NAME

regulator, **regulator_get_by_name**, **regulator_get_by_id**, **regulator_release**, **regulator_get_name**, **regulator_enable**, **regulator_disable**, **regulator_stop**, **regulator_status**, **regulator_get_voltage**, **regulator_set_voltage**, **regulator_check_voltage**, **regulator_get_by_ofw_property** - regulator methods

SYNOPSIS

device regulator
#include <dev/extres/regulator/regulator.h>

int
regulator_get_by_name(*device_t cdev*, *const char *name*, *regulator_t *regulator*);

int
regulator_get_by_id(*device_t cdev*, *device_t pdev*, *intptr_t id*, *regulator_t *regulator*);

int
regulator_release(*regulator_t regulator*);

int
regulator_get_name(*regulator_t regulator*);

int
regulator_enable(*regulator_t reg*);

int
regulator_disable(*regulator_t reg*);

int
regulator_stop(*regulator_t reg*);

int
regulator_status(*regulator_t reg*, *int *status*);

int
regulator_get_voltage(*regulator_t reg*, *int *uvolt*);

int
regulator_set_voltage(*regulator_t reg*, *int min_uvolt*, *int max_uvolt*);

int

regulator_check_voltage(*regulator_t reg, int uvolt*);

int
regulator_get_by_ofw_property(*device_t dev, phandle_t node, char *name, regulator_t *reg*);

DESCRIPTION

The regulator framework allow drivers to enable, disable and change regulator voltage.

RETURN VALUES

All functions returns 0 on success or ENODEV if the regulator or one of its parent was not found.

INTERFACE

regulator_get_by_name(*device_t cdev, const char *name, regulator_t *regulator*)
Resolve a regulator based on its name. All regulators names are unique. This will also increment the refcount on the regulator.

regulator_get_by_id(*device_t cdev, device_t pdev, intptr_t id, regulator_t *regulator*)
Resolve a regulator based on its id. All regulators ids are unique. This will also increment the refcount on the regulator.

regulator_get_by_ofw_property(*device_t dev, phandle_t node, char *name, regulator_t *reg*)
Resolve a regulator based on the fdt property named name. If node is 0 then the function will get the ofw node itself. This will also increment the refcount on the regulator. Returns 0 on success or ENOENT if the ofw property does not exists.

regulator_release(*regulator_t regulator*)
This disables the regulator, decrements the refcount on it and frees the regulator variable passed.

regulator_get_name(*regulator_t regulator*)
Returns the name of the regulator. All regulator names are unique.

regulator_enable(*regulator_t reg*)
Enable the regulator. If the regulator supports a voltage range, the one configured in the hardware will be the output voltage. If the regulator was already enabled by another driver this simply increments the enable counter.

regulator_disable(*regulator_t reg*)
Disable the regulator. If the regulator was also enabled by another driver this simply decrements the enable counter. If the regulator was not previously enabled we will kassert.

regulator_stop(*regulator_t reg*)

Disable the regulator in hardware. This ensures the regulator is disabled even if it was enabled by bootloader. This should not be called on regulator that has previously been enabled by a driver. Returns 0 on success or EBUSY if another consumer enabled it.

regulator_status(*regulator_t reg, int *status*)

Get the hardware status of the regulator. status will contain a bit mask with thoses possible value :

REGULATOR_STATUS_ENABLED

The regulator is enabled.

REGULATOR_STATUS_OVERCURRENT

The hardware reports that too much current is being drawn.

regulator_get_voltage(*regulator_t reg, int *uvolt*)

Get the current voltage set for the regulator in microvolts.

regulator_set_voltage(*regulator_t reg, int min_uvolt, int max_uvolt*)

Change the voltage for the regulator. If a range is acceptable by the hardware or driver different values can be provided as min and max. Returns 0 on success or ERANGE if the regulator doesn't support this voltage range.

regulator_check_voltage(*regulator_t reg, int uvolt*)

Checks if the regulator support the given voltage. Returns 0 on success or ERANGE if the regulator doesn't support this voltage range.

HISTORY

The **regulator** framework first appear in FreeBSD 12.0. The **regulator** framework was written by Michal Meloun <*mmel@FreeBSD.org*>. The **regulator** manual page was written by Emmanuel Vadot <*manu@FreeBSD.org*>.