# General description of the CPUFreq core and CPUFreq notifiers

Authors:

- Dominik Brodowski <linux@brodo.de>
- David Kimdon <dwhedon@debian.org>
- Rafael J. Wysocki <rafael.j.wysocki@intel.com>
- Viresh Kumar <viresh.kumar@linaro.org>

## 1. General Information

The CPUFreq core code is located in drivers/cpufreq/cpufreq.c. This cpufreq code offers a standardized interface for the CPUFreq architecture drivers (those pieces of code that do actual frequency transitions), as well as to "notifiers". These are device drivers or other part of the kernel that need to be informed of policy changes (ex. thermal modules like ACPI) or of all frequency changes (ex. timing code) or even need to force certain speed limits (like LCD drivers on ARM architecture). Additionally, the kernel "constant" loops_per_jiffy is updated on frequency changes here.

Reference counting of the cpufreq policies is done by cpufreq_cpu_get and cpufreq_cpu_put, which make sure that the cpufreq driver is correctly registered with the core, and will not be unloaded until cpufreq_put_cpu is called. That also ensures that the respective cpufreq policy doesn't get freed while being used.

## 2. CPUFreq notifiers

CPUFreq notifiers conform to the standard kernel notifier interface. See linux/include/linux/notifier.h for details on notifiers.

There are two different CPUFreq notifiers - policy notifiers and transition notifiers.

### 2.1 CPUFreq policy notifiers

These are notified when a new policy is created or removed.

The phase is specified in the second argument to the notifier. The phase is CPUFREQ_CREATE_POLICY when the policy is first created and it is CPUFREQ_REMOVE_POLICY when the policy is removed.

The third argument, a `void *pointer`, points to a struct cpufreq_policy consisting of several values, including min, max (the lower and upper frequencies (in kHz) of the new policy).

### 2.2 CPUFreq transition notifiers

These are notified twice for each online CPU in the policy, when the CPUfreq driver switches the CPU core frequency and this change has no any external implications.

The second argument specifies the phase - CPUFREQ_PRECHANGE or CPUFREQ_POSTCHANGE.

The third argument is a struct cpufreq_freqs with the following values:

| | |
|---|---|
| policy | a pointer to the struct cpufreq_policy |
| old | old frequency |
| new | new frequency |
| flags | flags of the cpufreq driver |

## 3. CPUFreq Table Generation with Operating Performance Point (OPP)

For details about OPP, see Documentation/power/opp.rst

dev_pm_opp_init_cpufreq_table -

This function provides a ready to use conversion routine to translate the OPP layer's internal information about the available frequencies into a format readily providable to cpufreq.

**Warning**

Do not use this function in interrupt context.

Example:

```
soc_pm_init()
{
        /* Do things */
        r = dev_pm_opp_init_cpufreq_table(dev, &freq_table);
        if (!r)
                policy->freq_table = freq_table;
        /* Do other things */
}
```

> **Note**
>
> This function is available only if CONFIG_CPU_FREQ is enabled in addition to CONFIG_PM_OPP.

dev_pm_opp_free_cpufreq_table

Free up the table allocated by dev_pm_opp_init_cpufreq_table