Charger Manager

C. 2011 MyungJoo Ham <myungjoo.ham@samsung.com>, GPL

Charger Manager provides in-kernel battery charger management that requires temperature monitoring during suspend-to-RAM state and where each battery may have multiple chargers attached and the userland wants to look at the aggregated information of the multiple chargers.

Charger Manager is a platform_driver with power-supply-class entries. An instance of Charger Manager (a platform-device created with Charger-Manager) represents an independent battery with chargers. If there are multiple batteries with their own chargers acting independently in a system, the system may need multiple instances of Charger Manager.

1. Introduction

Charger Manager supports the following:

- Support for multiple chargers (e.g., a device with USB, AC, and solar panels)

  A system may have multiple chargers (or power sources) and some of they may be activated at the same time. Each charger may have its own power-supply-class and each power-supply-class can provide different information about the battery status. This framework aggregates charger-related information from multiple sources and shows combined information as a single power-supply-class.

- Support for in suspend-to-RAM polling (with suspend_again callback)

  While the battery is being charged and the system is in suspend-to-RAM, we may need to monitor the battery health by looking at the ambient or battery temperature. We can accomplish this by waking up the system periodically. However, such a method wakes up devices unnecessarily for monitoring the battery health and tasks, and user processes that are supposed to be kept suspended. That, in turn, incurs unnecessary power consumption and slow down charging process. Or even, such peak power consumption can stop chargers in the middle of charging (external power input < device power consumption), which not only affects the charging time, but the lifespan of the battery.

  Charger Manager provides a function "cm_suspend_again" that can be used as suspend_again callback of platform_suspend_ops. If the platform requires tasks other than cm_suspend_again, it may implement its own suspend_again callback that calls cm_suspend_again in the middle. Normally, the platform will need to resume and suspend some devices that are used by Charger Manager.

- Support for premature full-battery event handling

  If the battery voltage drops by "fullbatt_vchkdrop_uV" after "fullbatt_vchkdrop_ms" from the full-battery event, the framework restarts charging. This check is also performed while suspended by setting wakeup time accordingly and using suspend_again.

- Support for uevent-notify

  With the charger-related events, the device sends notification to users with UEVENT.

2. Global Charger-Manager Data related with suspend_again

In order to setup Charger Manager with suspend-again feature (in-suspend monitoring), the user should provide charger_global_desc with setup_charger_manager(*struct charger_global_desc* *). This charger_global_desc data for in-suspend monitoring is global as the name suggests. Thus, the user needs to provide only once even if there are multiple batteries. If there are multiple batteries, the multiple instances of Charger Manager share the same charger_global_desc and it will manage in-suspend monitoring for all instances of Charger Manager.

The user needs to provide all the three entries to *struct charger_global_desc* properly in order to activate in-suspend monitoring:

*char *rtc_name;*

The name of rtc (e.g., "rtc0") used to wakeup the system from suspend for Charger Manager. The alarm interrupt (AIE) of the rtc should be able to wake up the system from suspend. Charger Manager saves and restores the alarm value and use the previously-defined alarm if it is going to go off earlier than Charger Manager so that Charger Manager does not interfere with previously-defined alarms.

*bool (*rtc_only_wakeup)(void);*

This callback should let CM know whether the wakeup-from-suspend is caused only by the alarm of "rtc" in the same struct. If there is any other wakeup source triggered the wakeup, it should return false. If the "rtc" is the only wakeup reason, it should return true.

bool assume_timer_stops_in_suspend;

if true, Charger Manager assumes that the timer (CM uses jiffies as timer) stops during suspend. Then, CM assumes that the suspend-duration is same as the alarm length.

3. How to setup suspend_again

Charger Manager provides a function "extern bool cm_suspend_again(void)". When cm_suspend_again is called, it monitors every battery. The suspend_ops callback of the system's platform_suspend_ops can call cm_suspend_again function to know whether Charger Manager wants to suspend again or not. If there are no other devices or tasks that want to use suspend_again feature, the platform_suspend_ops may directly refer to cm_suspend_again for its suspend_again callback.

The cm_suspend_again() returns true (meaning "I want to suspend again") if the system was woken up by Charger Manager and the polling (in-suspend monitoring) results in "normal".

4. Charger-Manager Data (struct charger_desc)

For each battery charged independently from other batteries (if a series of batteries are charged by a single charger, they are counted as one independent battery), an instance of Charger Manager is attached to it. The following

struct charger_desc elements:

*char *psy_name;*
: The power-supply-class name of the battery. Default is "battery" if psy_name is NULL. Users can access the psy entries at "/sys/class/power_supply/[psy_name]/".

enum polling_modes polling_mode;

- CM_POLL_DISABLE:
  - do not poll this battery.
- CM_POLL_ALWAYS:
  - always poll this battery.
- CM_POLL_EXTERNAL_POWER_ONLY:
  - poll this battery if and only if an external power source is attached.
- CM_POLL_CHARGING_ONLY:
  - poll this battery if and only if the battery is being charged.

unsigned int fullbatt_vchkdrop_ms; / unsigned int fullbatt_vchkdrop_uV;
: If both have non-zero values, Charger Manager will check the battery voltage drop fullbatt_vchkdrop_ms after the battery is fully charged. If the voltage drop is over fullbatt_vchkdrop_uV, Charger Manager will try to recharge the battery by disabling and enabling chargers. Recharge with voltage drop condition only (without delay condition) is needed to be implemented with hardware interrupts from fuel gauges or charger devices/chips.

unsigned int fullbatt_uV;
: If specified with a non-zero value, Charger Manager assumes that the battery is full (capacity = 100) if the battery is not being charged and the battery voltage is equal to or greater than fullbatt_uV.

unsigned int polling_interval_ms;
: Required polling interval in ms. Charger Manager will poll this battery every polling_interval_ms or more frequently.

enum data_source battery_present;

- CM_BATTERY_PRESENT:
  - assume that the battery exists.
- CM_NO_BATTERY:
  - assume that the battery does not exists.
- CM_FUEL_GAUGE:
  - get battery presence information from fuel gauge.
- CM_CHARGER_STAT:
  - get battery presence from chargers.

*char **psy_charger_stat;*
: An array ending with NULL that has power-supply-class names of chargers. Each power-supply-class should provide "PRESENT" (if battery_present is "CM_CHARGER_STAT"), "ONLINE" (shows whether an external power source is attached or not), and "STATUS" (shows whether the battery is {"FULL" or not FULL} or {"FULL", "Charging", "Discharging", "NotCharging"}).

*int num_charger_regulators; / struct regulator_bulk_data *charger_regulators;*
: Regulators representing the chargers in the form for regulator framework's bulk functions.

*char *psy_fuel_gauge;*
: Power-supply-class name of the fuel gauge.

*int (*temperature_out_of_range)(int *mC); / bool measure_battery_temp;*
: This callback returns 0 if the temperature is safe for charging, a positive number if it is too hot to charge, and a negative number if it is too cold to charge. With the variable mC, the callback returns the temperature in 1/1000 of centigrade. The source of temperature can be battery or ambient one according to the value of measure_battery_temp.

5. Notify Charger-Manager of charger events: cm_notify_event()

If there is an charger event is required to notify Charger Manager, a charger device driver that triggers the event can call cm_notify_event(psy, type, msg) to notify the corresponding Charger Manager. In the function, psy is the charger driver's power_supply pointer, which is associated with Charger-Manager. The parameter "type" is the same as irq's type (enum cm_event_types). The event message "msg" is optional and is effective only if the event type is "UNDESCRIBED" or "OTHERS".

6. Other Considerations

At the charger/battery-related events such as battery-pulled-out, charger-pulled-out, charger-inserted, DCIN-over/under-voltage, charger-stopped, and others critical to chargers, the system should be configured to wake up. At least the following should wake up the system from a suspend: a) charger-on/off b) external-power-in/out c) battery-in/out (while charging)

It is usually accomplished by configuring the PMIC as a wakeup source.