Qualcomm Linux Power And Thermal Guide

80-70018-30 AD

April 10, 2025

Contents

1 Power management

Qualcomm® Linux® is enabled with power and thermal features that allow you to monitor and optimize power consumption while maintaining the junction temperature within the limit.

1.1 Overview of power management

This guide helps you to develop energy-efficient products or applications that balance performance and power consumption.

Note: The guide uses the terms low-power state and sleep state interchangeably to indicate the power-saving mechanism.

It includes:

- Application processor subsystem (APSS) architecture
- CPU active power management through CPU frequency scaling mechanisms and its tunables
- CPU idle power management through CPU sleep power states

The features and power states explained here are for reference only.

Note: See Hardware SoCs that are supported on Qualcomm Linux.

1.2 Get started with power optimization

This section explains how to develop software using Qualcomm Linux. Before you begin, set up your infrastructure as described in Qualcomm Linux Build Guide. This guide also provides information about the common build workflow.

Prepare performance build

Preparing the performance build ensures all power features are optimized and functioning correctly for effective power management. Enable SSH in Permissive mode to prepare the performance build. For instructions see Sign in using SSH.

The default Qualcomm build is the performance build. For evaluation of any power measurement, use the performance build.

The performance build uses the following kernel configurations:

```
kernel_platform/kernel/arch/arm64/configs/qcom_defconfig
```

```
kernel_platform/kernel /arch/arm64/configs/qcom_addons.config
```

The kernel configurations are defined in the source code kernel recipe at:

```
layer/meta-qcom-hwe/recipes-kernel/linux/linux-kernel-qcom_6.6.bb.
```

```
KERNEL_DEFCONFIG = "${S}/arch/arm64/configs/qcom_defconfig"
```

```
KERNEL_CONFIG_FRAGMENTS:append = " ${S}/arch/arm64/configs/qcom_
addons.config"
```

1.3 Understand power architecture

The CPU subsystem architecture describes the organization of CPU cores and their power domains. Understanding the architecture helps you to monitor the CPU core usage and balance the power consumption and performance of the device.

QCS6490/QCS5430

APSS is a set of CPUs organized into clusters. All CPU cores in a cluster operate at a similar frequency. Each CPU core in a cluster has a dedicated L1 and L2 cache.

The CPU subsystem consists of three clusters and L3 cache as follows:

- Qualcomm® Kryo™ CPU Prime cluster, which has a performance CPU core
- Kryo Gold cluster, which has CPU cores with balanced power and performance
- Kryo Silver cluster, which has low power CPU cores for lightweight applications
- Common L3 cache

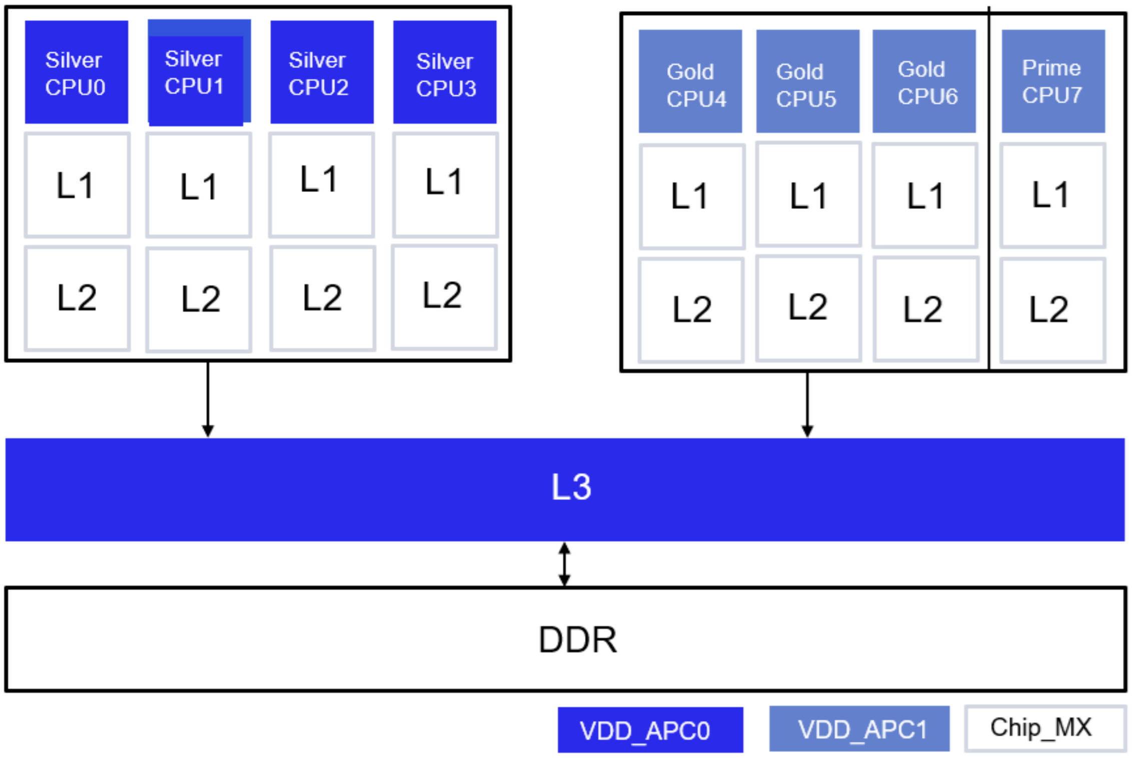

Figure : QCS6490/QCS5430 APSS CPU subsystem architecture

The CPU subsystem operates on two power domains, each supplied by a dedicated switched mode power supply (SMPS) rail.

The following are the supported power domains:

Table : Power domains and their descriptions

Power domains	Description
`VDD_APC0`	Power domain for the Silver cluster CPU cores and L3 cache digital circuit
`VDD_APC1`	Power domain for the Gold and Prime clusters CPU cores
`Chip_MX`	Power domain for the L1, L2, and L3 cache memories

Note: To know more about CPU core arrangement, see QCS6490 Data Sheet and QCS5430 Data Sheet.

QCS9075

The CPU subsystem consists of two clusters with a dedicated L3 cache for each cluster. Each CPU core in a cluster has a dedicated L1 and L2 cache. All CPU cores in a cluster operate at a similar frequency.

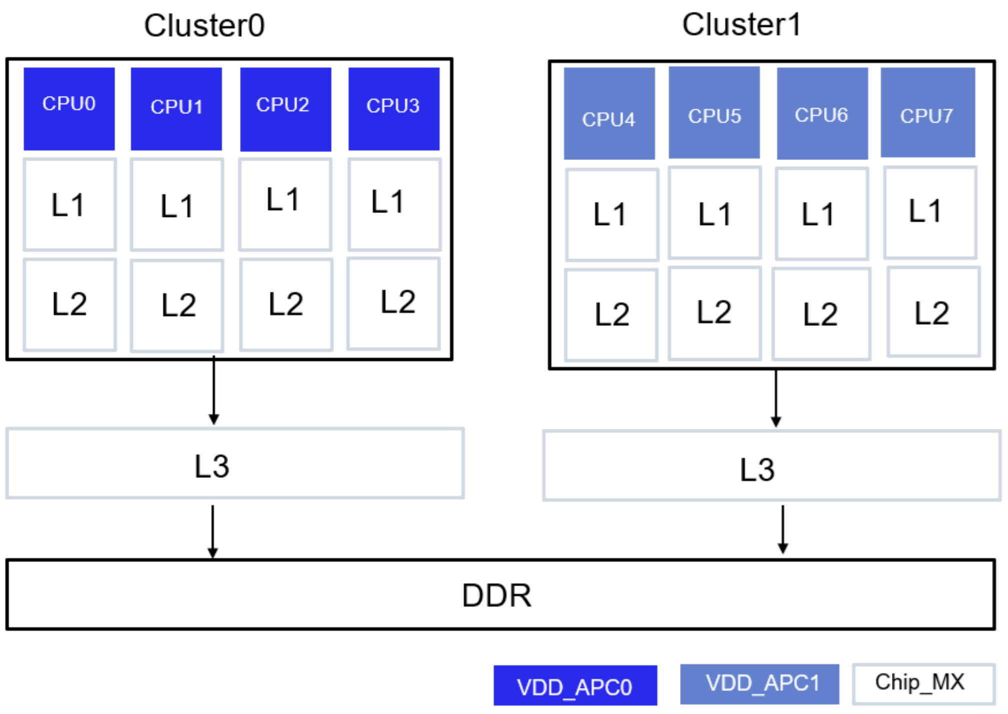

Figure : QCS9075 APSS CPU subsystem architecture

The CPU subsystem operates on two power domains, each supplied by a dedicated SMPS rail.

The following are the supported power domains:

Table : Power domains and their descriptions

Power domains	Description
`VDD_APC0`	Power domain for Cluster0 CPU cores and L3 cache digital circuit
`VDD_APC1`	Power domain for Cluster1 CPU cores and L3 cache digital circuit
`Chip_MX`	Power domain for the L1, L2, and L3 cache memories

To know more about CPU core arrangement, see QCS9075 Data Sheet.

QCS8275

APSS is a set of CPUs organized into clusters. All CPU cores in a cluster operate at a similar frequency. Each CPU core in a cluster has a dedicated L1 and L2 cache.

The CPU subsystem consists of three clusters and L3 cache as follows:

- Kryo Prime cluster with performance CPU cores
- Kryo Gold cluster, which has CPU cores with balanced power and performance
- Kryo Silver cluster, which has low power CPU cores for lightweight applications
- Silver cluster has a dedicated L3 cache
- Gold and Prime clusters share a dedicated L3 cache

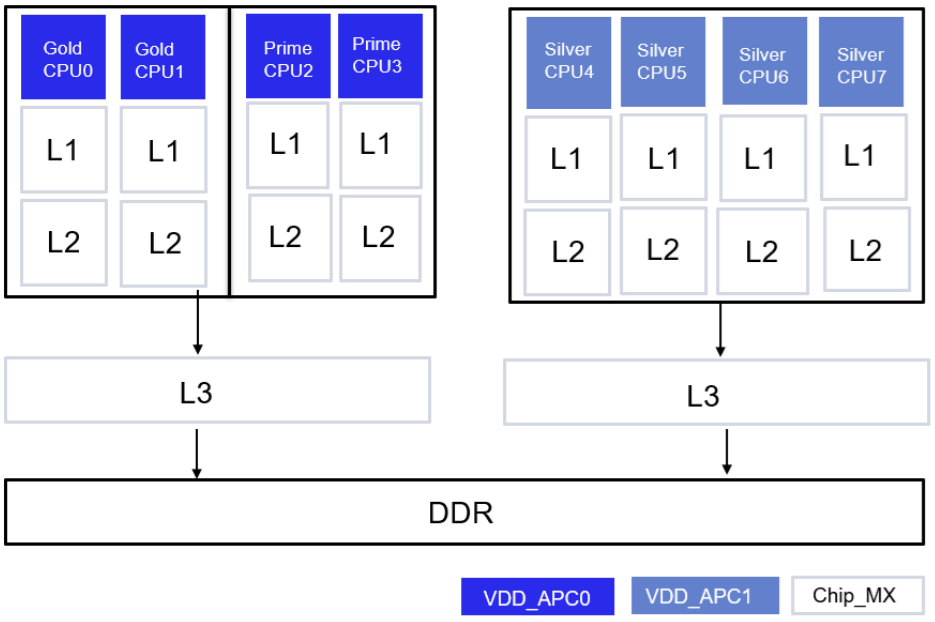

Figure : QCS8275 APSS CPU subsystem architecture

The CPU subsystem operates on two power domains, each supplied by a dedicated SMPS rail.

The following are the supported power domains:

Table : Power domains and their descriptions

Power domains	Description
`VDD_APC0`	Power domain for the Gold and Prime cluster CPU cores and L3 cache digital circuit
`VDD_APC1`	Power domain for the Silver cluster CPU cores and L3 cache digital circuit
`Chip_MX`	Power domain for the L1, L2, and L3 cache memories

To know more about CPU core arrangement, see QCS8275 Data Sheet.

1.4 Power states and features

The power saving states, also referred to as sleep states, enable you to review the power states before, during, and after running an application. The process ensures a comprehensive power sanity check of the device.

Review SoC sleep states

The SoC supports several sleep states based on the requirement of resources, such as, double data rate (DDR), crystal oscillator (XO) clock, digital power supply `(Chip_CX)`, and memory domains power supply `(Chip_MX)`.

Note: The SoC sleep states are enabled on QCS6490/QCS5430.

As the device transitions from active state to SoC sleep state, the power consumption decreases.

For example, when the device is idle with no use cases running, it's in the SoC sleep state (AOSD). This is the deepest sleep state and consumes the least power. In this state, the resources like, XO clock, memory, and power supply of digital and memory domains also enter their respective sleep states.

The following table lists the supported SoC sleep states.

Note: The SoC sleep states listed in the table are for information only. You can't enable or disable SoC sleep states.

Table : SoC sleep states

SoC sleep states	Resource states
Active	• XO clock on • `Chip_CX` (Digital) and `Chip_MX` (Memory) power domains operate at active voltage level
DDR collapse	• Memory in Self-Refresh mode • XO clock on • `Chip_CX` and `Chip_MX` power domains are configured with active voltage
XO shutdown (CXSD)	• Memory in Self-Refresh mode • XO clock off • `Chip_CX` power domain configured with least voltage • `Chip_MX` configured with active voltage
SoC sleep (AOSD)	• The deepest system sleep state where the system is expected to achieve the lowest sleep power, which is the most desired state for power management • Memory in Self-Refresh mode • XO clock off • `Chip_CX` and `Chip_MX` power domains configured with the least voltage

Check SoC sleep state

To retrieve SoC sleep statistics, run the following commands:

```
mount -t debugfs none /sys/kernel/debug
```

```
cat /sys/kernel/debug/qcom_stats/aosd
```

```
cat /sys/kernel/debug/qcom_stats/cxsd
```

The following is a sample output:

```
Count:  3

Last Entered At:  1087943710378

Last Exited At:  1088442890377

Accumulated Duration:  6668562002
```

The following table explains the output fields:

Table : SoC sleep states output fields

Field	Explanation
`Count`	• Indicates the number of times the SoC entered a particular power state • A nonzero count indicates that the SoC has exercised sleep states
`Last Entered At`	Indicates the last sleep entry timestamp in ticks
`Last Exited At`	Indicates the last sleep exit timestamp in ticks
`Accumulated duration`	Total amount of time in sleep, represented in ticks

Note: The XO clock frequency of 19.2 MHz determines the ticks.

CPU power management

CPU power management techniques aim to reduce the power consumption of the CPU during both active and idle periods by using the resources efficiently.

For example, run the cores at the necessary frequency while keeping them in a Low-Power state as much as possible.

You can use various power optimization techniques and CPU core sleep states to achieve optimized CPU power on Qualcomm Linux.

Dynamic voltage and frequency scaling

Dynamic voltage and frequency scaling (DVFS) is a technique used to reduce power consumption of the CPU during active periods. This technique is used to switch the CPU core frequency based on load requirement. DVFS helps in balancing the power, performance, and thermal behavior of the device.

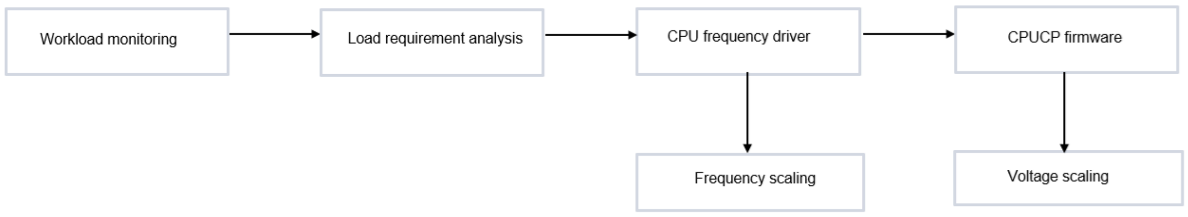

Figure : DVFS workflow

For more information about DVFS, see index: kernel/git/torvalds/linux.git.

For example, if the available CPU frequencies are 1 GHz, 1.5 GHz, and 2 GHz, and the application must run at 1 GHz, then CPU's maximum frequency should be 1 GHz. This mechanism optimizes CPU power by ensuring appropriate use of CPU frequency and voltage, thereby supporting high performance.

Commands	Explanation

Customize CPU frequency tunables

The following table lists the commands used to tune CPU frequency for balanced power and performance:

Table : Commands to tune CPU frequency

Commands	Explanation
`cat /sys/devices/system/cpu/cpufreq/policy<x>/scaling_cur_freq`	Reads the existing CPU frequency
`cat /sys/devices/system/cpu/cpufreq/policy<x>/scaling_available_frequencies`	Reads the supported frequencies
`cat /sys/devices/system/cpu/cpufreq/policy<x>/scaling_min_freq`	Reads the minimum supported CPU frequency
`echo <cpu freq in KHz> > /sys/devices/system/cpu/cpufreq/policy<x>/scaling_min_freq`	• Sets the minimum CPU frequency • Overwrites the existing CPU minimum frequency • Sets any frequency from `scaling_available_frequencies`
`cat /sys/devices/system/cpu/cpufreq/policy<x>/scaling_max_freq`	Reads the maximum supported CPU frequency
`echo <cpu freq in KHz> > /sys/devices/system/cpu/cpufreq/policy<x>/scaling_max_freq`	• Sets the maximum supported CPU frequency • Overwrites existing maximum CPU frequency • Sets any frequency from `scaling_available_frequencies`

QCS6490/QCS5430

Note: `<x>` refers to 0, 4, 7.

- 0 is for the Silver cluster
- 4 is for the Gold cluster
- 7 is for the Prime cluster

- Unit for frequency is KHz

QCS9075

Note: `<x>` refers to 0, 4.

- 0 is for Cluster0
- 4 is for Cluster1
- Unit for frequency is KHz

QCS8275

Note: `<x>` refers to 0, 2, 4.

- 0 is for the Gold cluster
- 2 is for the Prime cluster
- 4 is for the Silver cluster
- Unit for frequency is KHz

Any modifications made through `SSH` commands revert to their default values upon device reset.

Efficient CPU idle power management

CPU idle power management, also known as idle power management, keeps the CPU cores in the appropriate sleep state for efficient use of energy when the device is idle.

CPU sleep states

A CPU core supports several sleep states. Each sleep state has associated entry and exit latencies and different levels of power consumption. The selection of a core's sleep state depends on how soon that core is required again for another execution.

A C-state number represents each CPU sleep state. Two key parameters are associated with each C-state:

- Power saving: The power saving of the CPU increases as you select higher C-states, but it results in increased wake-up latency.

- Wake-up latency: This is the time taken to transition a CPU core from sleep state to running state.

For example, a deeper sleep state (C4) has higher wake-up latency and lower power consumption as shown in the following figure.

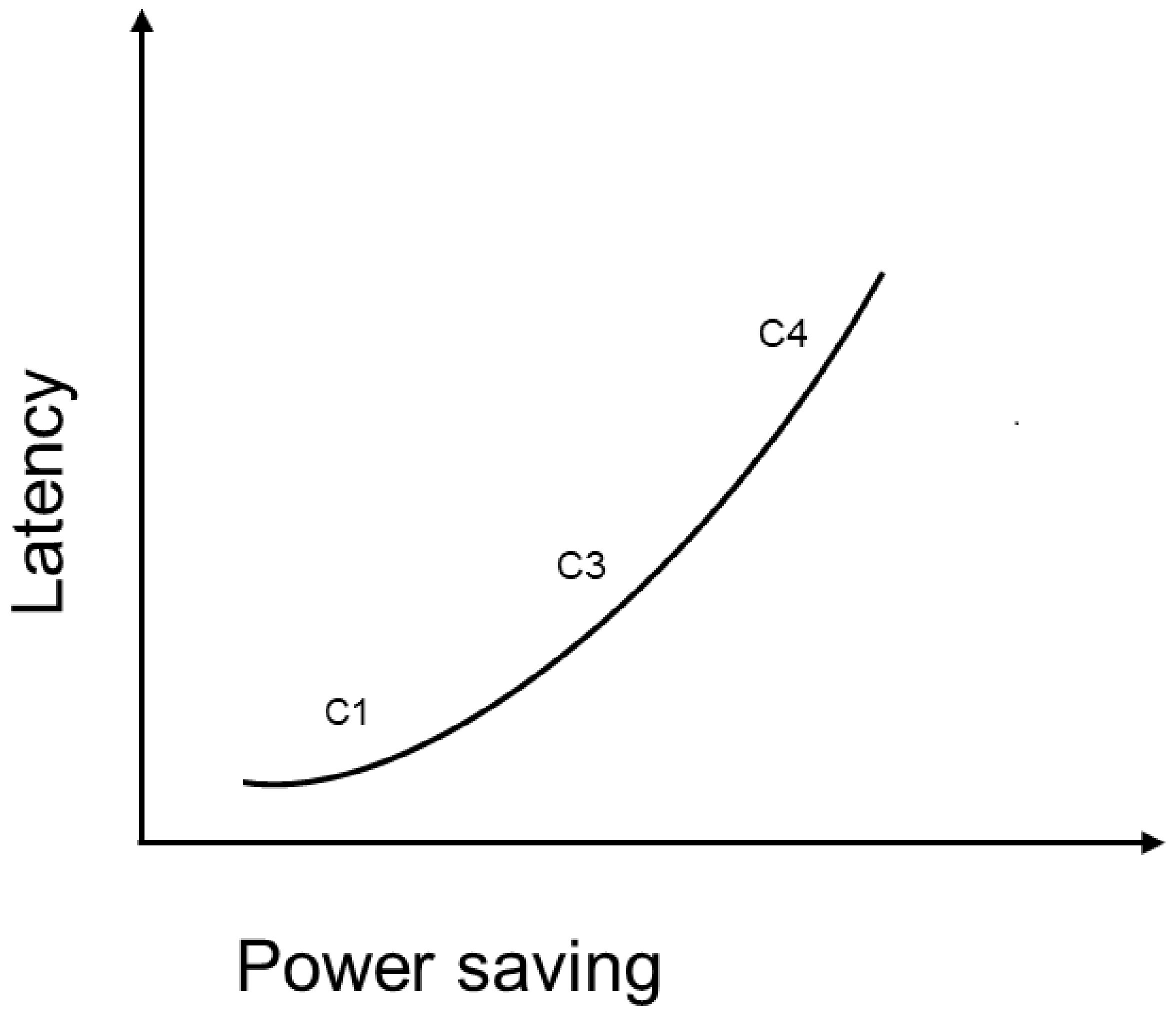

Figure : Tradeoff between power and latency

Use the following command to view the latencies of CPU sleep state:

```
cat /sys/devices/system/cpu/cpu<X>/cpuidle/state<Y>/latency
```

Note: Here, `<X>` represents the core ID, which varies from 0 to 7, and `<Y>` is the index of the Linux power state (0, 1, 2). The following table defines the Linux power states.

You can study and configure sleep states to tune applications for balanced power and performance.

The following table lists the supported CPU sleep states and their associated C-state numbers:

Note: The CPU sleep states are enabled by default.

Table : Supported core power state

CPU sleep state	Description	Linux power state
C1	• CPU core clock domain off • Power domains `VDD_APC0/ VDD_APC1` on • Generic interrupt controller (GIC) clock on	State0 (S0)
C3	• CPU core clock domain off • Core power collapse with L1 and L2 cache flushed • Power domains `VDD_APC0/ VDD_APC1` on	State1 (S1)
C4	• C3 + PLL off for the Silver cluster • C3+ PLL power domains off for the Gold and the Prime clusters	State2 (S2)

Enable/Disable CPU sleep state

The following table lists how to configure CPU sleep states:

Note: If you change the configuration of a sleep state, you should check the power impact on the device.

Table : Commands to enable/disable CPU power states

Linux power state	Enable/Disable	Commands
S0	Enable	`echo 0 > /sys/devices/system/cpu/cpu<X>/cpuidle/state0/disable`
	Disable	`echo 1 > /sys/devices/system/cpu/cpu<X>/cpuidle/state0/disable`
S1	Enable	`echo 0 > /sys/devices/system/cpu/cpu<X>/cpuidle/state1/disable`
	Disable	`echo 1 > /sys/devices/system/cpu/cpu<X>/cpuidle/state1/disable`
S2	Enable	`echo 0 > /sys/devices/system/cpu/cpu<X>/cpuidle/state2/disable`

Linux power state	Enable/Disable	Commands
	Disable	`echo 1 > /sys/devices/system/cpu/cpu<X>/cpuidle/state2/disable`

Note: `<X>` denotes the core ID, which varies from 0 to 7.

Check CPU idle statistics

To check the statistics of CPU core sleep states run the following command:

```
cat /sys/devices/system/cpu/cpu<X>/cpuidle/state<Y>/usage
```

Here, `<X>` is the core ID, which varies from 0 to 7, and `<Y>` is the index of the Linux power state (0, 1, and 2).

The following is an example of the output:

```
cd /sys/devices/system/cpu

grep -r "" cpu*/cpuidle/*/usage

cpu0/cpuidle/state0/usage:97681

cpu0/cpuidle/state1/usage:95

cpu1/cpuidle/state0/usage:180055

cpu1/cpuidle/state1/usage:59
```

Note: The `usage` count represents the number of times `cpu<X>` enters a Linux power state.

Monitor Suspend Power Management

The Suspend state is the deepest sleep state in the APSS with all user space processes frozen and drivers releasing their resources. During the Suspend state, all cores enter the C4 state followed by the AOSD Sleep state.

A user space module can configure the Suspend state using the following command:

```
echo mem > /sys/power/autosleep
```

Check suspend statistics

To check suspend statistics run the following commands:

```
cd /sys/power/suspend_stats
cat success
cat fail
```

The success count indicates the number of times the APSS has successfully entered the Suspend state while the fail count indicates the number of times the APSS has failed to enter the Suspend state.

Efficient wakeup latency with power QoS APIs

The power management quality of service (PM QoS) API allows drivers to specify the latency requirement (in microseconds).

If the wake-up latency of a CPU core sleep state is longer than the specified QoS latency, then the CPU core doesn’t enter the respective sleep state.

For more information about APIs, see index: kernel/git/torvalds/linux.git.

1.5 SoC power reference measurements

This guide provides the reference power measurements of SoC.

See Qualcomm Linux Power and Thermal Guide - Adendum for QCS6490/QCS5430 SoC power measurements.

CPU active power versus performance

Specint2k6 is a performance benchmark test suite for the CPU. The following figure shows the QCS6490 CPU active power measured at various Specint2k6 performance scores:

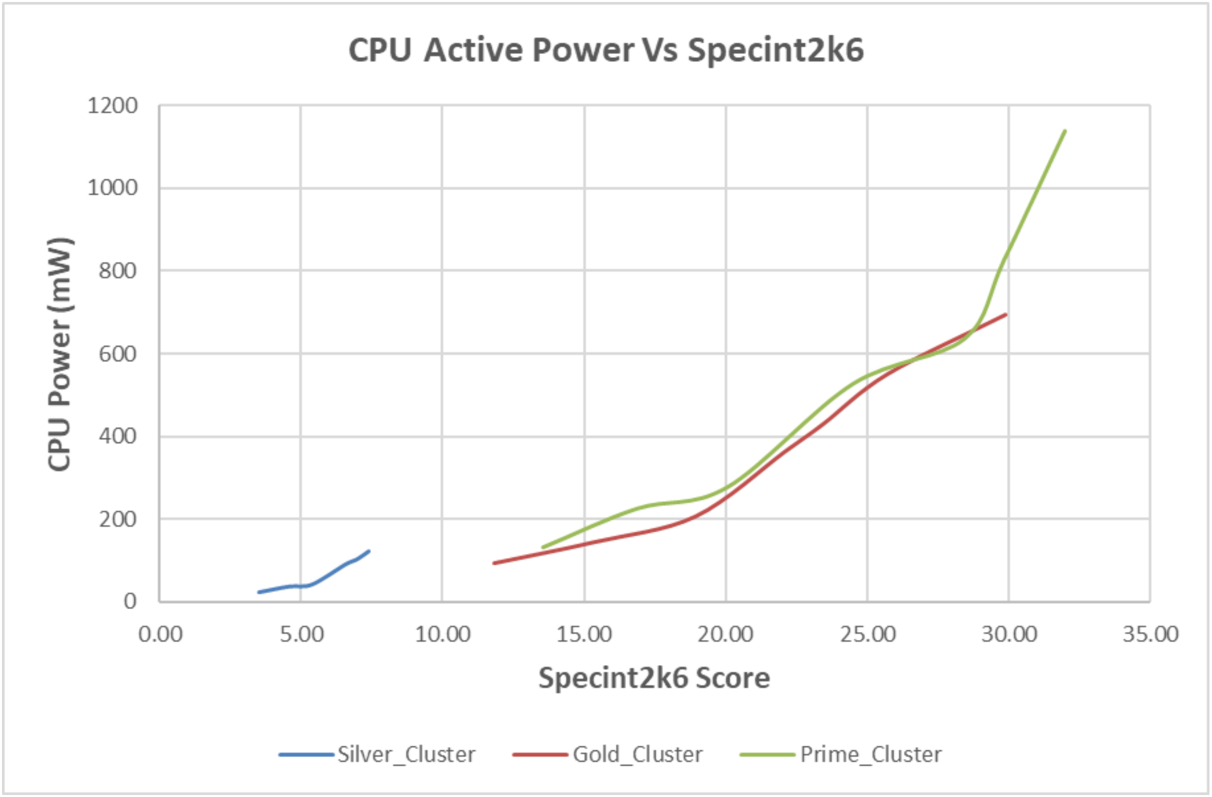

Figure : QCS6490 CPU active power measured at different Specint2k6

Note: Test cases for QCS9075 and QCS8275 will be added in the future release.

1.6 Debug high sleep current

To enhance the power efficiency of the device, it's crucial to optimize the power consumption of both the SoC and the CPU.

Note: The information is applicable for QCS6490/QCS5430.

To know more details, follow the steps:

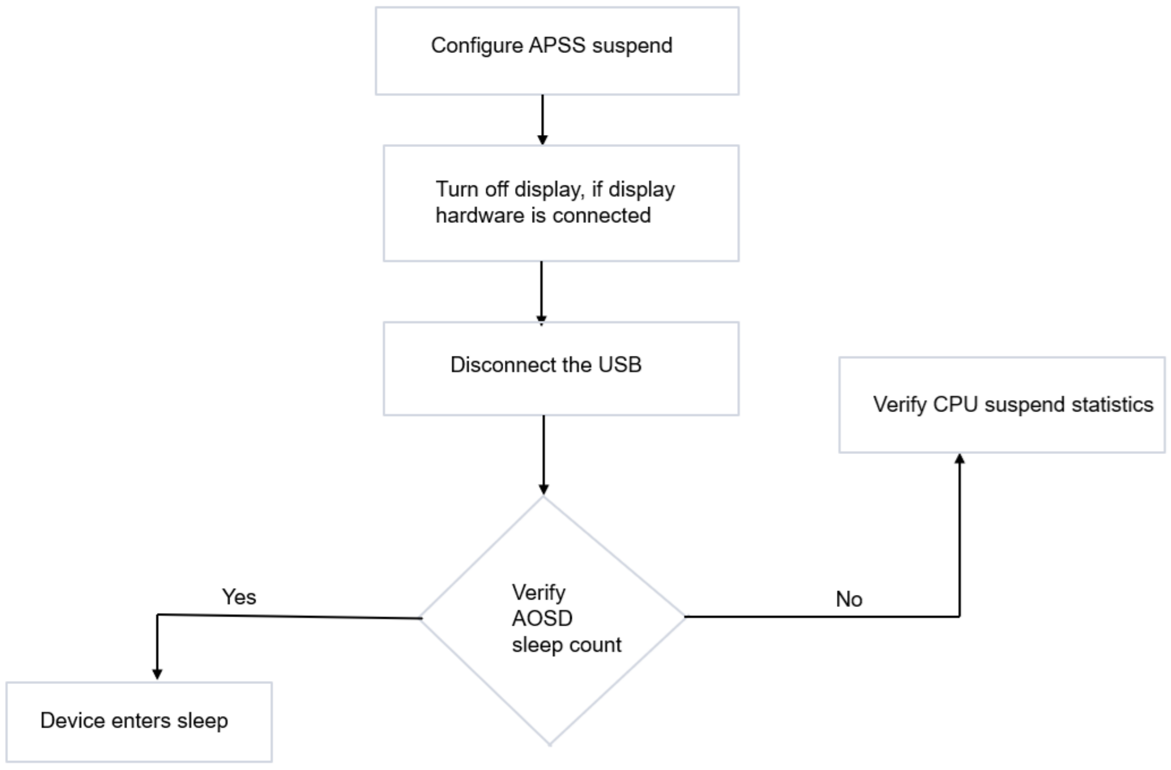

Figure : Debug flowchart

Prerequisites

1. To configure APSS suspend run the following command:

```
echo mem >/sys/power/autosleep
```

2. Turn off the display, if the display hardware isn't connected.

3. Disconnect the USB.

Procedure

To optimize high power consumption, follow the steps:

1. AOSD sleep is the lowest power state. Verify the AOSD sleep count using the commands provided in Check SoC sleep state.

2. If the AOSD count value is zero or static, verify the success count of CPU suspend using the instructions provided in Check suspend statistics.

3. If the CPU suspend count value is zero or static, check if any active wake lock is preventing the CPU from sleep.

2 Thermal management

Thermal management is important to keep the junction and surface temperature of the device under control. The device operates most efficiently when the temperature is within the specified threshold.

2.1 Overview of thermal management

Thermal management is a configurable framework that allows you to control the temperature of the entire device, ensuring reliability of the device. It aids in maintaining the junction and surface temperatures of the device within specified limits even when running high workload applications.

This guide provides information about:

- Thermal architecture
- Heat dissipation techniques supported on Qualcomm Linux
- Tuning of thermal thresholds for mitigation based on the thermal requirements of the product
- Sending a notification to the user space
- Features to adjust temperature configurations to achieve the required thermal behavior

Note: See Hardware SoCs that are supported on Qualcomm Linux.

2.2 Get started with thermal management

This section describes how to develop software on Qualcomm Linux. Ensure to set up the infrastructure before building your software. For more information on setting up infrastructure, see Qualcomm Linux Build Guide. This guide also provides information about the common build workflow.

Prepare performance build

Preparing the performance build ensures all thermal features are optimized and functioning correctly for effective thermal management. Enable SSH in Permissive mode to prepare the performance build. For instructions, see Sign in using SSH.

The default Qualcomm build is the performance build. For evaluation of any thermal behavior, use the performance build.

The performance build uses the following kernel configurations:

```
kernel_platform/kernel/arch/arm64/configs/qcom_defconfig
```

```
kernel_platform/kernel /arch/arm64/configs/qcom_addons.config
```

The kernel configurations are defined in the source code kernel recipe at:

```
layer/meta-qcom-hwe/recipes-kernel/linux/linux-kernel-qcom_6.6.bb.
```

```
KERNEL_DEFCONFIG = "${S}/arch/arm64/configs/qcom_defconfig"
```

```
KERNEL_CONFIG_FRAGMENTS:append = " ${S}/arch/arm64/configs/qcom_
addons.config"
```

2.3 Understand thermal architecture

The thermal architecture shows how the thermal framework of Qualcomm Linux interacts with temperature sensors (Tsens) hardware, cooling map interfaces, and user space clients.

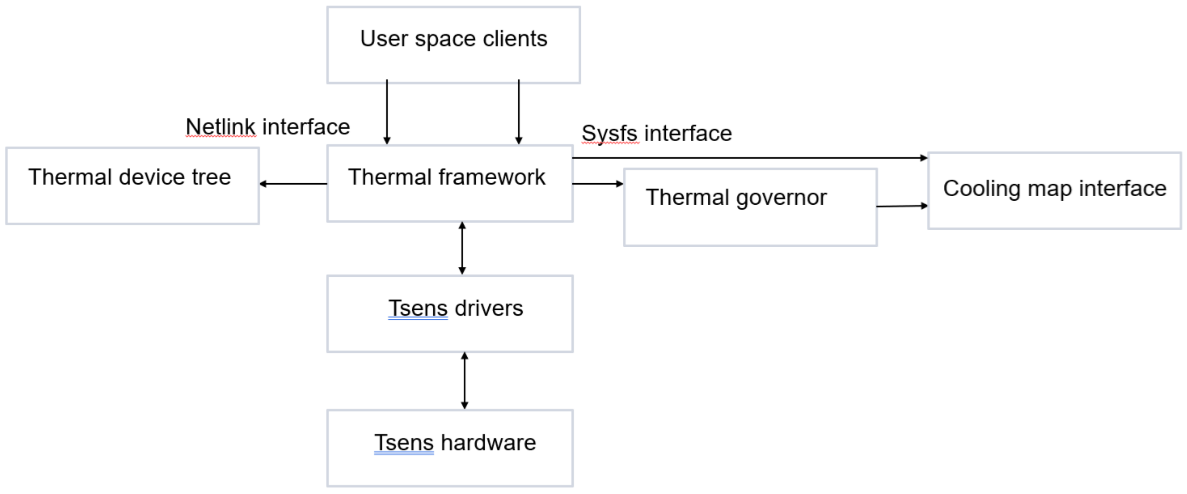

Figure : Thermal architecture

Table : Description of the components of thermal architecture

Components of thermal architecture	Description
Thermal device tree	• A thermal device tree is made up of various thermal zones. • A thermal zone is configured for every Tsens. • Each thermal zone consists of sensor hardware information, different levels of thermal thresholds, and their respective cooling actions. • All rules for a thermal zone are defined in the device tree.
Thermal framework	The thermal framework parses the thermal zone rules from the device tree and configures Tsens hardware to generate interrupts when temperature crosses the threshold.
Thermal governor	• The thermal governor is a temperature monitor algorithm that controls the temperature of a thermal zone. • It mitigates the cooling devices associated with the zone and keeps the temperature within the limit. • Upon receipt of interrupt notification from Tsens hardware, the thermal framework uses a stepwise thermal governor to communicate the cooling actions to the cooling map interface (CPU frequency driver).
Cooling map interface	• A cooling map interface is a collection of devices that can be throttled to reduce the temperature of Qualcomm Linux. • Every trip instance of the thermal zone is associated with a cooling device. • The thermal core framework aggregates all the cooling device instance requests and places the aggregated requests to the cooling device.

Components of thermal architecture	**Description**
User space clients	User space clients use the following interfaces to communicate with the thermal framework: • The `sysfs` interface is used to read the temperature of Tsens and kernel trip information. • The `Netlink` interface gets information of Tsens temperature notifications.

Thermal mitigation policies

Qualcomm Linux uses in-built software and hardware mitigation policies to regulate the device's thermal behavior.

The thermal framework applies the mitigation policies when required and operates in the following ways:

- Uses on-die Tsens to monitor the thermal response of the system
- Applies mitigation based on thermal threshold limits according to the device tree
- Provides software throttling, hardware throttling, software shutdown, and hardware shutdown mitigation policies to control temperature

Table : Temperature management techniques and description

Temperature management techniques	Description
Software throttling	• The Tsens triggers an interrupt whenever the temperature in the system crosses the configured threshold. • The thermal framework initiates the following mitigation actions on the CPU, GPU, and NSP in response to the interrupts: – Thermal DCVS ○ Reduces the maximum operating frequency if temperature thresholds are crossed ○ Voltage scaling occurs according to clock scaling, which reduces power consumption and temperature in turn – CPU idle injection The core is put in the deepest Low-Power state for the configured idle time.
Hardware throttling	• In-built hardware executes thermal mitigation actions ensuring system reliability • Reduces CPU and NSP clock speed to quickly recover from high thermal conditions
Software shutdown	The thermal framework initiates graceful shutdown upon receiving interruption from Tsens.
Hardware shutdown	The hardware initiates a device shutdown.

Note: The thermal mitigation techniques are for information purpose only. They help to understand performance regressions during high thermal conditions.

The following tables show the temperatures at which in-built software and hardware mitigation occur.

QCS6490/QCS5430

Table : Temperature for hardware and software throttling for QCS6490/QCS5430

Mitigation actions	CPU	NSP	GPU
Software throttling	115 ºC Mitigation action: Core idle injection	110 ºC Mitigation action: Frequency throttling	105 ºC Mitigation action: Frequency throttling
Hardware throttling	105 ºC Mitigation action: Frequency throttling	102 ºC Mitigation action: Frequency throttling	Not applicable
Software shutdown	118 ºC Mitigation action: Device shutdown		
Hardware shutdown	120 ºC Mitigation action: Device shutdown		

QCS9075

Table : Temperature for hardware and software throttling for QCS9075

Mitigation actions	CPU	NSP	GPU
Software throttling	116 ºC Mitigation action: Core idle injection	105 ºC Mitigation action: Frequency throttling	105 ºC Mitigation action: Frequency throttling
Hardware throttling	110 ºC Mitigation action: Frequency throttling	110 ºC Mitigation action: Frequency throttling	Not applicable
Software shutdown	118 ºC Mitigation action: Device shutdown		
Hardware shutdown	120 ºC Mitigation action: Device shutdown		

QCS8275

Table : Temperature for hardware and software throttling for QCS8275

Mitigation actions	CPU	NSP	GPU
Software throttling	116 ºC Mitigation action: Core idle injection	105 ºC Mitigation action: Frequency throttling	105 ºC Mitigation action: Frequency throttling
Hardware throttling	110 ºC Mitigation action: Frequency throttling	110 ºC Mitigation action: Frequency throttling	Not applicable
Software shutdown	118 ºC Mitigation action: Device shutdown		
Hardware shutdown	120 ºC Mitigation action: Device shutdown		

Note: To mitigate thermal conditions, you can create your own thermal zone rules based on a PCB thermistor. For more information, see Configure a new thermal zone.

2.4 Learn about thermal interfaces

The thermal architecture includes the `sysfs` interface, which reads the thermal zone configurations and applies cooling actions to the CPU. The user space clients use the `sysfs` interface to communicate with the cooling map interface.

For more information about `sysfs` interface, see index: kernel/git/torvalds/linux.git.

Users can interact with the device through the following interfaces:

Customize a thermal zone

The thermal zone connects Tsens hardware information, temperature thresholds, and mitigation actions. The thermal framework creates node IDs for each thermal zone rule defined in the device tree.

The following table lists the `sysfs` commands used to read and customize the thermal zone configurations:

Note: `<x>` in the table represents thermal zone ID.

Table : Description of thermal zone commands

Commands	Description
`ls /sys/class/thermal/thermal_zone*`	Lists all thermal zones and their IDs that the device contains
`cat /sys/class/thermal/thermal_zone<x>/type`	Provides the names of thermal zones that the device contains
`echo enabled > /sys/class/thermal/thermal_zone<x>/mode`	Enables a thermal zone **Note:** All thermal zones are enabled by default.
`echo disabled > /sys/class/thermal/thermal_zone<x>/mode`	Disables a thermal zone

`echo <trip_temp> >/sys/class/thermal/thermal_zone<x>/trip_point_<A>_temp`
Here,
`trip_temp` is the temperature threshold in m℃. `A` denotes trip point numerical. For example, you can modify the temperature threshold for a trip point 0 using the following command:
`echo 95000 > /sys/class/thermal/thermal_zone20/trip_point_0_temp`
	• Modifies threshold temperature for trip point A
	• Verify the number of trip points available to the thermal zone in the thermal zone directory:
	`/sys/class/thermal/thermal_zone<x>`
	and run the command

Commands	**Description**

`echo <trip_hyst> > /sys/class/thermal/thermal_zone<x>/trip_point_<A>_hyst`
Here,
`trip_hyst` is hysteresis value in m℃. `A` denotes trip point numerical. For example, you can modify hysteresis for trip point 0, which means a clear threshold is triggered at trip_temp-1000 m℃ using the following command:
`echo 1000 > /sys/class/thermal/thermal_zone20/trip_point_0_hyst`
	• Modifies hysteresis value for trip point A
	• Verify the number of trip points available to the thermal zone in the thermal zone directory:
	`/sys/class/thermal/thermal_zone<x>`
	and run the command

Note: Modifications to temperature threshold configurations should be for test purpose only. Don't change any thermal zone configurations of Tsens. Resetting the device changes modifications performed through sysfs commands to default values.

You can add a product specific thermal zone and its rules in the device tree to control the temperature of the device according to the thermal requirement as described in: Configure a new thermal zone.

Control heat dissipation with cooling devices

The performance of some of the components can be adjusted to reduce heat dissipation. These components are considered as cooling devices. The CPU cores are cooling devices as their performance can be lowered to reduce heat dissipation.

For more information about cooling devices, see index: kernel/git/torvalds/linux.git.

The following table lists the `sysfs` command that describes how to read information about cooling devices.

Note: `<x>` in the table represents cooling device ID.

Table : Description of commands for cooling devices

Commands	Description
`ls /sys/class/thermal/cooling_device*`	Lists all cooling devices and their IDs that the device contains
`cat /sys/class/thermal/cooling_device<x>/type`	Provides the name of the cooling device
`cat /sys/class/thermal/cooling_device<x>/cur_state`	Reads the existing mitigation state of the cooling device
`cat /sys/class/thermal/cooling_device<x>/max_state`	Reads the maximum supported mitigation state of the cooling device

QCS6490/QCS5430

The following table lists the CPU cooling devices that the user space module of QCS6490/QCS5430 uses for CPU frequency mitigation:

Table : Available CPU cooling devices to user space for QCS6490/QCS5430

Cooling devices	Description
`cpufreq-cpu<x>`	Cooling devices that are used to limit the maximum operating frequency of the CPU to control CPU temperature. • `X` – cluster CPU number • `cpufreq-cpu0` – Silver cluster • `cpufreq-cpu4` – Gold cluster • `cpufreq-cpu7` – Prime cluster

QCS9075

The following table lists the CPU cooling devices that the user space module of QCS9075 uses for CPU frequency mitigation:

Table : Available CPU cooling devices to user space for QCS9075

Cooling devices	Description
`cpufreq-cpu<x>`	Cooling devices that are used to limit the maximum operating frequency of the CPU to control CPU temperature. • `X` – cluster CPU number • `cpufreq-cpu0` – Cluster0 • `cpufreq-cpu4` – Cluster1

Note: The list of cooling devices for QCS8275 will be updated in a future release.

Receive sensor temperature notifications

The thermal core framework uses the `Netlink` interface to provide sensor temperature notifications to the user space.

The user space applications use the `Netlink` library API to get notifications of the sensor temperature and take required action. `Netlink` library APIs are available at: index: kernel/git/torvalds/linux.git.

2.5 Configure a new thermal zone

For optimal performance of your product or application, it's important to customize and control its thermal behavior. You can configure a new thermal zone for a newly added external thermistor to control the surface temperature of Qualcomm Linux.

The following sample code shows how to configure a thermal zone using a device tree.

For example, monitor a temperature threshold of 50 °C and apply mitigation to the CPU cooling devices.

Note: By default, the system configures a stepwise thermal governor.

Parameters that are specific to the thermal zone are explained in index: kernel/git/torvalds/linux.git.

```
thermal-zones {
    <thermal zone name> {
        polling-delay-passive = <2000>;
        polling-delay = <0>;
        thermal-sensors = < thermal sensor phandle and sensor
specifier used to monitor this thermal zone>;
        trips {
            trip0: apc-trip {
                temperature = <50000>;   /*Threshold to trigger
thermal mitigation */
                hysteresis = <2000>;     /*The offset of clear
threshold, the clear threshold is 48000 */
            };
        };
        cooling-maps {
            map0 {
                 trip = <&apc_trip>;
                 cooling-device = <&cpufreq-cpu4 THERMAL_NO_LIMIT 5>;
/* Reduce Gold cluster CPU fmax freq by 5 levels */
            };
        };
    };
};
```

2.6 Debug using ftrace method

Thermal logging helps to understand the active configurations of the thermal zones and the existing thermal mitigation levels. To retrieve thermal configuration and active mitigation levels, use ftrace logs.

ftrace method

To enable thermal ftace logs run the following commands:

```
echo 1 > /sys/kernel/tracing/events/thermal/enable
```

QCS6490/QCS5430

The following are the log snippets of different thermal events:

```
/* In the following snippet, "cpu0-thermal" represents thermal zone
and “temp” indicates existing temperature of thermal zone */
sh-1420    [001] ..... 1132975.934216: thermal_temperature: thermal_
zone=cpu0-thermal id=13 temp_prev=47700 temp=110000
/*In the following snippet, “trip” indicates trip level and trip
level 1 is triggered for cpu0-thermal thermal zone*/
sh-1420    [001] ..... 1132975.934449: thermal_zone_trip: thermal_
zone=cpu0-thermal id=13 trip=1 trip_type=PASSIVE
/*In the following snippet, “type” represents the cooling device and
"target" represents active mitigation of the cooling device*/
sh-1420 [001] ..... 1132975.934473: cdev_update: type=idle-cpu0
target=100
```

QCS9075

The following are the log snippets of mitigation trigger and clear thermal events for QCS9075:

```
/* In the following snippet, " cpu-1-2-1-thermal" represents thermal
zone
and “trip” indicates trip level and trip level 1 is triggered for
cpu-1-2-1-thermal thermal zone*/
irq/222-c252000-115     [000] ..... 250901.340001:
thermal_zone_trip: thermal_zone=cpu-1-2-1-thermal
id=16 trip=1 trip_type=PASSIVE
/*In the following snippet, “type” represents the cooling device and
"target" represents active mitigation of the cooling device*/
irq/222-c252000-115     [000] ..... 250901.340009: cdev_update:
type=idle-cpu6 target=100 kworker/u17:0-23271   [000] ..... 250901.
355101:
thermal_temperature: thermal_zone=cpu-1-2-1-thermal id=16 temp_
prev=30100 temp=34100
kworker/u17:0-23271   [000] ..... 250901.339084: thermal_temperature:
thermal_zone=cpu-1-2-1-thermal id=16 temp_prev=30100 temp=29800
kworker/u17:0-23271   [000] ..... 250901.339139: cdev_update:
type=idle-cpu6 target=0
```

Note: The log snippet for QCS8275 will be updated in a future release.

3 References

Table : Related resources

Resources
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/cpu-freq?h=v6.6
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/power/pm_qos_interface.rst?h=v6.6
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/devicetree/bindings/thermal/thermal-cooling-devices.yaml?h=v6.6
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/devicetree/bindings/thermal/thermal-zones.yaml?h=v6.6
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/tools/thermal/thermal-engine/thermal-engine.c?h=v6.6
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/tools/lib/thermal?h=v6.6

Table : Acronyms and terms

Acronyms or terms	Definition
APSS	Application processor subsystem
CPUSS	CPU subsystem
DDR	Double data rate
DVFS	Dynamic voltage and frequency scaling
GIC	Generic interrupt controller
PM QoS	Power management quality of service
SMPS	Switched mode power supply
Tsens	Temperature sensors
XO	Crystal oscillator