Fuel-gauge and BD72720 docs

This document has not been thoroughly validated. There may be mistakes. It is intended to be a starting point for understanding the driver(s), but the reader should eventually verify the operation and concepts from the code. All improvements to this document are welcome. Please, send the suggestions to matti.vaittinen@fi.rohmeurope.com.

The BD72720 is a Power Management IC for a customer project. Drivers are mostly trivial stuff, except:

- The fuel-gauge which is implemented in-kernel, and which needs the battery details to be given
- Regulator sub-run-level control.

The development version of the BD72720 Linux driver can be found from:

https://github.com/RohmSemiconductor/Linux-Kernel-PMIC-Drivers/tree/bd72720-on-6.6

(The most recent tag being

https://github.com/RohmSemiconductor/Linux-Kernel-PMIC-Drivers/tree/bd72720-drop1-fully_untested-on6.6-v1.1

as writing of this)

Please note that the branch is not stable, it may get rebased and commits may change. The git tags however are intended to be permanent. Tags relevant to the BD72720 should begin with prefix ‘bd72720-’

Sub-run-level control:

The BUCK1 and LDO1 can be configured to either be controlled in usual way (voltage and enable/disable state can be selected individually for both. At RUN state they are changed via I2C register writes. Low power states (IDLE / DEEP IDLE / SUSPEND) can have predefined voltages / enable states which become effective when the PMIC transitions to one of the low-power states). This is same as the other BUCKs and LDOs.

In addition to this usual control, the BD72720 (and BD71828) has a ‘sub run level’ feature. The PMIC can be configured to use sub-run levels RUN0, RUN1, RUN2, and RUN3. In this mode the voltage for each of the sub-run levels can be pre-configured (princible is same as with voltages for IDLE / DEEP IDLE / SUSPEND states). Then the PMIC RUN state can be changed using two GPIOs.

The Linux regulator subsystem is designed to treat regulators as individual entities* and does not support controlling voltages for multiple regulators at once. Thus the sub-run level control is implemented exporting a custom in-kernel API from the BD72720 regulator driver.

In-kernel API

The BD72720 regulator driver exports functions:

int bd71828_set_runlevel_voltage(struct regulator *regulator, unsigned int uv, unsigned int level)

bd71828_set_runlevel_voltage - Changes the run-level voltage for given regulator

** @regulator: pointer to regulator for which run-level voltage is to be changed*

** @uv: New voltage for run-level in micro volts*

** @level: run-level for which the voltage is to be changed*

int bd71828_set_runlevel(struct regulator *regulator, unsigned int level)

bd71828_set_runlevel - Changes the run-level

** @regulator: pointer to one of the BD72720 regulators obtained by a call to regulator_get*

** @level: New run-level the system should enter*

** Changes the system to run-level which was given as argument. This*

** operation will change state of all regulators which are set to be*

** controlled by run-levels. Note that 'regulator' must point to a*

** regulator which is controlled by run-levels.*

int bd71828_get_runlevel(struct regulator *regulator, unsigned int *level)

bd71828_get_runlevel - get the current system run-level.

** @regulator: pointer to one of the BD71828 regulators obtained by a call to regulator_get*

** @level: Pointer to value where current run-level is stored*

** Returns the current system run-level. Note that 'regulator' must*

** point to a regulator which is controlled by run-levels.*

sysfs ABI

A file *'runlevel'* is created under the sysfs. Reading the file will provide the current system run-level. The system can be changed to a new run-level by writing the new target run-level to this file.

In-kernel fuel-gauge (simple-gauge)

ROHM has developed a fuel-gauge algorithm which can estimate the state of charge (remaining battery capacity) by utilizing a coulomb-counter implemented in the PMIC. The idea of a coulomb-counter is to periodically measure current flowing from/to the battery (voltage over known "sense resistor"), and accumulate these values. Thus, in theory, when the initial amount of energy stored in the battery is known, we can know the amount of energy left in the battery by subtracting the amount of drawn current from the battery since the beginning. The standard way to inform the current state of charge is to tell the persentage of the current capacity (from full battery capacity).

There are several caveats. For example:

- Full battery capacity varies depending on the conditions.
    - Temperature
    - Aging
- Systematic errors in current measurement will accumulate to large errors during time

    - ADC offset
- Knowing the initial battery state

In order to get reliable information about battery state of charge, one needs know the battery details and also mitigate all of the above. The fuel-gauge algorithm implemented in the in-kernel fuel-gauge driver attempts to do this mitigation. Driver(s) do also inform the user-space about the state of charge and other charging / power supplying related properties (like voltages, currents and whether the device is being charged). The ROHM fuel-gauge is implemented in two Linux drivers. drivers/power/supply/simple-gauge.c (which contains the generic logic for the coulomb counter reading and applying the error corrections) and drivers/power/supply/bd71827-power.c which contains the PMIC IC specific hardware accesses.

These drivers need to get the battery and system specific details. Originally the details have been provided as module parameters. Since the use of module parameters is generally discouraged (in upstream Linux), and introducing new module parameters is not well received, the current implementation relies heavily on device-tree.

Following device-tree properties are needed for fuel-gauge:

This section lists only the properties required for fuel-gauging. The rest of the PMIC drivers need more information from the device-tree. Usual binding documentation lists those. **TODO:** The binding docs aren’t a great documentation. It is more or less a scattered around collection of all known device-tree properties. There is no good mapping explaining which of the properties are required on a specific system. (For example, the BD71828 and BD72720 PMIC documentation only lists the properties which are *specific* for these ICs, and then only refer to a generic documentation which lists a lot of bindings. There, however, is no proper document telling which of the generic properties are supported or required on BD71828 or BD72720, or, TBH, on any other device from any other manufacturer).

This page attempts to list charging related properties used by BD71828 and BD72720 charging / fuel gauging.

In the charger (bd72720) device-tree node:

compatible = “rohm,bd72720“

Measuring the current is done by measuring voltage drop over a known sense resistor. The sense resistor size must be told using:

rohm,charger-sense-resistor-milli-ohms = <milli ohms>;

NOTE: The original BD71828 support accidentally introduced

rohm,charger-sense-resistor-ohms

property. This is not feasible as the sense resistors are typically less than 1 Ohm. Usually between 10 and 50 milli ohms.

monitored-battery - reference to the battery node containing the battery properties

In the battery node:

(Documentation for battery-node bindings is in Documentation/devicetree/bindings/power/supply/battery.yaml) Following are used by the fuel-gauge:

compatible = “simple-battery”

device-chemistry = (currently not required but prints a warning if not given) the battery chemistry. Supported values are:

- “nickel-cadmium”
- “nickel-metal-hydride”
- “lithium-ion”
- “lithium-ion-polymer”
- “lithium-ion-iron-phosphate”
- “lithium-ion-manganese-oxide”

TODO: Should SW require certain type (or exclude some of these?)

Tables of “OCV”/“state of charge” -value pairs

The battery voltage can be used as a decent indicator for a state of charge, when a battery is relaxed (there has been no big current in/out flow in a while) and no (significant) current is drawn from a battery. Battery manufacturers may provide OCV tables which can be used to map the OCV => SOC. This has some issues, like the requirement that the battery must be relaxed and no current should be drawn. The BD72720 has some built-in mechanisms to detect relaxed battery. BD72720 may use OCV to correct coulomb counter value (based on a measured battery voltage at relaxed state) in order to mitigate the error accumulation. BD72720 does also estimate the initial state (after startup) based on OCV values.

ocv-capacity-celsius - Temperature(s) corresponding the OCV tables (below) *ocv-capacity-table-0, ocv-capacity-table-1, ocv-capacity-table-2, …*

OCV tables containing "open circuit voltage" and corresponding "battery capacity percent" - value pairs for a temperature. The integer at the end of the property name corresponds to index of ocv-capacity-celsius table, where the temperature where this table is accurate is stored.

Capacity of full battery

BD72720 needs to know the capacity of full battery to be able to compute the energy left in battery. The ideal capacity of battery is used as a basis of calculating the amount of energy the battery can contain. (Impact of the temperature and aging may be estimated).

charge-full-design-microamp-hours = ideal full battery capacity in uAh. (from battery manufacturer)

Maximum and minimum battery voltages are used to compute a "low voltage threshold" (if not explicitly set) as well as to be able to handle situations where the battery SOC or voltage are abnormally high / low. The minimum voltage is also used by the ROHM zero-correct algorithm which operates based on the VDR tables.

voltage-max-design-microvolt = Maximum battery voltage

voltage-max-design-microvolt = Minimum battery voltage. This is the voltage when battery is considered empty by the system and causes a shutdown.

Impact of aging (Optional)

The BD72720 has a separate counter to count charged uAhs. This is used by the drivers to compute charge cycles, and to estimate the impact of aging to the battery capacity. NOTE: The cycle count is not stored by the PMIC driver because the PMIC has no permanent storage. It is the responsibility of the user to store the cycle count.

degrade-cycle-microamp-hours - dropped capacity / charging cycle in uAh

Impact of the temperature (Optional)

The simple-gauge supports estimating the impact of the temperature (to the battery capacity) by using temperature ranges where capacity drops linearly. The temperature impact to capacity is not really linear, but by using many enough, short enough, temperature areas, we will get a decent estimate. It is also possible to not use the temperature degradation tables but to trust the ROHM VDR algorithm to correct the estimates when the SOC is closing to 0%

temp-degrade-table - array of value triplets. First value is “capacity change / degree C” when temperature differs from given ‘set-point’ temperature. Second value is the “capacity degradation at given ‘set-point’ temperature”. Third value being the ‘set-point’ temperature.

VDR zero-correction (Optional)

Finally, ROHM has developed a “zero-correction” algorithm, which uses a voltage drop rate (VDR) tables defined by ROHM for a sample battery. These battery specific tables should be obtained from the ROHM. Using the VDR is optional but can improve accuracy. The VDR tables are given using following properties:

rohm,volt-drop-temp-millikelvin - The temperatures corresponding to “very-low”, “low”, “normal” and “high” VDR tables. In kelvin.

rohm,volt-drop-soc - The SoC values on tenths of percents (1000 <=> 100%)

- these values should correspond to zero-correction voltages given in “*volt-drop-*-temp-microvolt*”. Eg, there should be same number of entries in all the tables, and index <n> in *volt-drop-*-temp-microvolt* table should correspond to index <n> in *rohm,volt-drop-soc* table.

rohm,volt-drop-high-temp-microvolt - VDR values for ‘high’ temperature

rohm,volt-drop-normal-temp-microvolt - VDR values for ‘normal’ temperature

rohm,volt-drop-low-temp-microvolt - VDR values for ‘low’ temperature

rohm,volt-drop-very-low-temp-microvolt - VDR values for ‘very-low’ temperature

Current/voltage limits for charging phases

The driver also supports setting the current/voltage limits of different charging phases from the device-tree. This, however, may cause problems in cases where the charging is done prior to the software is running and setting the limits. Please ask about the device-tree properties if you consider the approach of having software defined charging profiles appropriate for your system.

Values propagated through the Linux power-supply class

The simple-gauge and BD72720-charger drivers do provide information to users via the Linux power-supply class. Following power-supply class properties are computed provided to users via the standard power-supply interfaces:

BD72720 charger driver

charger (visible via simple-gauge):

- POWER_SUPPLY_PROP_ONLINE
- POWER_SUPPLY_PROP_VOLTAGE_NOW

battery:

- POWER_SUPPLY_PROP_STATUS
- POWER_SUPPLY_PROP_HEALTH
- POWER_SUPPLY_PROP_CHARGE_TYPE
- POWER_SUPPLY_PROP_ONLINE
- POWER_SUPPLY_PROP_PRESENT
- POWER_SUPPLY_PROP_VOLTAGE_NOW
- POWER_SUPPLY_PROP_TECHNOLOGY
- POWER_SUPPLY_PROP_CURRENT_AVG
- POWER_SUPPLY_PROP_CURRENT_NOW
- POWER_SUPPLY_PROP_VOLTAGE_MAX
- POWER_SUPPLY_PROP_VOLTAGE_MIN
- POWER_SUPPLY_PROP_CURRENT_MAX

simple-gauge:

- POWER_SUPPLY_PROP_CYCLE_COUNT (also writable so cycle count preceding reset can be restored)
- POWER_SUPPLY_PROP_CAPACITY
- POWER_SUPPLY_PROP_CHARGE_FULL_DESIGN
- POWER_SUPPLY_PROP_CHARGE_FULL
- POWER_SUPPLY_PROP_CHARGE_NOW
- POWER_SUPPLY_PROP_TEMP

Battery internal impedance

A new thing the BD72720 adds is battery internal impedance check. This is not completely evaluated yet, and the BD72720 or simple-gauge do not support it as of now. This might be relevant in the future, so here’s small bits and pieces I’ve found out.

Sometime ago I was reviewing some additions Mr. Linus W did to the power-supply class. He added excellent comments in the linux/power_supply.h to the power_supply_battery_info structure’s (and the power_supply_maintenance_charge_table structure’s) documentation - I’ll cite some of this here:

```
* DETERMINING BATTERY CAPACITY:
 *
 * Several members of the struct deal with trying to determine the remaining
 * capacity in the battery, usually as a percentage of charge. In practice
 * many chargers uses a so-called fuel gauge or coloumb counter that measure
 * how much charge goes into the battery and how much goes out (+/- leak
 * consumption). This does not help if we do not know how much capacity the
 * battery has to begin with, such as when it is first used or was taken out
 * and charged in a separate charger. Therefore many capacity algorithms use
 * the open circuit voltage with a look-up table to determine the rough
 * capacity of the battery. The open circuit voltage can be conceptualized
 * with an ideal voltage source (V) in series with an internal resistance
(Ri)
 * like this:
 *
 *      +-------> IBAT >----------------+
 *      |                    ^          |
 *     [ ] Ri                |          |
 *      |                    | VBAT     |
 *      o <----------        |          |
 *     +|           ^        |         [ ] Rload
 *    .---.         |        |          |
 *    | V |         | OCV    |          |
 *    '---'         |        |          |
 *      |           |        |          |
 *  GND +-------------------------------+
 *
 * If we disconnect the load (here simplified as a fixed resistance Rload)
 * and measure VBAT with a infinite impedance voltage meter we will get
 * VBAT = OCV and this assumption is sometimes made even under load, assuming
 * Rload is insignificant. However this will be of dubious quality because
the
 * load is rarely that small and Ri is strongly nonlinear depending on
 * temperature and how much capacity is left in the battery due to the
 * chemistry involved.
 *
 * In many practical applications we cannot just disconnect the battery from
 * the load, so instead we often try to measure the instantaneous IBAT (the
 * current out from the battery), estimate the Ri and thus calculate the
 * voltage drop over Ri and compensate like this:
 *
 *   OCV = VBAT - (IBAT * Ri)
```

So, one way to overcome problems of mitigating the temperature and aging impact, as well as the impact of the load to OCV - we could perhaps use the battery internal impedance.

The BD72720 has following implementation for measuring the battery internal impedance:

At the beginning of the constant voltage charging, the system load is disconnected from the battery and supplied directly by DCIN. The current and the voltage are measured.
Then the charging is stopped for a while, which causes the voltage to drop.

Now, the dropped voltage and current are measured.

The battery internal impedance can then be calculated using the current and voltage differences between the two measurements.

Custom sysfs user-interfaces for charger:

bd72720 driver:

charging - can be used to show / set the charging status. value 1 means “charging enabled”, value 0 is “charging disabled”.