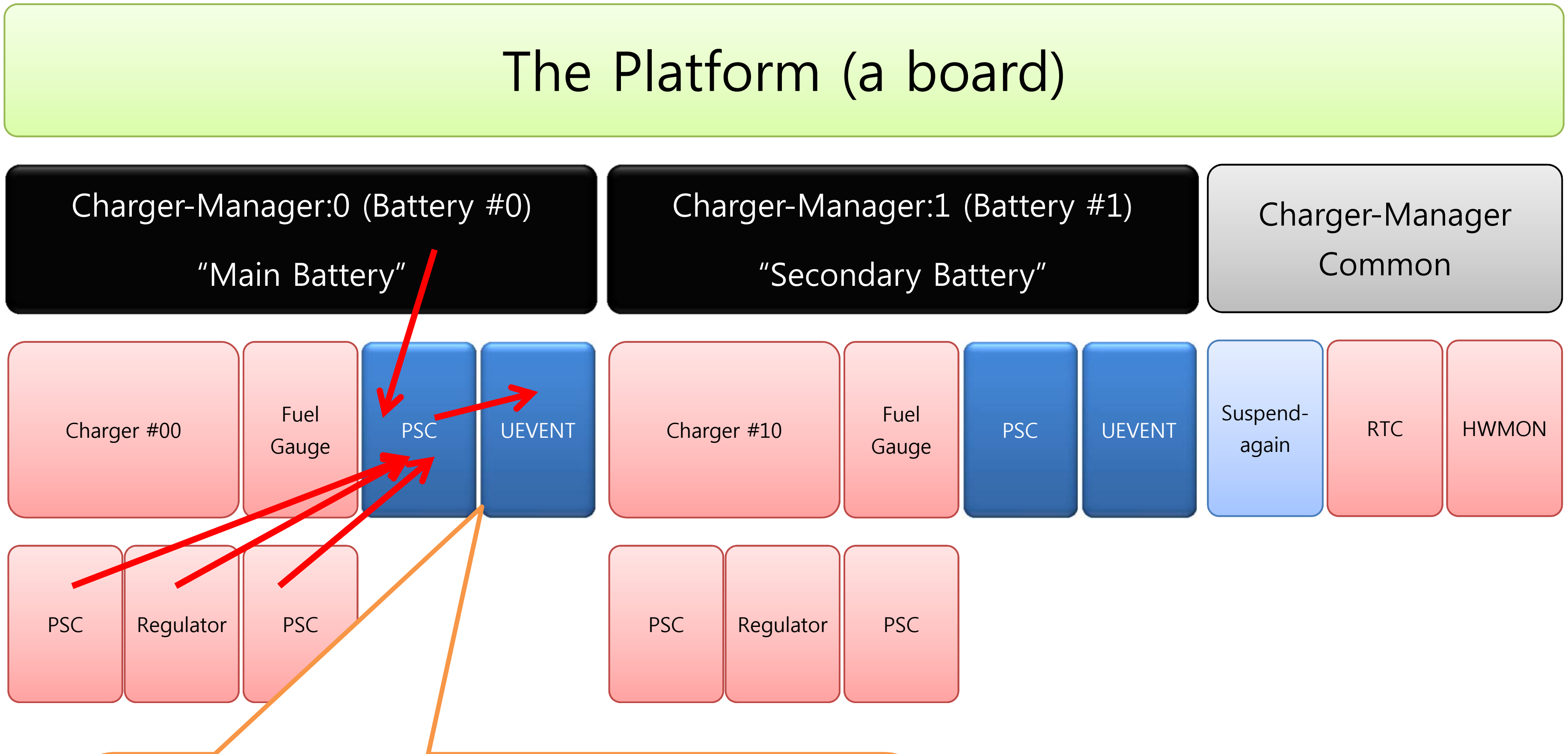
Parts that Charger-Manager has

Parts that users need to provide

Design: Fuel-Gauge

- Provide a Power-Supply-Class name that supports
  - PRESENT: Battery presence in 1/0 (optional if a charger provides PRESENT)
  - VOLTAGE_NOW: Battery voltage in μV
  - CURRENT_NOW: Current from battery in μA (optional)
  - CAPACITY: Remaining battery capacity in %
  - CHARGE_NOW: Charging status in μA or "any positive-number"/0 (optional)
- The entries should be accessible before "suspend_noirq( )" and after "resume_noirq( )" sequences
  - They must be accessible at suspend(), resume(), suspend_again().

Design: Information for Userland



The Platform (a board)

Charger-Manager:0 (Battery #0)

"Main Battery"

Charger-Manager:1 (Battery #1)

"Secondary Battery"

Charger-Manager Common

Charger #00

Fuel Gauge

PSC

UEVENT

Charger #10

Fuel Gauge

PSC

UEVENT

Suspend-again

RTC

HWMON

PSC

Regulator

PSC

PSC

Regulator

PSC

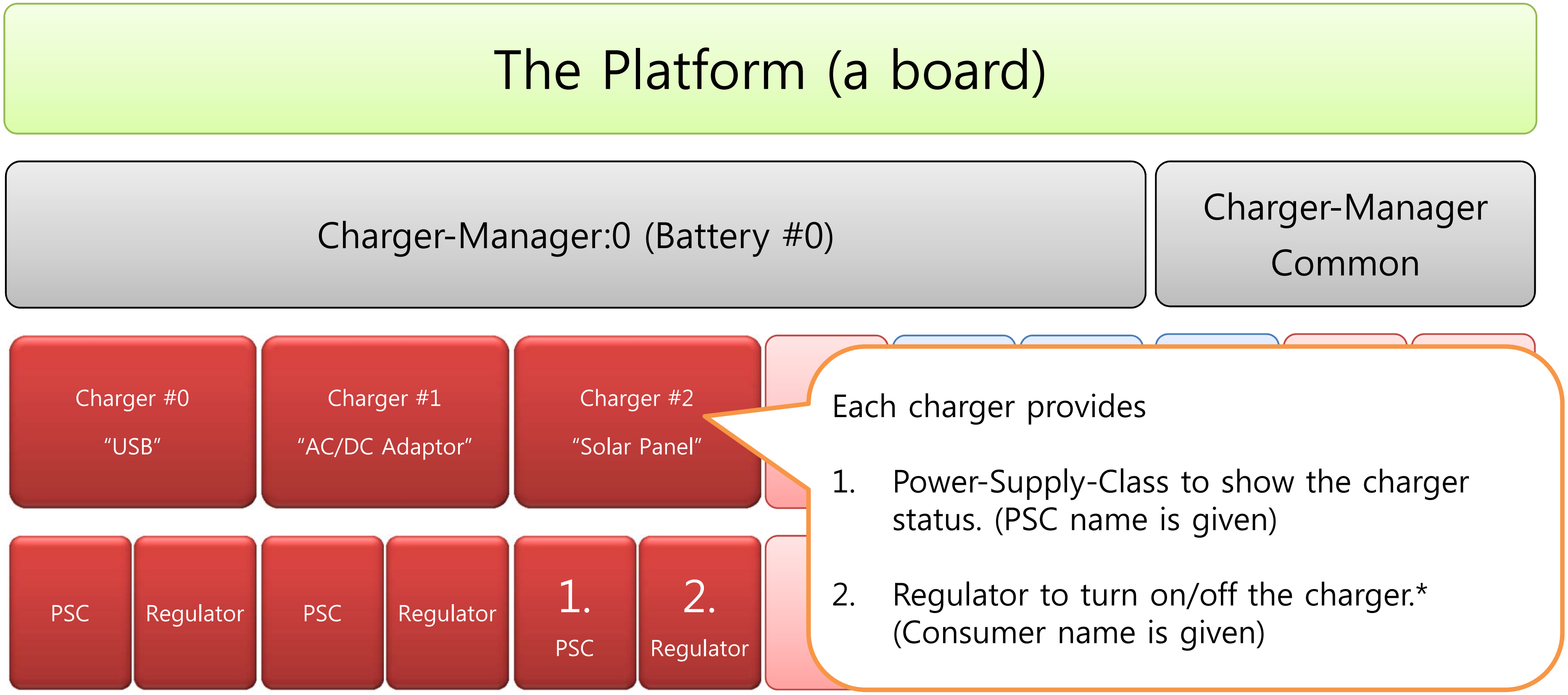
P-S-C information about the battery aggregates related chargers, fuel-gauge, and the charger-manager itself. Default name = "battery".

UEVENT notifies changes in the status.

Parts that Charger-Manager has

Parts that users need to provide

Design: Allow multiple chargers per battery.



The Platform (a board)

Charger-Manager:0 (Battery #0)

Charger-Manager Common

Charger #0 "USB"

Charger #1 "AC/DC Adaptor"

Charger #2 "Solar Panel"

PSC

Regulator

PSC

Regulator

1. PSC

2. Regulator

Each charger provides

1. Power-Supply-Class to show the charger status. (PSC name is given)
2. Regulator to turn on/off the charger.* (Consumer name is given)

* The charger regulators are used

- To force_disable() at emergency (too hot/too cold)
- To force restart charging at a given condition: call disable() and enable().
- RFC: need to use regulator_get_exclusive() ?

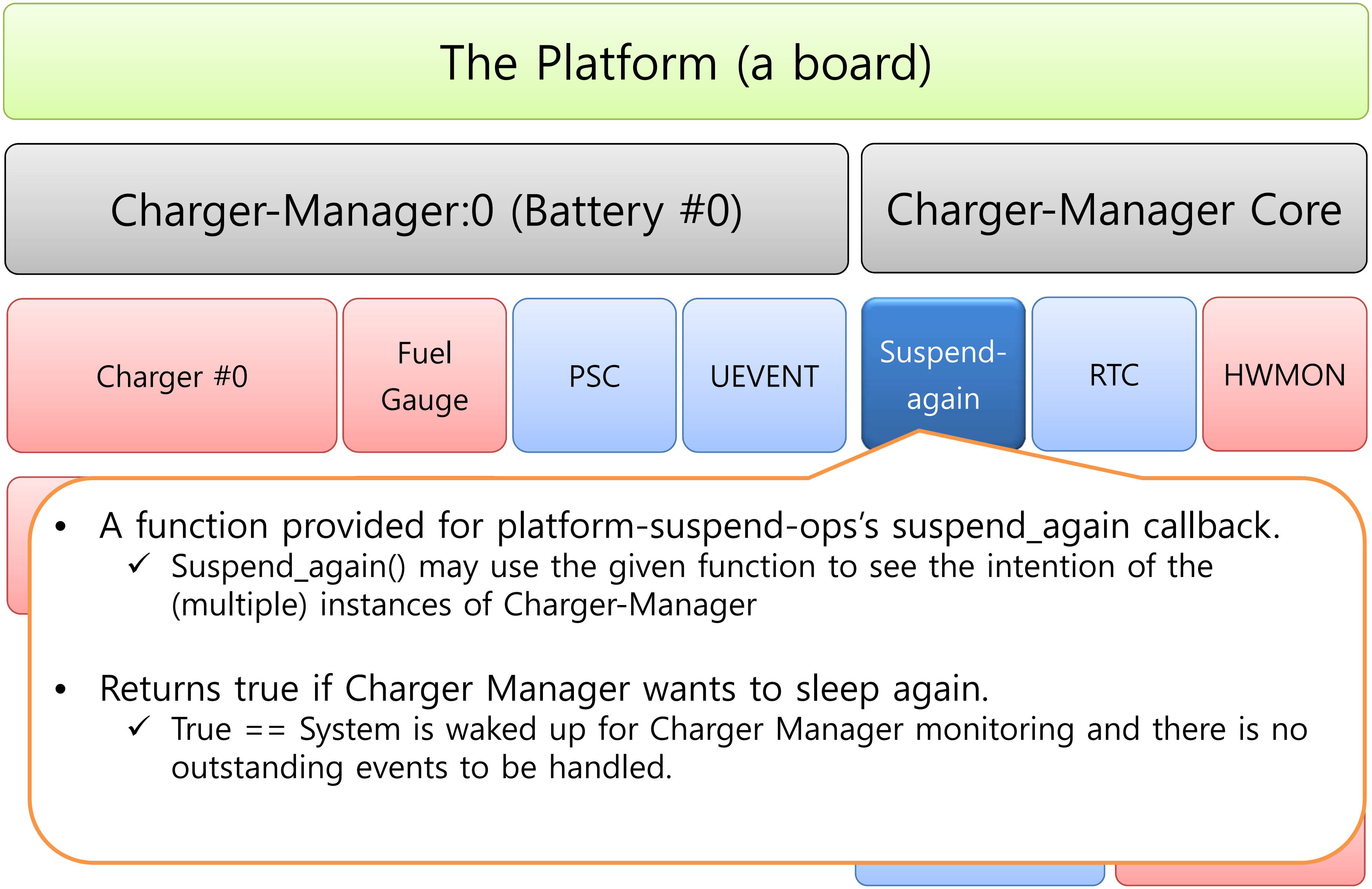
🟦 Parts that Charger-Manager has

🟥 Parts that users need to provide

Design: Chargers

- Provide a regulator
  - With REGULATOR_CHANGE_STATUS flag enabled.
  - Recommended to supply “is_enabled()”.
    - Otherwise, restart charging with voltage drop may not work properly.
  - The charger should supply “enable()” and “disable()” callbacks.
- Provide a Power-Supply-Class name that supports
  - PRESENT: Battery presence in 1/0 (optional if fuel gauge provides PRESENT)
  - ONLINE: External power connection in 1/0
  - STATUS: FULL/Discharging/Not-Charging/Charging
- The entries should be accessible before “suspend_noirq( )” and after “resume_noirq( )” sequences
  - They must be accessible at suspend(), resume(), suspend_again().

Design: In-suspend Monitoring



The Platform (a board)

Charger-Manager:0 (Battery #0)

Charger-Manager Core

Charger #0

Fuel Gauge

PSC

UEVENT

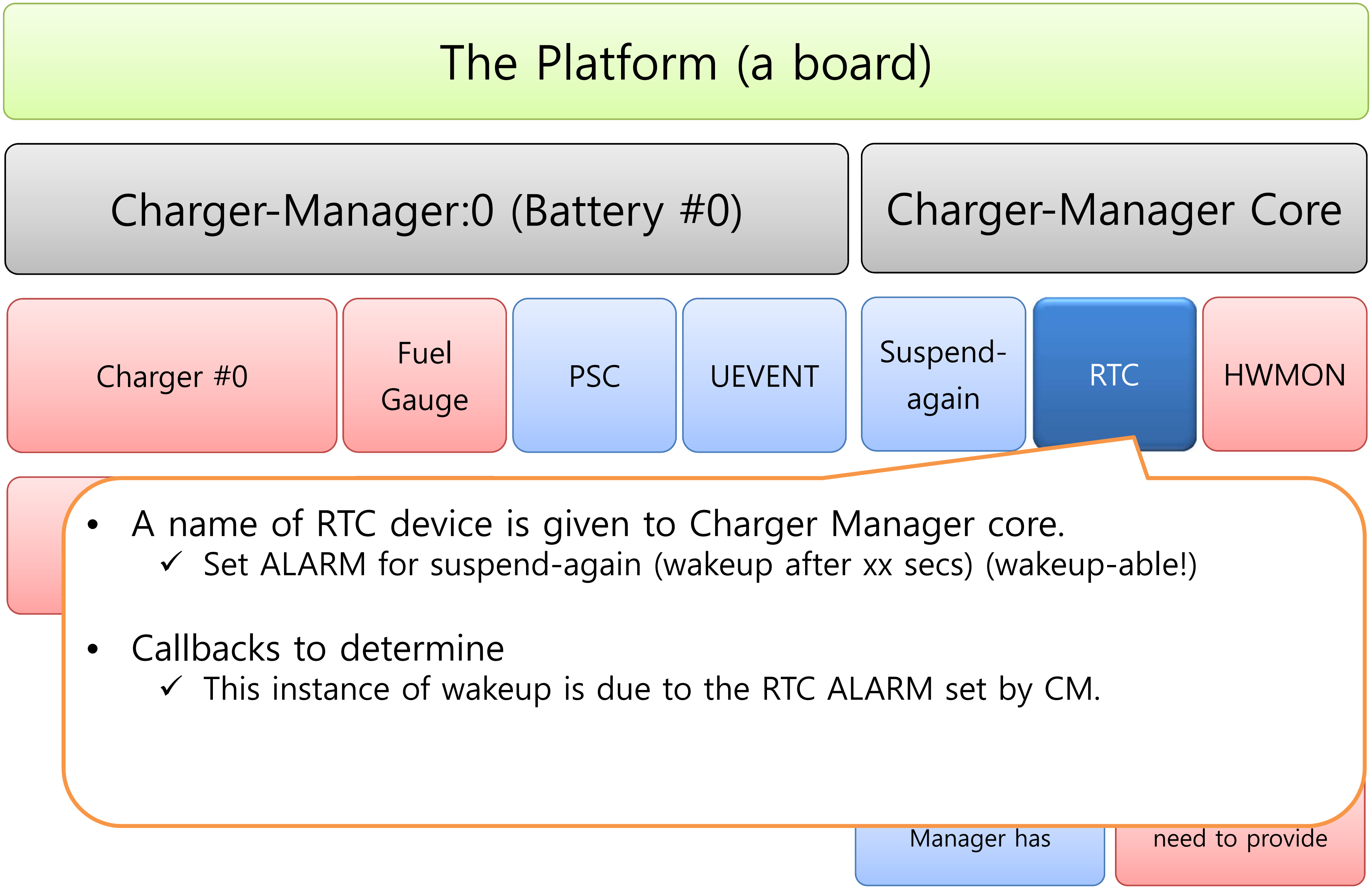
Suspend-again

RTC

HWMON

- A function provided for platform-suspend-ops’s suspend_again callback.
    - ✓ Suspend_again() may use the given function to see the intention of the (multiple) instances of Charger-Manager
- Returns true if Charger Manager wants to sleep again.
    - ✓ True == System is waked up for Charger Manager monitoring and there is no outstanding events to be handled.

Design: In-suspend Monitoring



The Platform (a board)

Charger-Manager:0 (Battery #0)

Charger-Manager Core

Charger #0

Fuel Gauge

PSC

UEVENT

Suspend-again

RTC

HWMON

- A name of RTC device is given to Charger Manager core.
  - ✓ Set ALARM for suspend-again (wakeup after xx secs) (wakeup-able!)
- Callbacks to determine
  - ✓ This instance of wakeup is due to the RTC ALARM set by CM.

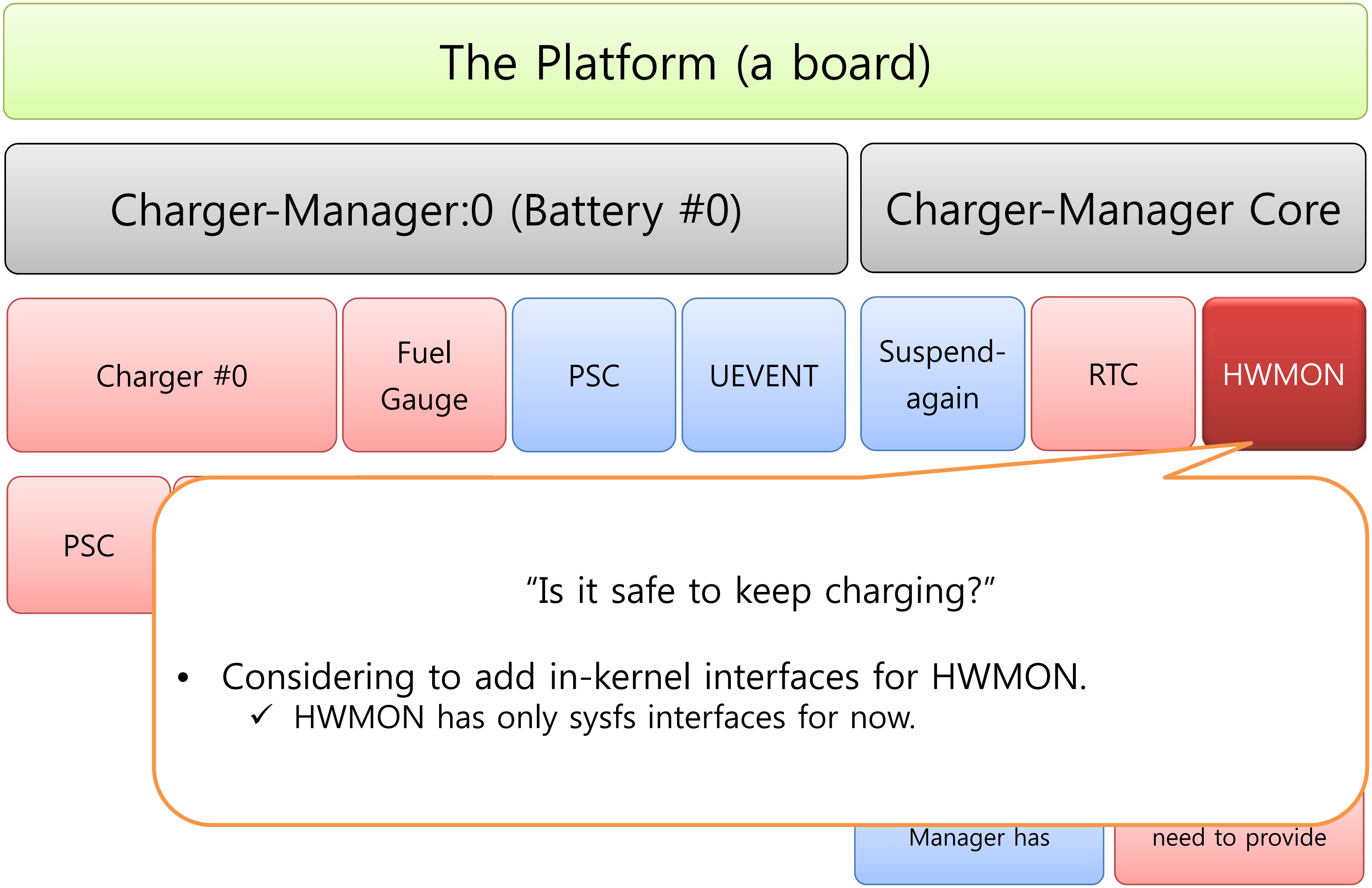
Manager has

need to provide

Design: RTC

- Provide an RTC device name
  - E.g., “rtc0”
  - Supports ALARM set, TIME read
  - ALARM should be able to wake up from suspend
- A callback to determine an RTC ALARM wakeup
  - bool (*is_rtc_only_wakeup_reason)(void)

Design: HWMON



The Platform (a board)

Charger-Manager:0 (Battery #0)

Charger-Manager Core

Charger #0

Fuel Gauge

PSC

UEVENT

Suspend-again

RTC

HWMON

PSC

Manager has

need to provide

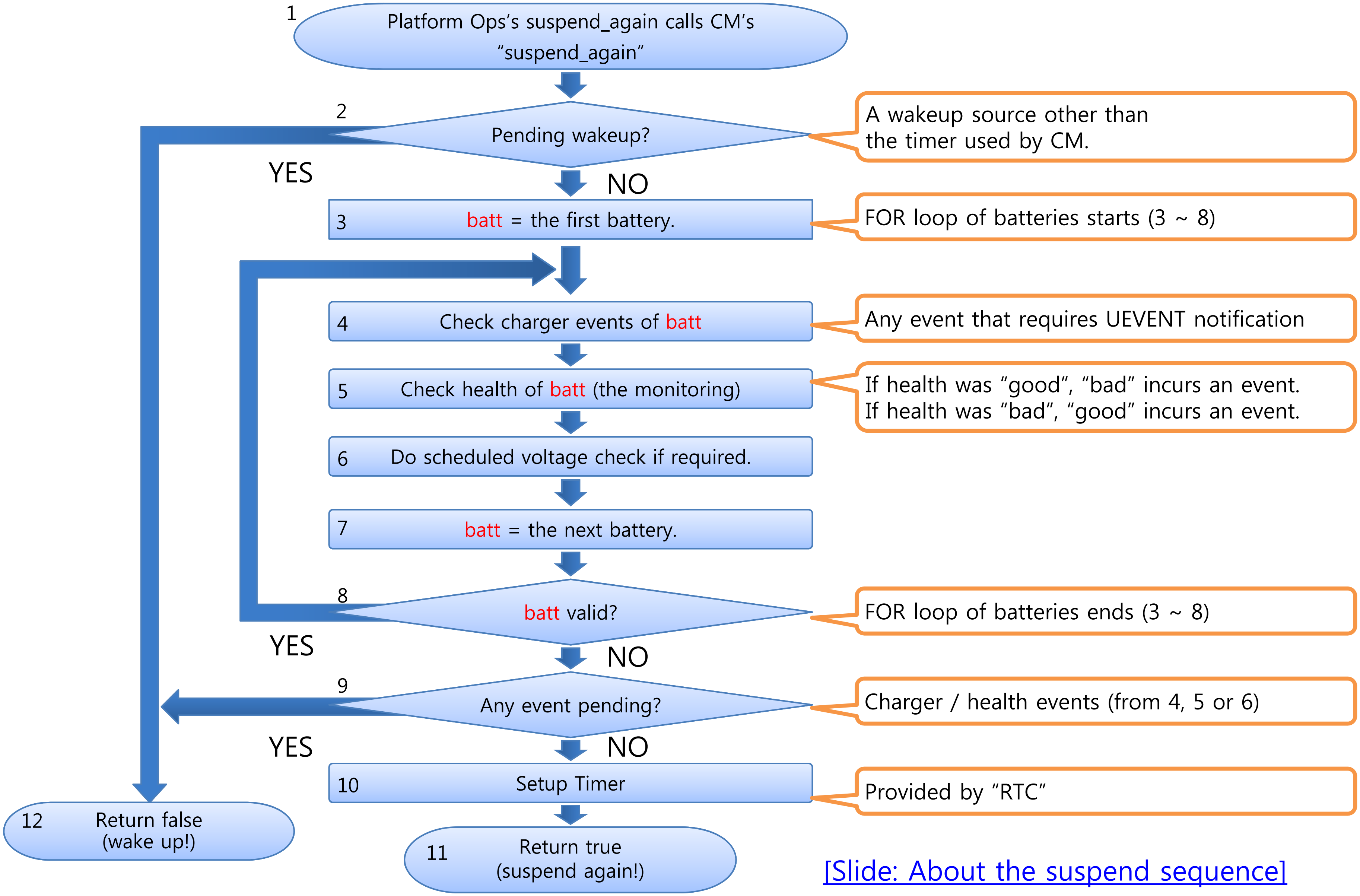
"Is it safe to keep charging?"

- Considering to add in-kernel interfaces for HWMON.
  - ✓ HWMON has only sysfs interfaces for now.

Design: HWMON (RFC)

- Candidate 1: Supply a callback (current implementation)
  - int is_temperature_error(int *mC)
    - Returns >0: too hot, <0: too cold.
- Candidate 2: Supply a HWMON name (looks better)
  - Need to modify HWMON framework.
    - An interface to access HWMON data in kernel without accessing sysfs.

Design: In-suspend Monitoring: Suspend_Again



1 Platform Ops's suspend_again calls CM's "suspend_again"

2 Pending wakeup?

A wakeup source other than the timer used by CM.

YES

NO

3 batt = the first battery.

FOR loop of batteries starts (3 ~ 8)

4 Check charger events of batt

Any event that requires UEVENT notification

5 Check health of batt (the monitoring)

If health was "good", "bad" incurs an event.
If health was "bad", "good" incurs an event.

6 Do scheduled voltage check if required.

7 batt = the next battery.

8 batt valid?

FOR loop of batteries ends (3 ~ 8)

YES

NO

9 Any event pending?

Charger / health events (from 4, 5 or 6)

YES

NO

10 Setup Timer

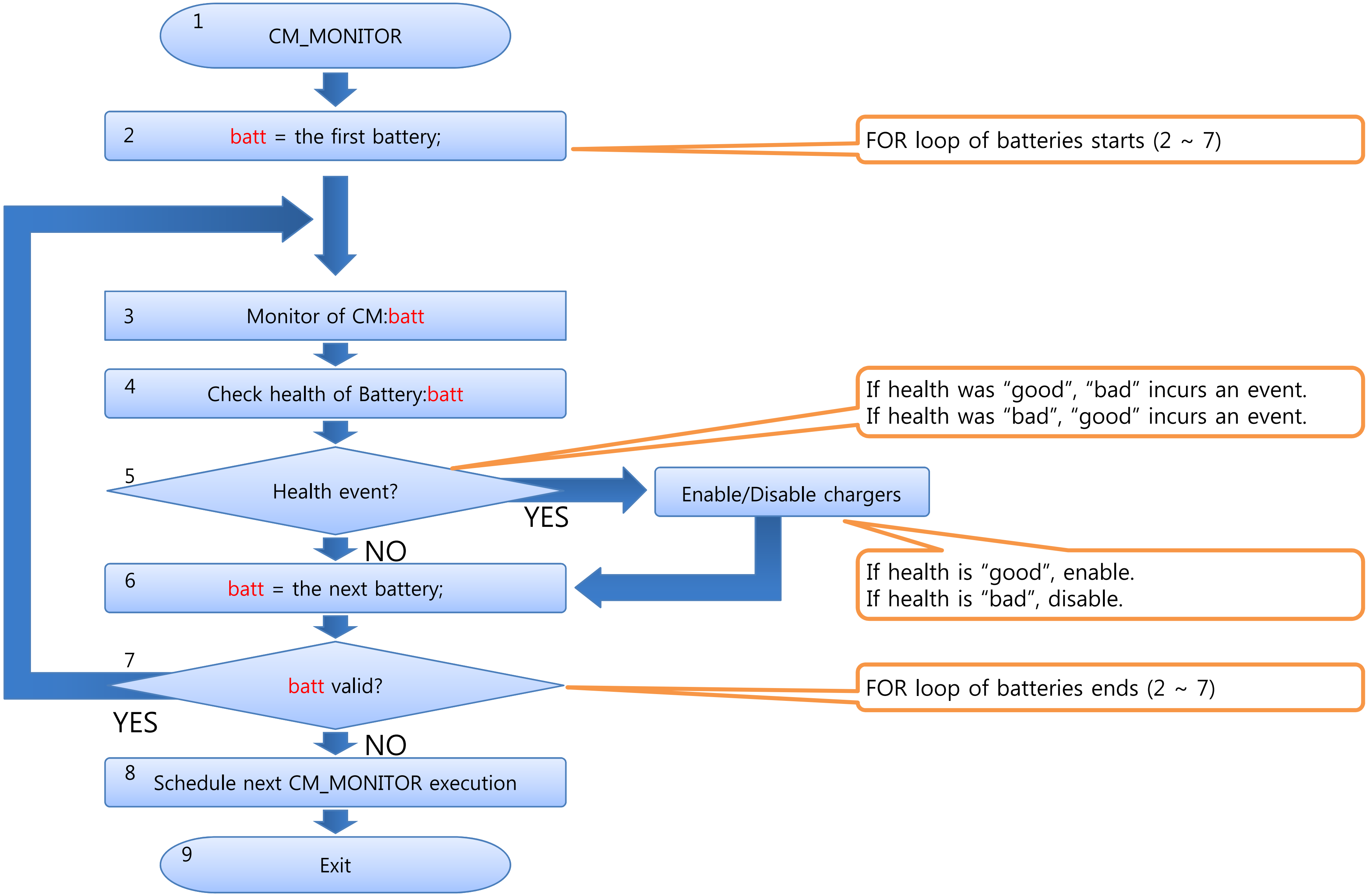
Provided by "RTC"

12 Return false (wake up!)

11 Return true (suspend again!)

[Slide: About the suspend sequence]

Design: Monitoring in Running State


1 CM_MONITOR
2 batt = the first battery;
3 Monitor of CM:batt
4 Check health of Battery:batt
5 Health event?
YES
Enable/Disable chargers
NO
6 batt = the next battery;
7 batt valid?
YES
NO
8 Schedule next CM_MONITOR execution
9 Exit

FOR loop of batteries starts (2 ~ 7)

If health was "good", "bad" incurs an event.
If health was "bad", "good" incurs an event.

If health is "good", enable.
If health is "bad", disable.

FOR loop of batteries ends (2 ~ 7)

MyungJoo Ham

Design: Interrupt/Event Handling

- Interrupt/Event Handling
  - Provide IRQ numbers to CM to handle those interrupts
  - Call CM event functions to handle the events
- With the interrupts/events
  - CM notifies userland with UEVENT
    - With the given interrupt/event name.
  - CM cancels suspend_again to handle the interrupts
    - If the interrupt is marked to be “wakeup”.
  - CM restarts chargers
    - If the interrupt/event is marked so.

More Detail in Appendix

- Interface for board files (platform files)
  - Global CM data
  - CM platform data for each battery
- Other in-kernel APIs
- Interface for user processes

General Issues

in implementing charger-related drivers

General Issues

Caution: JIG

- Do NOT use JIG power when charger is tested
  - E.g., w/ 30P JTAG JIG, set “JIG ON” switch OFF and “POWER” switch OFF.
  - JIG power disrupts battery and PMIC behavior.
  - When JIG power is used, we do not need charging anyway.

Debugging

- Serial is suspended at the early stage of suspend. To activate serial,
  - Resume serial (UART) temporarily during suspend_again
  - Disable “console_suspend” or
    call resume_console() and suspend_console() at suspend_again.
- Print out every uevent_notify() result at Charger-Manager framework.
- Look at /proc/interrupts for event counting
- Look at /sys/class/power_supply/battery/uevent for status summary
- Check the charger status with current/power meters attached at the charger.
- Note that MAX17042 (and other current-based fuel gauges) requires some tuning to get the correct capacity and current values.

General Issues

PMIC Drivers

- Battery “PRESENT” information from MAX8998/8997/LP3974/…
  - “PRESENT” is not updated if there is no chargers attached. (H/W chip spec)
  - The value is only valid when there is a charger attached.

Measuring Battery Voltage while Charging

- May be exaggerated by the charger.

Temperature Measurement

- Look at the circuit schematics and the board carefully
- Be careful on which thermistor is measuring whose temperature.
  - Ambient temp?
  - Chip temp?
  - Battery core temp?
  - Battery surface temp?
  - …
- The specification may differ with different measuring points.

General Issues

Measuring Charger Current

- Some fuel gauges (such as MAX17042) provide the current information. However, it requires tuning values based on the circuit and cannot be measured during sleep or deep-idle.
- Thus, it is recommended to use external current/power meter on the charger at development & debugging stages.

Heat Dissipation during Charging

- Don't be alarmed by some heat while charging.
- While charging, high current is poured into the device. Thus, some heat dissipation from the device (PMIC and battery), which makes the device hotter than normal, is normal.

How Battery Temperature Monitoring is Implemented

- Stop charging if TEMP > HI
  - Continue charging if TEMP < HI - e
- Stop charging if TEMP < LOW
  - Continue charging if TEMP > LOW + e

General Issues

Reading values from related devices

- I2C subsystem cannot be used after suspend_noirq / before resume_noirq.
  - Usually PMIC and Fuel Gauge use I2C.
- ADC may be not available after its own suspend / before resume.

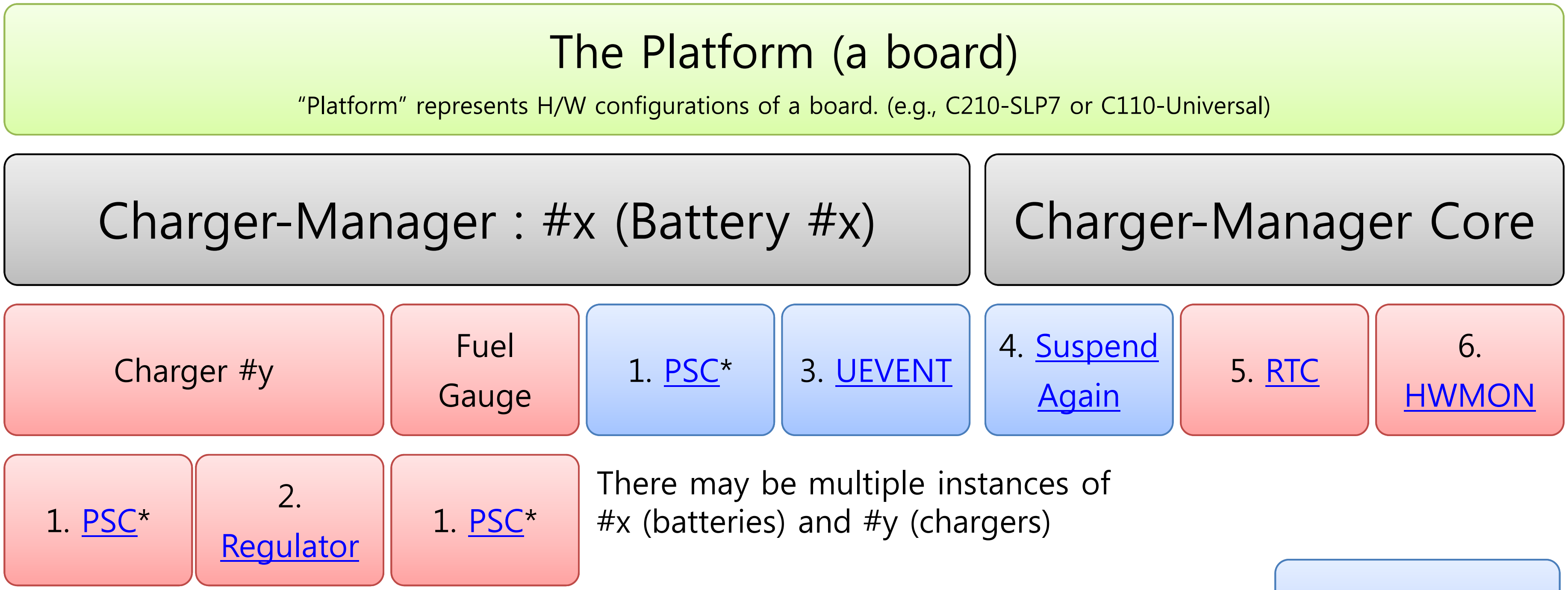
Thank You!

Appendix

1. Related Framework
2. The Interface in Detail
3. Usage Example
4. Related Work (History)
5. References

Appendix: Related Framework

Related Framework



The Platform (a board)

“Platform” represents H/W configurations of a board. (e.g., C210-SLP7 or C110-Universal)

Charger-Manager : #x (Battery #x)

Charger-Manager Core

Charger #y

Fuel Gauge

1. PSC*

3. UEVENT

4. Suspend Again

5. RTC

6. HWMON

1. PSC*

2. Regulator

1. PSC*

There may be multiple instances of #x (batteries) and #y (chargers)

1. PSC (Power-Supply-Class)
2. Regulator framework (usually PMIC provides)
3. UEVENT (Event notification to user processes)
4. Suspend_Again API of platform_suspend_ops
5. RTC (Real Time Clock)
6. HWMON (Temperature ADC sensor)

🟦 Parts that Charger-Manager provides

🟥 Parts that users need to provide to CM

* PSC framework is used in different types of devices.
Note: the links in this page go to slides

Related Framework: Power Supply Class

Power Supply Class

- Kernel documents
  - Link: Documentation/power/power-supply-class.txt
- The drivers should provide
  - Charger
    - PRESENT: Battery presence in 1/0 (optional if fuel gauge provides PRESENT)
    - ONLINE: External power connection in 1/0
    - STATUS: FULL/Discharging/Not-Charging/Charging/Unknown (should provide FULL or not. Other statuses are optional)
  - Fuel Gauge
    - PRESENT: Battery presence in 1/0 (optional if charger provides PRESENT)
    - VOLTAGE_NOW: Battery voltage in μV
    - CURRENT_NOW: Current from battery in μA (optional)
    - CAPACITY: Remaining battery capacity in %
    - CHARGE_NOW: Charging status in 1/0 or in μA (optional)
- CM provides information to userland (SLP Platform) with PSC
  - Battery-related APIs for SLP [Slide: Interface for Users]

Related Framework: Regulator

Regulator Framework

- SPS documents
  - Link: System SW/Linux/drivers/regulator/Regulator.Core
  - Link: System SW/Linux/drivers/regulator/PMIC_w_MAX8997
- Kernel documents
  - Link: Documentation/power/regulator/*.txt
- The driver should provide
  - Charger's callbacks
    - Enable: charging is permitted (charger decides whether to charge or not)
      - If the battery is full or external power source does not exist, charger (either the driver or the chip) will not charge.
    - Disable: charging is prohibited
      - Even if the charger (driver & chip) thinks it can charge, it must stop charging.
    - Is_enabled: shows the Enable/Disable state

Related Framework: UEVENT

UEVENT Notify

- Kernel documents
  - Link: Documentation/kobject.txt
- Notifies an event to a user process.
  - Use /sys/class/power_supply/[CM-Name]/ to get the notification
    - CM-Name: the name of an instance of CM. (default = "battery")
- Uevent is generated when
  - Recharge starts after once fully-charged.
  - Health monitoring stops chargers.
  - Health monitoring resumes chargers after a stop.
  - Battery is fully charged.
  - Battery is inserted / removed.
  - An IRQ registered for CM by a user is invoked.

  ➔ Every event invokes uevent notification.

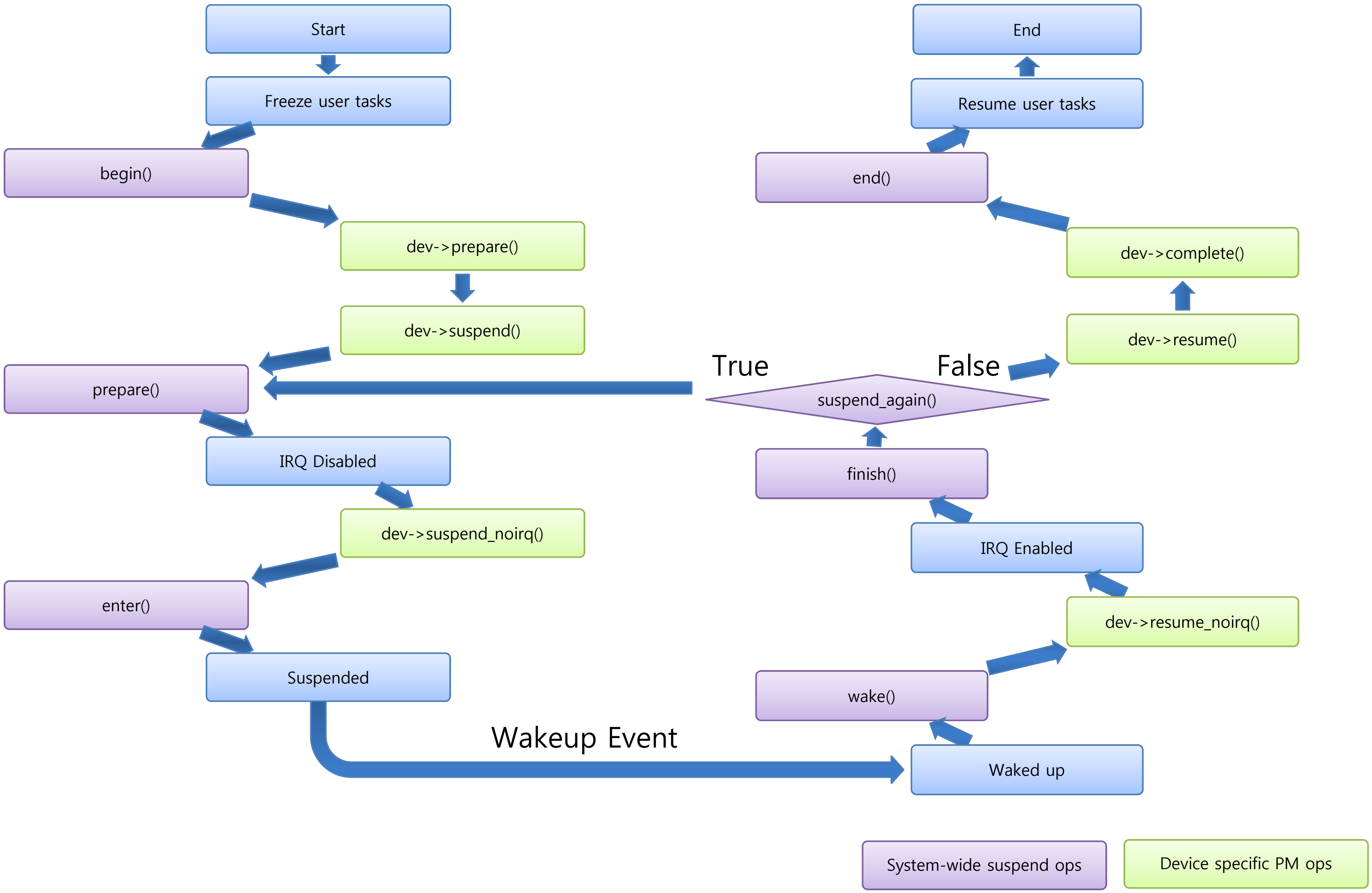
  ➔ Read files in /sys/class/power_supply[CM-Name]/* to see the status in detail.

Related Framework: Suspend_Again (platform_suspend_ops)

suspend_again callback of platform_suspend_ops (#include <linux/suspend.h>)

- SPS documents
  - Link: System SW/Linux/kernel/power/suspend
- Kernel
  - Applied at Linux 3.0
- In order to provide
  - Kernel-side in-suspend polling (battery health monitoring)
  - Execute a delayed-work at the scheduled time while suspended.
    - Check the battery voltage 30s after being fully charged.
  - Why we need this additional callback, suspend_again: [LINK]

Related Framework: Suspend Sequence w/ Suspend_Again



MyungJoo Ham

Related Framework: Why suspend-again ops is added?

- Issue of kernel in-suspend polling
  - There is NO kernel service for periodic wakeup/suspend-again
  - After a wakeup, userspace takes the control. (kernel cannot decide to enter suspend again)
- Solutions
  - Add suspend-again API
    (look at SPS Document, System SW/Linux/kernel/power/suspend)
    - Poll sensors with periodic wakeup/suspend-again.
  - Change H/W to create interrupts for critical events (high/low temperature)
    - Not supported in most H/W
    - Kernel only needs to handle the interrupts

Related Framework: RTC

Real Time Clock (RTC)

- Kernel documents
  - Link: Documentation/rtc.txt
- RTC is required to provide in-suspend monitoring
  - Setup a timer to wakeup from suspend.
- Users should provide a callback to setup an alarm to wakeup from suspend after a specified time. [Link: the RTC-related interface]
- An alternative method.
  - Users may provide a name of RTC (e.g., /dev/rtc0, /dev/rtc1, …) and let CM handle if the RTC's AIE (alarm interrupt enable) can wake up.
  - As of 2011/6/8, RTC AIE Wakeup is being tested and debugged.

Related Framework: HWMON

HWMON (HW Monitor) Framework with

- Kernel documents
  - Link: Documentation/hwmon/sysfs-interface
  - Link: Documentation/hwmon/ntc
    - A common driver for thermistors
- The HWMON driver should provide
  - Ambient temperature
  - OR, battery temperature
- Note: Accessing HWMON is indirect in current revision.
  - Board file (platform) provides a callback that accesses HWMON to Charger-Manager.
  - The callback may be using non standard HWMON driver.
- Accessing HWMON directly is being considered.
  - Then, the board file (platform) will provide the name of HWMON sensor to Charger-Manager
  - Standard HWMON driver should be used.
    - Lm_sensors: Linux hardware monitoring (Link)

Appendix: The Interface in Detail

Interfaces: For Board: Global CM Data (1/2)

The platform (board) should provide global Charger-Manager data

- Describe struct charger_global_desc
  - Name of rtc device
    - char *rtc;
  - Determine if the wakeup timer is the only wakeup reason; if there are other wakeup sources, suspend_again should stop.
    - bool (*is_wktimer_only_wkreason)(void);
  - If true, CM does not rely on jiffies during suspend.
    - bool assume_timer_stops_in_suspend;
- Then, provide it by calling charger_manager(struct charger_global_desc *);

Interfaces: For Board: Global CM Data (2/2)

The platform (board) should use the suspend_again ops of CM in its platform_suspend_ops either by

- Use suspend_again of CM directly
  - E.g.,
    - suspend_ops.suspend_again = cm_suspend_again;
    - suspend_set_ops(&suspend_ops);
- Or, call suspend_again of CM in its own suspend_again ops.

- Note that the CM's suspend_again is cm_suspend_again in <linux/power/charger-manager.h>.

Interfaces: For Board: CM Data for Each Charger

Each battery should provide struct charger_desc with

- psy_name; /* The name of the battery. “battery” is used if NULL */
- polling_interval_ms; /* CM polling interval should be shorter than this */

- Battery-Full and recharging condition
    - fullbatt_vchkdrop_ms; /* Check voltage drop xx ms after Battery-Full */
    - fullbatt_vdhkdrop_uV; /* If the voltage drop is larger than xx uV, recharge */
    - fullbatt_uV; /* Battery-Full voltage */
- About chargers
    - psy_charger_stat; /* Array of PSC names ending with NULL */
    - charger_regulators; /* Array of regulator names ending with NULL */
    - int (*charger_enable)(bool); /* Controls chargers if charger_regulators is NULL */
- About fuel-gauge
    - psy_fuel_gauge; /* PSC name of fuel gauge */
- About interrupts
    - irq_batt_full /* Array of Battery-Full interrupts ending with NULL */
    - irq_batt_out /* Array of Battery-Removed interrupts ending with NULL */
    - irq_misc /* Array of miscellaneous interrupts ending with NULL */
- Battery health monitoring
    - int (*is_temperature_error)(int *mC) /* A callback to monitor the temperature */

Interfaces: Other In-Kernel APIs

The following APIs are provided to other modules in Kernel.

- struct charger_manager *get_charger_manager(char *psy_name);
  - Get a charger_manager pointer with its name.
- void notify_cm_battery_full(struct charger_manager *cm);
  - Notify Battery-Full event to the CM.
- void notify_cm_battery_out(struct charger_manager *cm);
  - Notify Battery-Removed event to the CM.
- void notify_cm_battery_misc(struct charger_manager *cm, char *msg);
  - Notify miscellaneous events to the CM.
- The notify functions are used when it does not generate an interrupt.
  - In such a case, the events will not be detected in suspend.
  - If the HW supports every interrupt described in the previous slides, they are not needed.

Interfaces: For User Processes (1/2)

SYSFS Location: /sys/class/power_supply/## (default = battery)

- Power-Supply-Class standard interface
  - Always provided:
    - capacity: 0 ~ 100 in percent. Represents the remaining capacity of the battery.
    - charge_full: 1 if full-charged. 0 if not full
    - health: Good / Overheat / Cold
    - online: 1 if external power source is available. 0 if not.
    - present: 1 if battery is present. 0 if not.
    - status: Charging / Not charging (charger is connected, but not charging) / Discharging (charger is not connected)
    - voltage_now: battery voltage in uV
  - Optional:
    - charge_now: charging current in uA
    - current_now: battery current in uA
    - temp_ambient: (exists only if ambient temperature is measured)
      - Ambient temperature in 1/10 of centigrade.
    - temp: (exists only if battery temperature is measured)
      - Battery temperature in 1/10 of centigrade.

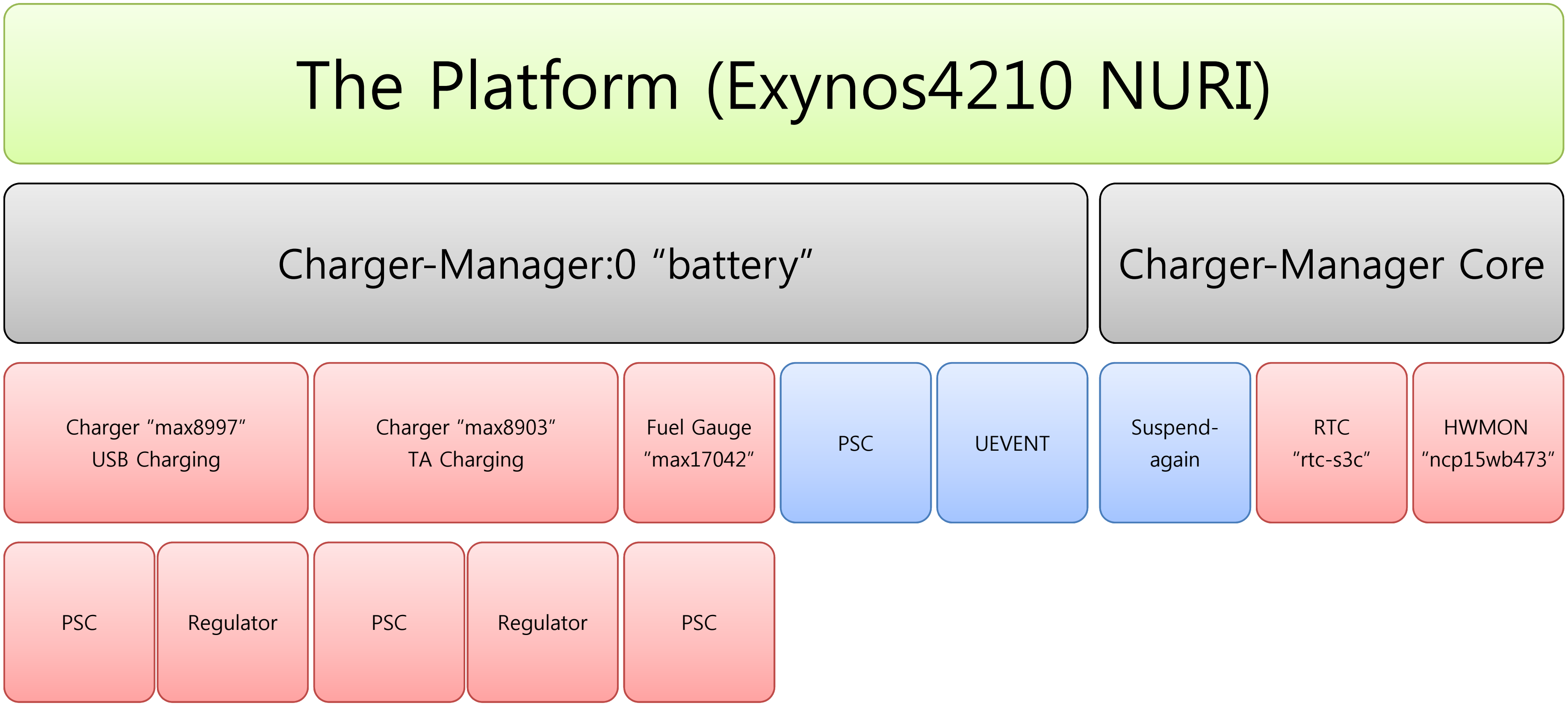
Interfaces: For User Processes (2/2)

SYSFS Location: /sys/class/power_supply/## (default = battery)

- UEVENT notification
  - /sys/class/power_supply/##/uevent

Appendix: Usage Example

Usage Example: Environment at a Glance



The Platform (Exynos4210 NURI)

Charger-Manager:0 “battery”

Charger-Manager Core

Charger “max8997”
USB Charging

Charger “max8903”
TA Charging

Fuel Gauge
”max17042”

PSC

UEVENT

Suspend-
again

RTC
”rtc-s3c”

HWMON
”ncp15wb473”

PSC

Regulator

PSC

Regulator

PSC

Usage Example: Kernel for Exynos4210-NURI

The code is opened to general public under GPL2. (git.infradead.org)

The Platform (board file)

- Platform info about chargers (CM): arch/arm/mach-exynos4/charger-nuri.c
- General platform info: arch/arm/mach-exynos4/mach-nuri.c

The battery #1 of 1 (Charger-Manager:0)

- Described in arch/arm/mach-exynos4/charger-nuri.c
- Two Chargers (MAX8997 and MAX8903)
- Name = “battery”
- Polling Interval = 30 s
- Force recharging if the voltage drops 50mV or more 30s after fully recharged.
- Stops charging if charging current < 50mA.

Usage Example: Kernel for Exynos4210-NURI

Charger #1 of 2: MAX8997

- Platform info: arch/arm/mach-exynos4/max8997-nuri.c
- Main driver: drivers/mfd/max8997.c
- IRQ driver: drivers/mfd/max8997-irq.c
- Regulator driver: drivers/regulator/max8997.c
- Regulator sysfs: /sys/class/regulator.#/
    - Find with "$ grep –H "^CHARGER$"/sys/class/regulator/regulator.*/name" at runtime.
- PSC driver: drivers/power/max8997.c
- PSC sysfs: /sys/class/power_supply/max8997_pmic/

Charger #2 of 2: MAX8903

- Platform info: arch/arm/mach-exynos4/mach-nuri.c
- Driver: drivers/power/max8903_charger.c
- PSC sysfs: /sys/class/power_supply/max8903_charger/
- Regulator sysfs: /sys/class/regulator.#/
    - Find with "$ grep -H "^VOUT_CHARGER$"/sys/class/regulator/regulator.*/name" at runtime.

Usage Example: Kernel for Exynos4210-NURI

Fuel-Gauge: MAX17042

- Platform info: arch/arm/mach-exynos4/mach-nuri.c
- Driver: /drivers/power/max17042_battery.c
- PSC sysfs: /sys/class/power_supply/max17042_battery/

RTC: RTC-S3C (Samsung SoC RTC)

- Platform info: arch/arm/mach-exynos4/mach-nuri..
- Driver: drivers/rtc/rtc-s3c.c
- Sysfs: /sys/class/rtc/rtc#/
  - Find with "$ grep "^s3c$" /sys/class/rtc/rtc*/name" at runtime

HWMON/Thermistor: NCP15WB473

- Platform info: arch/arm/mach-exynos4/mach-nuri.c
- Callback definitions: arch/arm/mach-exynos4/charger-nuri.c
- Driver: drivers/hwmon/ntc.c
- Sysfs: /sys/class/hwmon/hwmon#/
  - Find with "# grep -H "^ncp15wb473$" /sys/class/hwmon/hwmon*/device/name" at runtime

Usage Example: Kernel for Exynos4210-NURI

Power-Supply-Class provided by the Charger-Manager

- Located at /sys/class/power_supply/battery/

Appendix: Related Work (History)

Related Work (History)

- Generally, Linux kernel community considers monitoring and controlling chargers and batteries to be a “USERLAND” job, not kernel job.
  - Thus, no appropriate in-kernel solution, yet.
  - However, userland alone cannot address the whole charging issues
    - Chargers need to monitor the battery health (temperature) while the system is suspended. [Link: Discussion in Linux Power-Management Mailing List]
      - ➔ Requires the kernel to monitor and control the chargers.
    - As mentioned in [Slide: Motivation]
- /drivers/power/pda_power.c (mainline. Since 2007)
  - Monitor connection status (USB/AC) with polling and/or interrupts
  - Limited functionality. Lacks of:
    - In-suspend monitoring
    - Battery health monitoring (temperature)
    - Multiple chargers activated at the same time
    - Voltage checks
    - Chargers other than USB/AC (e.g., wireless, solar cell, …)

Related Work (History)

- /drivers/power/s5pc110_battery.c (Galaxy S/Tab kernel hack)
  - Designed only for a specific system (and specific service providers)
- /arch/arm/plat-samsung/sec-charger.c (2.6.36 SLP kernel hack)
  - Requires hacks in PM-Suspend.
  - Only for Samsung-SoC (difficult to be up-streamed)
  - Basis of the Charger-Manager framework
    - Support standard PMIC, Fuel-gauge drivers
    - Support multiple chargers
    - Being replaced with Charger-Manager currently.

Appendix: References

References

- Related Linux Kernel Mailing Lisst
  - Linux Kernel Mailing List (the main)
    - http://www.tux.org/lkml/
  - Linux Kernel Power Management
    - https://lists.linux-foundation.org/mailman/listinfo/linux-pm
  - Linux ARM Kernel
    - http://lists.arm.linux.org.uk/mailman/listinfo/linux-arm-kernel/
  - Linux Hardware Monitoring Mailing List
    - http://lists.lm-sensors.org/mailman/listinfo/lm-sensors
  - RTC-Linux
    - http://groups.google.com/group/rtc-linux

References

- Related Linux kernel discussion & patches in progress affecting the Charger-Manager implementation.
  - Suspend-Again API
    - [RFC Patch v1] PM / Core: suspend_again cb for syscore_ops
      - Agreed on the general need for suspend_again.
    - [RFC Patch v2] PM / Core: suspend_again callback for device PM
      - Why suspend-to-RAM is special for suspend_again.
      - Where suspend_again should be located in the suspend sequence [slide].
      - About the side effects of suspend_again.
      - Why userspace cannot manage charging?
    - [RFC Patch v3] PM / Core: suspend_again callback for suspend_ops
      - Minor style issues
    - [Patch v4] PM / Core: suspend_again callback for suspend_ops
      - About suspend_again helpers: dpm_partial_resume & dpm_partial_suspend
        - Requires further survey.

References

- Related Linux kernel discussion & patches in progress affecting the Charger-Manager implementation.
  - RTC PIE Interface
    - RTC: Selectively enable PIE-Hrtimer emulation
      - Agreed on the general need for non-emulated PIE support.
      - Requires further survey on the interface.
    - RTC: Modify PIE interface to allow longer periods. (Not submitted)
      - Requires further survey on the interface.

References

- Kernel Documents & Source Codes
  - Charger-Manager Framework (not public yet. available at party git server)
    - Documentation/power/charger-manager.txt
    - Header: include/linux/power/charger-manager.h
    - Code: drivers/power/charger-manager.c
    - Example: arch/arm/mach-s5pv310/charger-slp7.c
  - Regulators (implements charger enable/disable)
    - Documentation/power/regulator/*.txt
    - Header: include/linux/regulator/driver.h
    - Example: drivers/regulator/max8997.c
  - Power-Supply-Class
    - Documentation/power/power-supply-class.txt
    - Header: include/linux/power_supply.h
    - Example: drivers/power/max17042_battery.c

References

- Kernel Documents & Source Codes
  - HWMON (Lm_sensors) Interface
    - Documentation/hwmon/sysfs-interface
    - Documentation/hwmon/ntc
    - Header: include/linux/hwmon-sysfs.h
    - Example: drivers/hwmon/ntc.c
  - RTC
    - Documentation/rtc.txt
    - Header: include/linux/rtc.h, include/linux/rtc/rtc-s3c.h
    - Example: drivers/rtc/rtc-s3c.c
  - UEVENT
    - Documentation/kobject.txt

References: Images

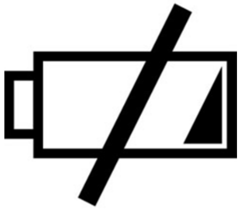
- http://www.mediamacro.com/sponsored/samsung-galaxy-note/

- http://www.iphonespies.com/apple-news/apple-investigating-iphone-fires-and-explosion-reports/

- Microsoft Office Clip Art

- http://www.wholesale-electrical-electronics.com/p-0-5-1-2w-linear-power-adapters-ac-ac-ac-dc-852025.html (edited)

- http://www.clker.com/clipart-12836.html



- http://www.omnimaga.org/index.php?topic=6143.0 (edited)