Linux voltage and current regulator framework

About

This framework is designed to provide a standard kernel interface to control voltage and current regulators.

The intention is to allow systems to dynamically control regulator power output in order to save power and prolong battery life. This applies to both voltage regulators (where voltage output is controllable) and current sinks (where current limit is controllable).

Author: Liam Girdwood <lrg@slimlogic.co.uk>

Nomenclature

Some terms used in this document:

- Regulator
  - Electronic device that supplies power to other devices. Most regulators can enable and disable their output while some can control their output voltage and or current.

    Input Voltage -> Regulator -> Output Voltage
- PMIC
  - Power Management IC. An IC that contains numerous regulators and often contains other subsystems.
- Consumer
  - Electronic device that is supplied power by a regulator. Consumers can be classified into two types:-

    Static: consumer does not change its supply voltage or current limit. It only needs to enable or disable its power supply. Its supply voltage is set by the hardware, bootloader, firmware or kernel board initialisation code.

    Dynamic: consumer needs to change its supply voltage or current limit to meet operation demands.
- Power Domain
  - Electronic circuit that is supplied its input power by the output power of a regulator, switch or by another power domain.

    The supply regulator may be behind a switch(s). i.e.:

    ```
    Regulator -+-> Switch-1 -+-> Switch-2 --> [Consumer A]
               |             |
               |             +-> [Consumer B], [Consumer C]
               |
               +-> [Consumer D], [Consumer E]
    ```

    That is one regulator and three power domains:

    - Domain 1: Switch-1, Consumers D & E.
    - Domain 2: Switch-2, Consumers B & C.
    - Domain 3: Consumer A.

    and this represents a "supplies" relationship:

    Domain-1 --> Domain-2 --> Domain-3.

    A power domain may have regulators that are supplied power by other regulators. i.e.:

    ```
    Regulator-1 -+-> Regulator-2 -+-> [Consumer A]
                 |
                 +-> [Consumer B]
    ```

    This gives us two regulators and two power domains:

    - Domain 1: Regulator-2, Consumer B.
    - Domain 2: Consumer A.

    and a "supplies" relationship:

    Domain-1 --> Domain-2
- Constraints
  - Constraints are used to define power levels for performance and hardware protection. Constraints exist at three levels:

Regulator Level: This is defined by the regulator hardware operating parameters and is specified in the regulator datasheet. i.e.

- voltage output is in the range 800mV -> 3500mV.
- regulator current output limit is 20mA @ 5V but is 10mA @ 10V.

Power Domain Level: This is defined in software by kernel level board initialisation code. It is used to constrain a power domain to a particular power range. i.e.

- Domain-1 voltage is 3300mV
- Domain-2 voltage is 1400mV -> 1600mV
- Domain-3 current limit is 0mA -> 20mA.

Consumer Level: This is defined by consumer drivers dynamically setting voltage or current limit levels.

e.g. a consumer backlight driver asks for a current increase from 5mA to 10mA to increase LCD illumination. This passes to through the levels as follows :-

Consumer: need to increase LCD brightness. Lookup and request next current mA value in brightness table (the consumer driver could be used on several different personalities based upon the same reference device).

Power Domain: is the new current limit within the domain operating limits for this domain and system state (e.g. battery power, USB power)

Regulator Domains: is the new current limit within the regulator operating parameters for input/output voltage.

If the regulator request passes all the constraint tests then the new regulator value is applied.

Design

The framework is designed and targeted at SoC based devices but may also be relevant to non SoC devices and is split into the following four interfaces:-

1. Consumer driver interface.

   This uses a similar API to the kernel clock interface in that consumer drivers can get and put a regulator (like they can with clocks atm) and get/set voltage, current limit, mode, enable and disable. This should allow consumers complete control over their supply voltage and current limit. This also compiles out if not in use so drivers can be reused in systems with no regulator based power control.

   See Documentation/power/regulator/consumer.rst

2. Regulator driver interface.

   This allows regulator drivers to register their regulators and provide operations to the core. It also has a notifier call chain for propagating regulator events to clients.

   See Documentation/power/regulator/regulator.rst

3. Machine interface.

   This interface is for machine specific code and allows the creation of voltage/current domains (with constraints) for each regulator. It can provide regulator constraints that will prevent device damage through overvoltage or overcurrent caused by buggy client drivers. It also allows the creation of a regulator tree whereby some regulators are supplied by others (similar to a clock tree).

   See Documentation/power/regulator/machine.rst

4. Userspace ABI.

   The framework also exports a lot of useful voltage/current/opmode data to userspace via sysfs. This could be used to help monitor device power consumption and status.

   See Documentation/ABI/testing/sysfs-class-regulator