# Thermal Core Overview



80-P9301-113 Rev. C

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

# Revision History

| Revision | Date | Description |
|---|---|---|
| A | September 2017 | Initial release |
| B | July 2018 | ▪ Updated the document title<br>▪ Added Slide 40 *Debugging Thermal Core (Sample Ftrace)* |
| C | November 2018 | Numerous changes were made to this document; it should be read in its entirety |

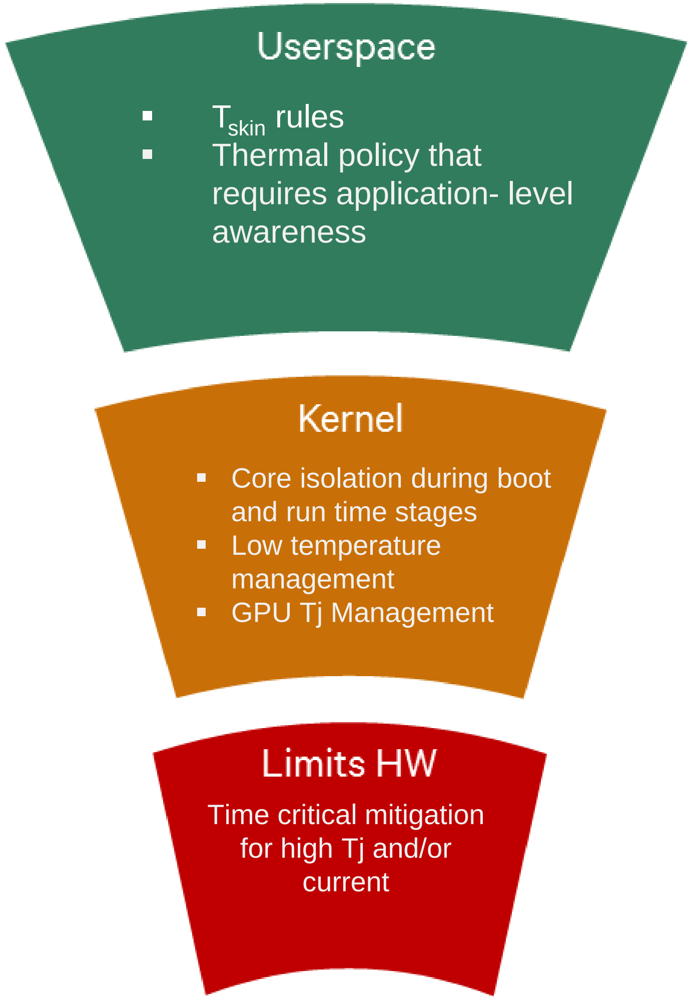
# Contents


- Introduction
- Thermal Software Architecture Overview
- Cooling Devices
- Thermal Zones
- Thermal Core Governors
- Virtual Sensors
- Debugging
- References
- Questions?

# Introduction

# Thermal Core Framework

- Thermal core is the replacement for the legacy framework kernel thermal monitor (KTM)
- Key functions of thermal core:
  - Core isolation (hotplugging)
  - GPU Tj management
  - Low temperature management, that is, VDD restriction
  - OEM $T_{skin}$ management (optional)
- Thermal core is part of the upstream Linux solution for thermal management
- Motivation for moving to the thermal core framework:
  - KTM’s legacy mitigation functions have either been moved to hardware or are no longer required for chipsets going forward
  - KTM emergency frequency mitigation, is now handled by limits management hardening (LMH) hardware
  - QTI’s initiative to make limits management solution available using exclusively up streamed code



# Thermal Core Fundamental Concepts

- Thermal zone: Provides a means to expose thermal sensors through the sysfs as well as define thermal configurations based off of the sensor.
- Cooling device: Any device that can be throttled to reduce temperature.

  Example: CPU, GPU, battery, backlight, and modem
- Thermal governor: A temperature monitor algorithm that controls the temperature of a thermal zone within a limit by mitigating the cooling devices associated with the zone.



# Thermal Core vs. Thermal Engine

- Thermal engine and thermal core provide redundant functionality in terms of thermal management for skin temperature.
- Thermal core is ideal for $T_{skin}$ management for devices that do not have user space available (thermal engine requires user space).
- Thermal core is available very early in boot, whereas thermal engine is available later in boot after the user space is up and running.
- The step_wise thermal governor in thermal core has greater flexibility than the SS algorithm in thermal engine. For example, you can define a thermal zone with multiple temperature ranges and make it behave dynamically for one or more of the ranges and like a monitor instance for one or more of the ranges (for example, hybrid of the SS and monitor algorithms).
- Thermal core requires recompiling for adding new thermal rules, whereas with thermal engine a new rule can be added by updating config and pushing to the file system.
- Thermal engine can still be used the same as previous chipsets.

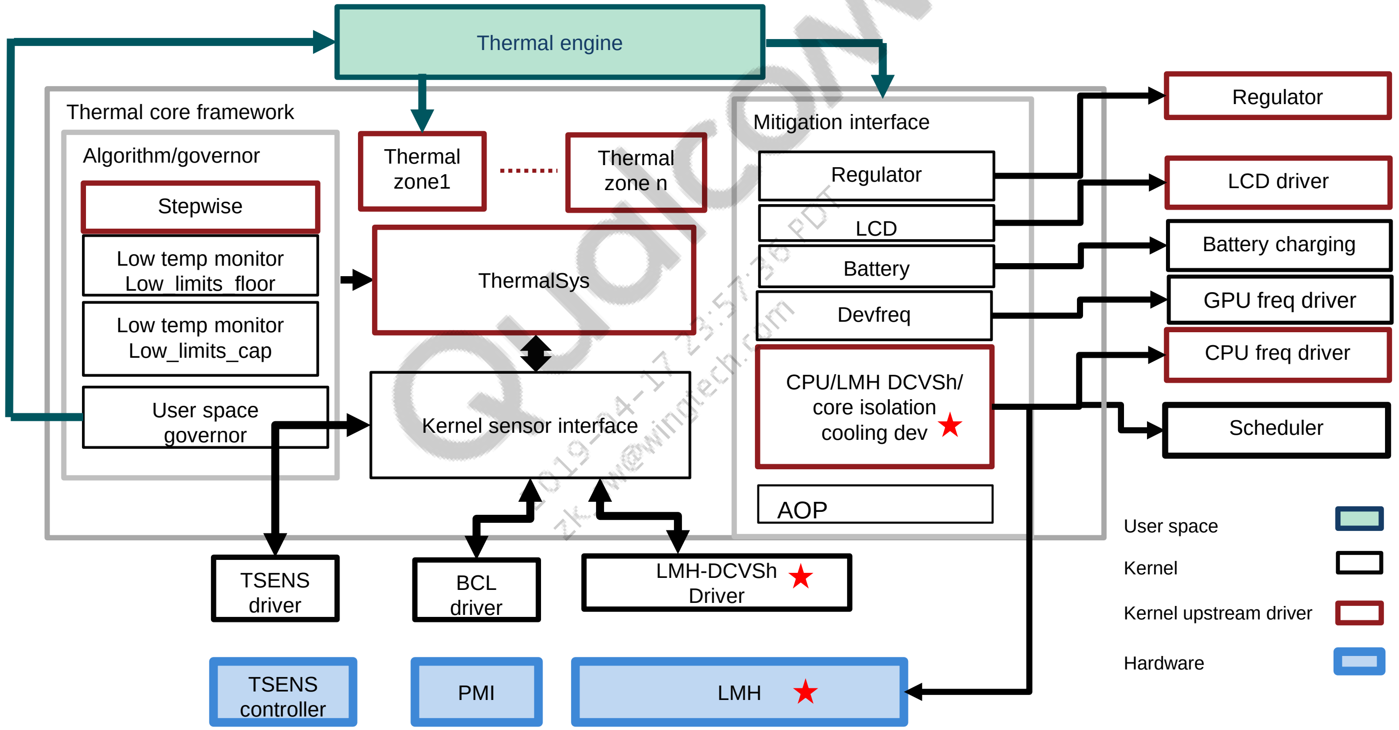
# Thermal Software Architecture Overview

# Thermal Architecture Changes

- KTM is completely removed and replaced by thermal core
- New thermal core governors have been added:
  - User space governor – Added to notify thermal engine when sensor trips (if device tree entry lists “user_space” as governor)
  - Two different low temperature monitor governors added – One to handle VDD restriction and one to handle low SOC and VBat conditions
  - Step_wise governor improved and extended – Bug fixes and hysteresis parameter
- Mitigation actions are aggregated using cooling devices instead of device drivers
- Cooling devices have been extended to have mitigation floors
- Of-thermal interface has been extended
  - Parses device tree containing sensor definitions and thermal policy configuration
  - Provides helper APIs for the sensor driver to aggregate the threshold across multiple thermal zones and send the aggregated threshold to the sensor(s) driver
  - Notifies multiple thermal zones about a threshold violation
  - Upstream version only supported step_wise governor; extended to support newly added governors as well

# Thermal Core Framework Architecture

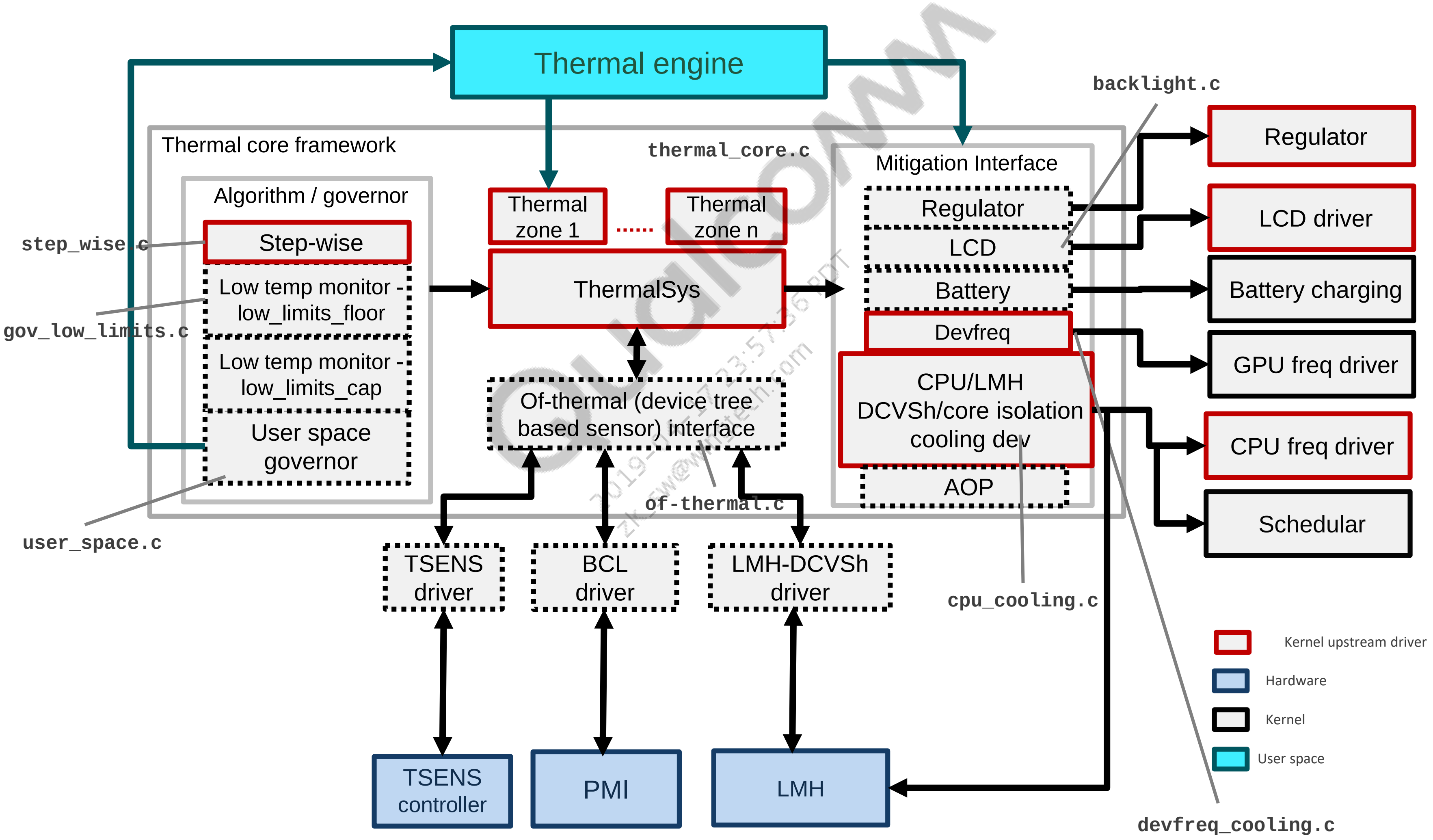
- The thermal core framework is represented as follows within the context of QTI's limits management solution:




★ Ignore LMH for chipsets with no LMH hardware

- Key changes:
  - KTM removed
  - Added and extended thermal core governors
  - Cooling devices added, mitigation actions aggregated using cooling devices
  - Of-thermal interface parses device tree as well as aggregates thresholds and sends to sensor drivers

# Key Thermal Core Source Code Files



# Cooling Devices

# What is a Cooling Device and Which Devices are Available?

- Cooling device is any device that can be throttled to reduce temperature. Same concept as mitigation device with thermal engine
- Cooling devices are visible from “sysfs”
- /sys/class/thermal/cooling_deviceX
- Available cooling devices are:

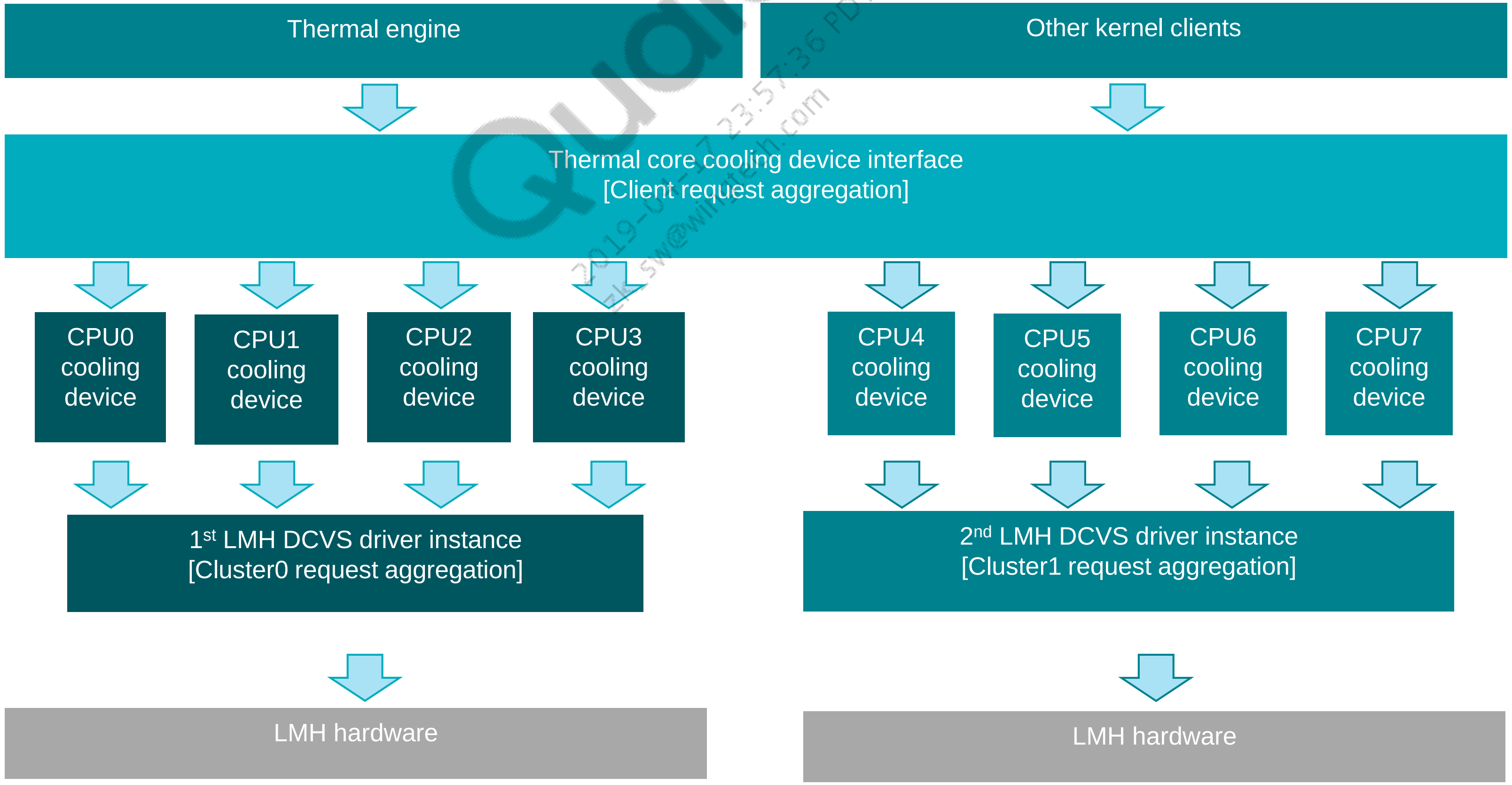
| Cooling device | Action |
|---|---|
| CPU | CPU frequency throttling, core isolation is last mitigation level |
| GPU (devfreq) | GPU frequency throttling |
| Battery | Charge rate throttling |
| Regulator | Increase of voltage on CPU rail |
| Backlight | Display backlight throttling |
| Modem | Adjustment of peak data rates, maximum Tx power |
| AOP | Voltage restriction on CX/EBI with RPMh architecture |

# How does Thermal Core handle Cooling Devices?

- One cooling device instance is associated with a thermal zone’s trip instance, for example, two trip points in a thermal zone for the same cooling device have two different cooling device instances.
- Each cooling device instance places its own vote independent of other instances.
- User space can request for a cap and floor mitigation and this is treated as a separate client.
- When there is a new request, thermal core framework aggregates all the cooling device instance requests and places the aggregated requests to cooling device.
- Thermal core framework does two different aggregations based on the governor type.
  - For governors limiting the cap, it takes maximum of all requests; and for governors limiting the floor, it takes minimum of all the requests.
  - Both the cap and floor request are then communicated to the cooling device driver.

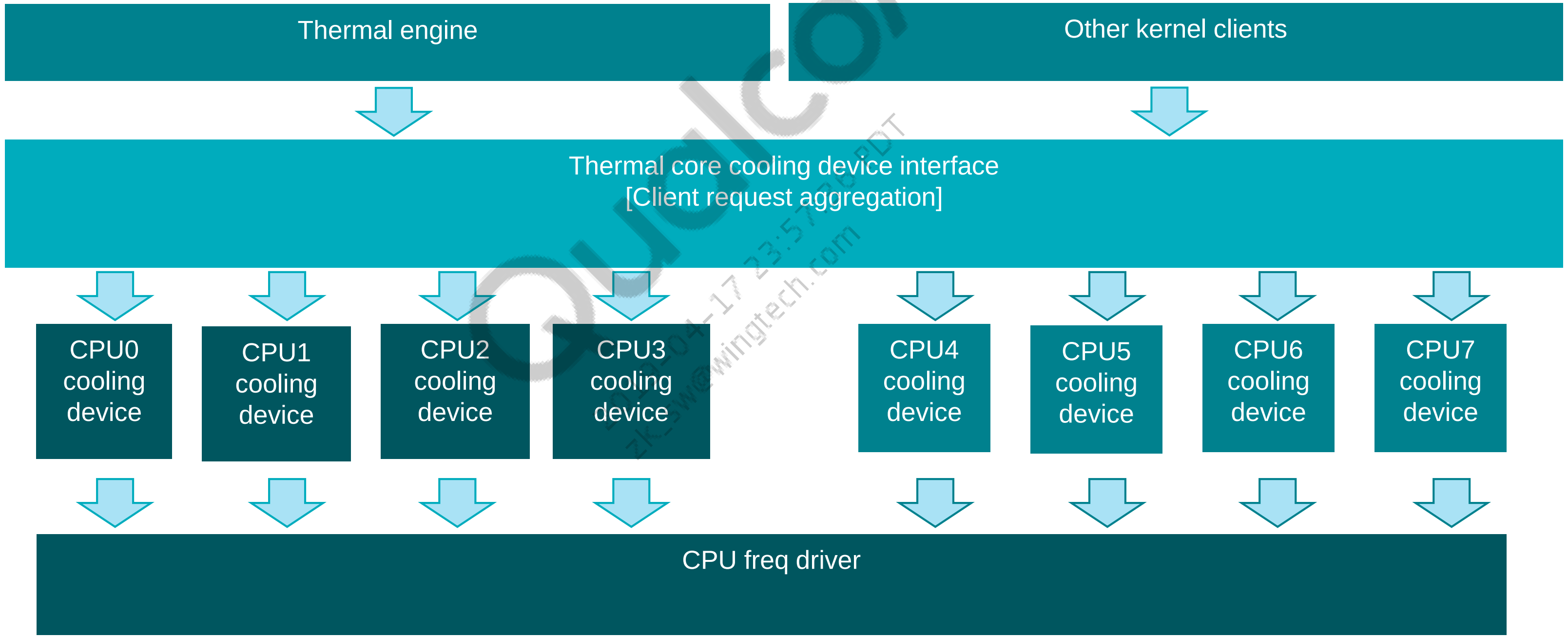
# CPU Mitigation Aggregation – Chipsets with LMH Hardware

- The thermal core interface aggregates the mitigation requests from the clients for each of the CPU cooling devices.
- There are two LMH DCVS driver instances, one per cluster, which aggregates the CPU mitigation requests across the CPU cooling devices that correspond to its cluster.



# CPU Mitigation Aggregation – Chipsets with no LMH Hardware

The thermal core interface aggregates the mitigation requests from all clients like thermal-engine, BCL, and so on for each of the CPU cooling devices, and sends the request to the CPU freq driver.



# Thermal Zones

# What is a Thermal Zone?

- Prior to kernel-4.x, thermal zones were primarily used for exposing thermal sensors through the sysfs
  Example:
  cat/sys/class/thermal/thermal_zoneX/temp
- In kernel-4.x, thermal zones serve two purposes for the thermal core framework:
  - Exposing thermal sensor (same as before)
  - Thermal sensor configuration (that is, adding thermal rules associated with sensor)

- Thermal sensor sdm845.dtsi example:

```
cpu1-gold-usr {
   polling-delay-passive = <0>;
   polling-delay = <0>;
   thermal-sensors = <&tsens0 8>;
   thermal-governor = "user_space";
   trips {
      active-config0 {
         temperature = <125000>;
         hysteresis = <1000>;
         type = "passive";
      };
   };
};
```

- Combination of sensor and rule sdm845.dtsi example:

```
pop-mem-step {
   polling-delay-passive = <10>;
   polling-delay = <0>;
   thermal-sensors = <&tsens1 2>;
   thermal-governor = "step_wise";
   trips {
      pop_trip: pop-trip {
         temperature = <95000>;
         hysteresis = <0>;
         type = "passive";
      };
   };
   cooling-maps {
      pop_cdev4 {
          trip = <&pop_trip>;
           cooling-device =<&CPU4 THERMAL_NO_LIMIT
                                 (THERMAL_MAX_LIMIT-1)>;
      };
...
```

# Thermal Zones – dtsi Files

- Thermal zones are configured in multiple dtsi files. For example, in SDM platform, it has sdm845.dtsi, sdm845-mtp.dtsi, pmi8998.dtsi.
- The following table shows which dtsi file to use for a given type of thermal zone entry.

| dtsi file | Type of entry |
|---|---|
| sdm845.dtsi (SoC dtsi file) | TSENS sensor exposure |
| | GPU Tj rules |
| | POP-mem Tj rule |
| | VDD restriction rules |
| | OEM $T_{skin}$ rules which use TSENS sensors (alternative to using thermal engine) |
| sdm845-mtp.dtsi (or OEM's platform dtsi) | PCB thermistor exposure |
| | OEM $T_{skin}$ rules which use TSENS sensors (alternative to using thermal engine) |
| pmi8998.dtsi | BCL sensors |
| | BCL rules |
| | PMIC alarm |
| | BMS sensors |

# Thermal Zones – sdm845.dtsi

| Type of entry | Entry name |
|---|---|
| TSENS sensor exposure | cpu0-silver-usr |
| | cpu1-silver-usr |
| | cpu2-silver-usr |
| | cpu3-silver-usr |
| | cpu0-gold-usr |
| | cpu1-gold-usr |
| | cpu2-gold-usr |
| | cpu3-gold-usr |
| | (… 13 additional TSENS) |
| GPU Tj rule (95°C) | gpu-virt-max-step |
| POP-mem Tj rule (95°C) | pop-mem-step |
| VDD restriction rules (5°C) | cpu0-gold-lowf |
| | cpu1-gold-lowf |
| | cpu2-gold-lowf |
| | cpu3-gold-lowf |
| | cpu0-silver-lowf |
| | cpu1-silver-lowf |
| | cpu2-silver-lowf |
| | cpu3-silver-lowf |
| | (… 13 additional VDD restriction rules) |
| OEM $T_{skin}$ rules (optional) | To be added by OEM |

- Thermal sensor sdm845.dtsi example:

```
cpu1-gold-usr {
   polling-delay-passive = <0>;
   polling-delay = <0>;
   thermal-sensors = <&tsens0 8>;
   thermal-governor = "user_space";
   trips {
      active-config0 {
         temperature = <125000>;
         hysteresis = <1000>;
         type = "passive";
      };
   };
};
```

- Combination of sensor and rule sdm845.dtsi example:

```
pop-mem-step {
   polling-delay-passive = <10>;
   polling-delay = <0>;
   thermal-sensors = <&tsens1 2>;
   thermal-governor = "step_wise";
   trips {
      pop_trip: pop-trip {
         temperature = <95000>;
         hysteresis = <0>;
         type = "passive";
      };
   };
   cooling-maps {
      pop_cdev4 {
          trip = <&pop_trip>;
           cooling-device =<&CPU4
THERMAL_NO_LIMIT

(THERMAL_MAX_LIMIT-1)>;
      };
...
```

# PCB Thermistor Thermal Zones (sdm845-mtp.dtsi)

- Rules for $T_{skin}$ that use PCB thermistors must be placed in the OEM’s platform dtsi

| dtsi file | Purpose of thermal zones | PCB thermistor |
|---|---|---|
| sdm845-mtp.dtsi<br>(or OEM's platform dtsi) | PCB thermistor exposure | xo-therm-adc |
| | | msm-therm-adc |
| | | pa-therm1-adc |
| | | quiet-therm-adc |
| | OEM $T_{skin}$ rules which use PCB Therms (alternative to using thermal engine) | OEM defined |

- Xo_therm entry example:

```
xo-therm-adc {
   polling-delay-passive = <0>;
   polling-delay = <0>;
   thermal-sensors = <&pm8998_adc_tm 0x4c>;
   thermal-governor = "user_space";
   trips {
      active-config0 {
      temperature = <65000>;
      hysteresis = <1000>;
      type = "passive";
   };
 };
};
```

# BCL Thermal Zones

- BCL events are identified by sysfs node “type”

```
cat /sys/class/thermal_zone*/type
    Ibat-high     pmi—ibat-lvl0
    Ibat-vhigh    pmi-ibat-lvl1
    Vbat_adc      pmi-vbat-lvl0
    Vbat_low      pmi-vbat-lvl1
    Vbat_too_low  pmi-vbat-lvl2
    soc
```

For targets SDM632, SDM450, and SDM439

- The file pmi8998.dtsi contains BCL sensors and BCL rules

| dtsi file | Purpose of thermal zones | Entry name | Threshold | Action |
|---|---|---|---|---|
| pmi8998.dtsi | BCL sensors | ibat-high | 4.2 A | Not defined in pmi dtsi |
| | | ibat-vhigh | 4.3 A | Not defined in pmi dtsi |
| | | vbat_low | 3.1 V | Not defined in pmi dtsi |
| | | vbat_too_low | 2.9 V | Not defined in pmi dtsi |
| | | pmi8998_tz | 105/125/145 | Not defined in pmi dtsi |
| | BCL rules | vbat | 3.3 V | Core isolation for all Gold cores |
| | | soc | 10% battery | Core isolation for all Gold cores |

# BCL Device Tree

- Sample example for vbat_adc BCL event dt configuration
- Similarly, for other events:

| | |
|---|---|
| Ibat-high | pmi–ibat-lvl0 |
| Ibat-vhigh | pmi-ibat-lvl1 |
| Vbat_adc | pmi-vbat-lvl0 |
| Vbat_low | pmi-vbat-lvl1 |
| Vbat_too_low | pmi-vbat-lvl2 |
| soc | |

For targets SDM632, SDM450, and SDM439

pm660.dtsi

```
vbat_adc {
        polling-delay-passive = <100>;
        polling-delay = <0>;
        thermal-governor = "low_limits_cap";
        thermal-sensors = <&bcl_sensor 2>;
        tracks-low;

        trips {
            pm660_vbat_adc: vbat-adc {
                    temperature = <3200>;
                    hysteresis = <100>;
                    type = "passive";
                          };
                    };
        };
```

sdm670-thermal.dtsi

```
vbat_adc {
        cooling-maps {
            vbat_map {
                trip = <&pm660_vbat_adc>;
                cooling-device =
                    <&CPU6 THERMAL_MAX_LIMIT
                           THERMAL_MAX_LIMIT>;
            };
            vbat_map {
                trip = <&pm660_vbat_adc>;
                cooling-device =
                    <&CPU7 THERMAL_MAX_LIMIT
                           THERMAL_MAX_LIMIT>;
            };
        };
    };
```

# Thermal Zone Parameters

| | |
|---|---|
| Thermal Zone Name | `pop-mem-step {` |
| Polling rate after 95C interrupt received | `polling-delay-passive = <10>;` |
| Polling rate for non-interrupt driven sensors | `polling-delay = <0>;` |
| Thermal sensor being monitored (tsens 2 and controller 1) | `thermal-sensors = <&tsens1 2>;` |
| Thermal algorithm that will be used to throttle cooling dev | `thermal-governor = "step_wise";` |
| Trip point(s) definitions | `trips {` |
| Name of trip point | `pop_trip: pop-trip {` |
| Trip point value | `temperature = <95000>;` |
| Clear point relative to temperature above | `hysteresis = <0>;` |
| Passive cooling device (e.g. CPU, GPU) | `type = "passive";` |
| | `};` |
| | `};` |
| Cooling action definitions (i.e. action to take based off trip) | `cooling-maps {` |
| Name of cooling map | `pop_cdev4 {` |
| Reference to trip point that will activate cooling map | `trip = <&pop_trip>;` |
| Action taken based off of trip.<br>Throttle CPU4, no lower limit bound,<br>Max mitigation is last level - 1 | `cooling-device =<&CPU4 THERMAL_NO_LIMIT (THERMAL_MAX_LIMIT-1)>;`<br>`};` |



**Note:** The complete list of parameters is available at */kernel/msm-4.9/Documentation/devicetree/bindings/thermal/thermal.txt.*

# Thermal Zone Parameter Description

- Cooling-device parameter uses the following format:

```
cooling-device =<&<mitigation_device> <perf_ceiling> <perf_floor>>;
```

  Example: cooling-device =<&CPU4 7 8>;
  - perf_ceiling parameter is the highest allowable performance level (that is, the ceiling)
  - perf_floor is the lowest allowable performance level (that is, the floor)
  - In either case, a lower index means less mitigation and higher index means deeper mitigation; there can be multiple of these levels depending on rule
  - perf_ceiling value should always be <= to perf_floor
  - THERMAL_NO_LIMIT is a macro that can be used as a parameter for both perf_ceiling and perf_floor
    - If used for perf_ceiling, it means that there is no lower limit and the cooling device can be in cooling state 0
    - If used for perf_floor, it means that there is no upper limit and the cooling device can be in max state
  - THERMAL_MAX_LIMT is an index for the deepest mitigation state
- Example on Slide 25 *Thermal Zone Parameters*:

```
cooling-device =<&CPU4 THERMAL_NO_LIMIT(THERMAL_MAX_LIMIT-1)>;
```

  This entry allows CPU4 to be throttled to the 2nd to last mitigation point (THERMAL_MAX_LIMIT-1). After clearing, the threshold associated with the cooling-map, the device returns to a fully unmitigated state (THERMAL_NO_LIMIT).

# Thermal Sysfs

The thermal zone parameters are readable via adb through thermal sysfs

Cat /sys/class/thermal/<param>

- type – Sensor name
- mode – Monitoring is enabled or disabled
- policy – Currently configured thermal governor
- available_policies – Thermal governors available for this target
- trip_point_#_temp – Trip temperature for trip number
- trip_point_#_hyst – Trip temperature hysteresis for trip number # (Hysteresis is relative to trip temperature and it is not an absolute trip temperature)
- trip_point_#_type – Trip type for the trip number
- cdev#/type – Cooling device name
- cdev#_trip_point – Trip number that this cooling device is associated to
- temp – Sensor reading

```
sdm845:/sys/class/thermal # cd thermal_zone23
sdm845:/sys/class/thermal/thermal_zone23 # ls
available_policies  integral_cutoff  offset             temp
cdev0               k_d              policy             trip_point_0_hyst
cdev0_lower_limit   k_i              power              trip_point_0_temp
cdev0_trip_point    k_po             slope              trip_point_0_type
cdev0_upper_limit   k_pu             subsystem          type
cdev0_weight        mode             sustainable_power  uevent
```

**Note**: integral_cutoff, k_d, k_i, k_po, k_pu are PID features that are currently not used by step_wise.

# Thermal Core Governors

# Thermal Core Governors

- What is a thermal governor?
  - The concept of thermal governor is same as algorithm type in thermal engine
  - Governor is a temperature monitor algorithm that controls the temperature of a thermal zone within a limit by mitigating the cooling devices associated with the zone
- Supported thermal core governors:
  - Step-wise
    - Equivalent to QTI simple step
    - Capable of monitor algorithm type used with thermal engine
  - User space: Notifies thermal engine of threshold trips
  - Low temperature monitor governors
    - low_limits_floor handles VDD restriction by placing a perf floor on cooling device
    - low_limits_cap handles SOC and VBat conditions by placing a perf ceiling on cooling device

# Step Wise Governor

- Thermal zone descriptor: `thermal-governor = "step_wise";`
- Governor can monitor for trip point thresholds and mitigate multiple cooling devices.
- After crossing a trip threshold, governor steps up or down the mitigation level by one step for each polling iteration.
- The associated cooling device can specify the minimum and maximum cap for the governors mitigation action for that trip threshold.
- Monitors the following trip point types for a thermal zone:
  - THERMAL_TRIP_PASSIVE – Passive refers to passive cooling device, that is, reducing CPU or GPU frequency.
  - THERMAL_TRIP_ACTIVE – Active refers to active cooling devices, that is, a fan.
  - THERMAL_TRIP_CRITICAL – Triggers a shutdown by a call to orderly_poweroff()
- Naming convention: Thermal zones that use this governor have ‘step’ as part of the zone name that is defined in the device tree.
  Example: pop-mem-step

# Step Wise Governor – dtsi Config

```
thermal-zones {
cpu0-tsens{
            polling-delay-passive = <10>;
            polling-delay = <0>;
            thermal-governor=<step_wise>;
            thermal-sensors = <&tsens0 6>; /* tsens controller and the sensor tsens ID*/
            trips {
                        mitig_level0: mitig_level0 {
                                    temperature = <95000>;
                                    type = ”passive” ;
                        } ;
                         mitig_level1: mitig_level1 {
                                    temperature = <100000>;
                                    type = ”passive” ;
                        } ;
            } ;
            cooling-maps {
                        map0 {
                                    trip = <& mitig_level0>;
                                    cooling-device = <&CPU0 THERMAL_NO_LIMIT 4>;
                        } ;
                        map1 {
                                    trip = <& mitig_level1>;
                                    cooling-device = <&CPU0 5 THERMAL_NO_LIMIT>;
                        } ;
            } ;
      } ;
} ;
```



# Step Wise as Monitor Governor

- Stepwise governor can be extended to work like thermal engine's monitor algorithm.
- Uses "step_wise" as governor type

```
thermal-governor = "step_wise";
```

- When the upper and lower mitigation limits for a cooling device are same, then this governor mitigates the cooling device to the particular state and does not step up or down.
- When the hysteresis temperature is added, the stepwise governor does not release the mitigation until the temperature goes below the hysteresis threshold.



# Step Wise as Monitor Governor – dtsi Config

```
thermal-zones{
        cpu0-tsens{
                polling-delay-passive=<10>;
                polling-delay=<250>;
                thermal-governor=<step_wise>;
                thermal-sensors=<&tsens0 6>; /* tsens controller and the sensor tsens ID*/
                trips{
                        mitig_level0: mitig_level0 {
                                temperature=<95000>;
                                hysteresis = <10000>;
                                type="passive" ;
                        };
                        mitig_level1: mitig_level1 {
                                temperature=<100000>;
                                hysteresis = <5000>;
                                type=" passive" ;
                        };
                };
                cooling-maps{
                        map0{
                                trip=<& mitig_level0>;
                                cooling-device=<&CPU0 4 4>;
                        };
                        map1{
                                trip=<& mitig_level1>;
                                cooling-device=<&CPU4 5 5>;
                        };
                };
        };
};
```

Lower Mitigation Limit

Upper Mitigation Limit

- Lower limit is the highest allowable performance level (that is, the perf ceiling). Here, lower means lower index.
- Upper limit is the lowest allowable performance level (that is, the perf floor). Here, upper means higher index.

**Note:** Since upper and lower mitigation values are the same, the rule behaves like a monitor rule, staying at level 4 until hysteresis is reached.

# User Space Governor

- Thermal zone descriptor:

  ```
  thermal-governor = “user_space";
  ```
- The user space governor’s role is to notify thermal engine of a thermal zone threshold being reached.
- Any thermal sensor that is to be used by the thermal engine must use the user space governor.
- The user space daemon will set the thresholds for each trip type using the thermal core sysfs.
- Thermal core communicates the trip to sensor driver and sensor driver should notify when the threshold is crossed.
- Once thermal core gets the threshold trip notification, thermal core will notify user space by triggering a uevent.
- Thermal zones that use this governor have “usr” as part of the zone name that is defined in the device tree.

  Example: cpu0-gold-usr

**Note:** Do not modify default device tree entries that use the user space governor.

# Low Temperature Monitor Governors

- Thermal zone descriptor:
  - `thermal-governor = “low_limits_floor";`
  - `thermal-governor = “low_limits_cap”;`
- Monitors the temperature and triggers mitigation when the temperature falls below a trip point
- low_limits_floor – For VDD restriction, places a floor mitigation; sensor using this governor has “-lowf”
- low_limits_cap – For SOC and VBat, places a scaling cap mitigation; sensor using this governor has “-lowc”

# Virtual Sensors

# How Does it Work?

- Any sensor that needs to be monitored by two different governors requires a new virtual sensor that monitors the sensor.
- Two different types of virtual sensors:
  - Virtual sensor for adding OEM custom thermal zones and rules.
    - OEM custom thermal zones require that a new thermal zone be created because the sensor will have already been used once with the user space governor.

      **Note: I**f you are adding a rule that uses the same governor, you can either simply extend the existing rule that uses that sensor in the dtsi file, or you can create a entirely new rule.
  - Virtual sensor for aggregating temperatures from multiple sensors
    - Selectable aggregation logic: maximum, minimum and coefficient-offset logic.
- Naming convention: Thermal zones that are virtual sensors have ‘virt’ as part of the zone name that is defined in the device tree.

  Example: gold-virt-max-step
  - Aggregation logic is included in name (for example, max in this example): max, min, coeff

# Define Virtual Sensors

Defining virtual sensor requires two entries:

- Thermal zone entry in the device tree file that contains the sensor type
  - Example: For PCB thermistors, use OEM's platform dtsi (typically, sdm845-mtp.dtsi)
  - Thermal zone entry is the same format as other thermal zones
  - No special flag needs to be defined for creating virtual sensor
- QTI virtual sensor file for setting aggregation logic (only required if virtual sensor needs to reference multiple sensors)
  - /kernel/msm-4.x/drivers/thermal/qcom/qti_virtual_sensor.c
  - Example: One of the default virtual sensors reports the maximum of the two GPU TSENS

```
{
   .virt_zone_name = "gpu-virt-max-step",  ← Use same name as was defined in sdm845-mtp.dtsi
   .num_sensors = 2,
   .sensor_names = {"gpu0-usr", "gpu1-usr"},  ← Sensor names
   .logic = VIRT_MAXIMUM,  ← Aggregation logic
},
```

# Virtual Sensor Example

- Platform device tree entry

```
gpu-virt-max-step {
   polling-delay-passive = <10>;
   polling-delay = <100>;
   thermal-governor = "step_wise";
   trips {
      gpu_trip0: gpu-trip0 {
      temperature = <95000>;
      hysteresis = <0>;
      type = "passive";
       };
    };
    cooling-maps {
      gpu_cdev0 {
         trip = <&gpu_trip0>;
         cooling-device = <&msm_gpu
         0 THERMAL_NO_LIMIT>;
        };
     };
};
```

- Qti_virtual_sensors.c entry

```
{
   .virt_zone_name = "gpu-virt-max-
   step", .num_sensors = 2,
   .sensor_names = {"gpu0-usr",
   "gpu1-usr"},
   .logic = VIRT_MAXIMUM,
},
```

**Note**: Virtual sensors such as gpu-virt-max-step use both the polling-delay-passive and polling-delay field. Since a virtual sensor is a non-interrupt driven sensor, it continuously polls the sensor (100 msec in this example) until it crosses the threshold set; then it switches to the polling-delay-passive rate of 10 msec.



# Debugging

# Debugging Thermal Core

- Ftrace events for thermal core framework: /d/tracing/events/thermal

| Ftrace event | Description |
|---|---|
| thermal_zone_trip | Trip event marker |
| thermal_device_update | The governor polling loop start marker |
| thermal_temperature | The temperature read by a zone |
| cdev_update_start | Cooling device mitigation update start marker |
| cdev_update | Cooling device mitigation update exit marker |
| thermal_handle_trip | Governor polling loop end marker |
| thermal_set_trip | Thermal set trip temperatures |

- There is an additional trace event for virtual sensors available in /d/tracing/events/thermal_virtual/. This prints the individual temperature of all the sensors and the computed virtual sensor reading.
- Ftrace prints the hardware aggregated frequency that is notified to the scheduler: /d/tracing/events/lmh/
- Thermal core kernel debug logs:

```
echo 'file thermal_core.c +p' > /sys/kernel/debug/dynamic_debug/control
```

# Debugging Thermal Core (Sample Ftrace)

For example, set the Silver cluster temperature threshold range from 30°C to 35°C

- Thermal mitigation triggering:

thermal_zone type and ID

```
irq/122-tsens-u-98    [000] ....  7266.585744: thermal_device_update: thermal_zone=cpu0-silver-step id=25 received event:0
irq/122-tsens-u-98    [000] ....  7266.585747: thermal_query_temp: thermal_zone=cpu0-silver-step id=25 temp=35000
irq/122-tsens-u-98    [000] ....  7266.585748: thermal_temperature: thermal_zone=cpu0-silver-step id=25 temp_prev=28200 temp=35000
irq/122-tsens-u-98    [000] ....  7266.585756: thermal_set_trip: thermal_zone=cpu0-silver-step id=25 low trip=30000 high trip=2147483647
irq/122-tsens-u-98    [000] ....  7266.585762: thermal_zone_trip: thermal_zone=cpu0-silver-step id=25 trip=0 trip_type=PASSIVE
irq/122-tsens-u-98    [000] ....  7266.585767: cdev_update_start: type=thermal-cpufreq-0 update start
irq/122-tsens-u-98    [000] ....  7266.585964: cdev_update: type=thermal-cpufreq-0 target=3 min_target=18446744073709551615
```

Target state

- Thermal mitigation releasing:

```
kworker/u17:1-418   [003] ....  7355.478937: thermal_device_update: thermal_zone=cpu0-silver-step id=25 received event:0
kworker/u17:1-418   [003] ....  7355.478991: thermal_query_temp: thermal_zone=cpu0-silver-step id=25 temp=29500
kworker/u17:1-418   [003] ....  7355.478999: thermal_temperature: thermal_zone=cpu0-silver-step id=25 temp_prev=29800 temp=29500
kworker/u17:1-418   [003] ....  7355.479056: thermal_set_trip: thermal_zone=cpu0-silver-step id=25 low trip=-2147483647 high trip=35000
kworker/u17:1-418   [003] ....  7355.479079: thermal_zone_trip: thermal_zone=cpu0-silver-step id=25 hyst=0 trip_type=PASSIVE
kworker/u17:1-418   [003] ....  7355.479106: cdev_update_start: type=thermal-cpufreq-0 update start
kworker/u17:1-418   [003] ....  7355.479477: cdev_update: type=thermal-cpufreq-0 target=0 min_target=18446744073709551615
kworker/u17:1-418   [003] ....  7355.479486: thermal_handle_trip: thermal_zone=cpu0-silver-step id=25 handle trip=0
```

# References

| Acronyms | |
|---|---|
| **Acronym or term** | **Definition** |
| KTM | Kernel thermal monitor |
| LMH | Limits management hardening |

# Questions?

https://createpoint.qti.qualcomm.com