Qualcomm Technologies, Inc.

DragonBoard™ 410c based on Qualcomm® Snapdragon™ 410E processor

Thermal Debugging Guide

September 2016

Questions or comments: https://www.96boards.org/DragonBoard410c/forum

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

LM80-P0436-8 Rev D

Revision history

Revision	Date	Description
D	September 2016	Updated to ‘E’ part.
C	July 10, 2015	Miscelaneous updates
B	May 22, 2015	URL Update
A	May 6, 2015	Initial release

Contents

Figures

Tables

1 Introduction

1.1 Purpose

This document provides guidance for debugging some of the most common thermal-related issues from the software perspective for developers using DragonBoard 410c based on Snapdragon 410E (APQ8016E). To provide sufficient context for the discussion on thermal debugging, this document includes information on the two different thermal instantiations: Kernel Thermal Monitor (KTM) and thermal engine.

- Thermal management is to manage:
  - silicon junction temperature limits.
  - memory case temperature limits.
  - external surface temperature limits.
- KTM keeps die temperatures within limits under all ambient conditions when full thermal engine is initialized.
- Thermal engine monitors temperature limits across the whole system.
- Simulation of mechanical design is the most important step in achieving best performance.
- Thermal management software controls thermal response.

1.2 Acronyms, abbreviations, and terms

Table 1-1 provides definitions for the acronyms, abbreviations, and terms used in this document.

Table 1-1 Acronyms, abbreviations, and terms

Term	Definition
ADC	Analog-to-Digital Converter
CPU	Central Processing Unit
DCVS	Dynamic Clock Frequency and Voltage Scaling
DTM	Digital Terrain Map
GFX	Grafix
GPU	Graphics Processing Unit
IOCTL	Input/Output Control
KTM	Kernel Thermal Monitor
LA	Linux Android
PFM	Pulse Frequency Modulation

Term	Definition
PID	Proportional-Integral-Derivative
PMIC	Power Management Intigrated Circuit
POP	Point of Presence
PSM	Phase Shift Modulation
PWM	Pulse Width Modulation
RAM	Random Access Memory
SS	Single Step
TM	Thermal Management
TSENS	Temperature sensor

1.3 Architecture

Figure 1-1 shows the four blocks of the thermal management framework: the thermal engine, sensor drivers, and other thermal management devices. In addition to these key components, there are management components addressing boot and kernel initialization.

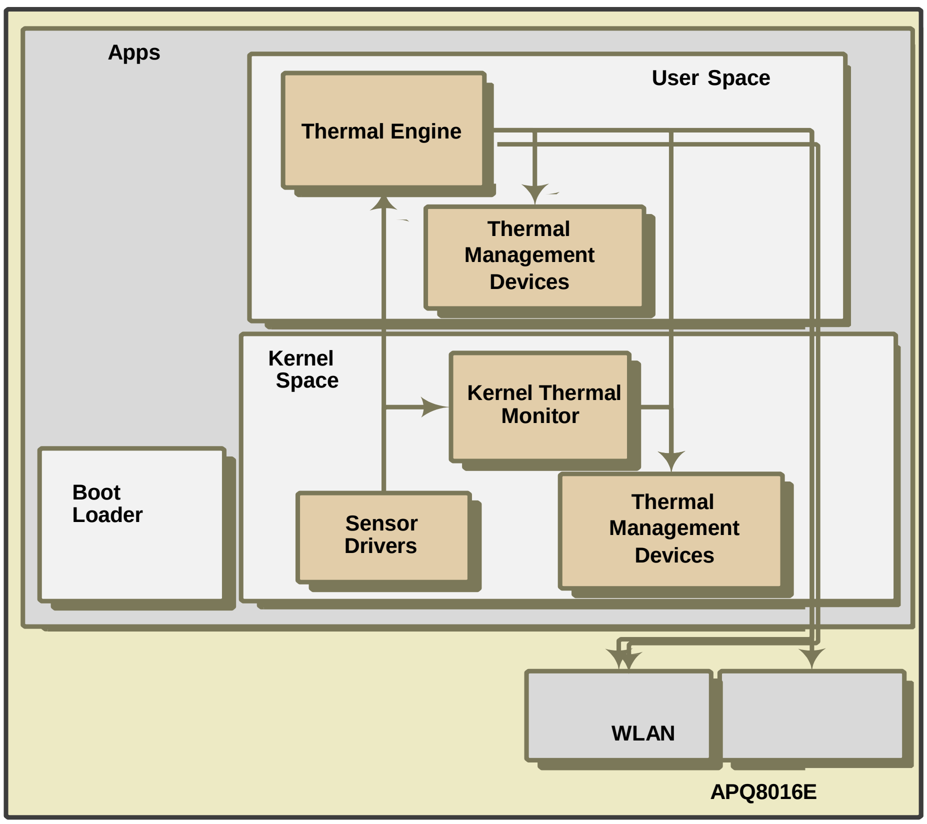

Figure 1-1 Thermal software architecture

The thermal engine runs as a super-user process in the Linux Android (LA) user space and is the central controller for thermal management. The thermal engine initializes the system on startup. The configuration of thresholds, set points, and management devices is read from defaults in the code. The parameters are used to set interrupt thresholds in the hardware for temperature sensors.

These parameters need to be tuned for each unique industrial design to achieve maximum performance levels while maintaining desired thermal specifications.

The sensor inputs for temperature readings come from sources in the reference design:

- Temperature sensors embedded in the APQ8016 die, which are connected to a hardware component called TSENS.
- Thermal management devices are software components that control hardware with high-power densities, i.e., CPUs and GPUs.

Along with the above architectural elements, two thermal algorithms are implemented to cover device boot and Linux kernel initialization prior to enabling the main thermal engine. The kernel thermal driver manages the CPU cluster’s performance during kernel initialization to ensure thermal limits are maintained.

1.4 Android thermal limits and features

- KTM
  - Protects system during kernel boot time
  - Sets a single 110ºC threshold for emergency CPU mitigation and CPU hot plug
  - Hands control over to thermal engine
- Thermal engine
  - Full-fledged thermal protection
  - Must be tuned for specific targets
- Thermal reset
  - Happens unexpectedly

1.5 Thermal management (TM) handover timeline

- Graceful handover between KTM and thermal engine

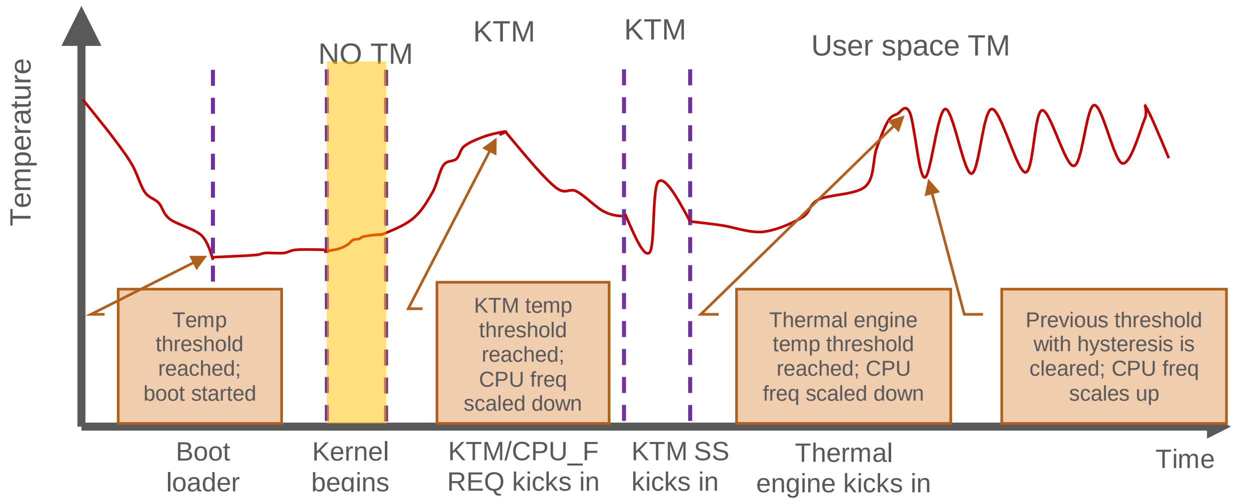

Figure 1-2 Thermal management handover timeline

1.6 Additional information

For additional information, go to http://www.96boards.org/db410c-getting-started/.

2 KTM

Kernel Thermal Monitor (KTM) keeps die temperatures within limits under all ambient conditions when full thermal engine is initialized. Full thermal engine monitors temperature limits across the whole system.

2.1 KTM functions and associated debugging

- Continuously checks current temperature from specified TSENS and performs the following actions:
  - **check_temp()** – In /drivers/thermal/msm_thermal.c; called every sampling period defined as msm_thermal_info.poll_ms.
  - **do_therm_reset()** – If any of the temperature sensors crosses a critical threshold, it causes a secure watchdog bite whose parameter is configured using the device tree binding, qcom,therm-reset-temp. This feature helps in debugging by generating a RAM dump with caches flushed as opposed to the RAM dump generated by the hardware reset. The following message will be observed:

    ```
    msm_thermal:msm_thermal_bite: TSENS:α reached temperature:β. System reset
    ```

  - **therm_get_temp()** – Retrieves the temperature on the specified sensor using the device tree binding, qcom, sensor_id. The variable temp needs to be printed to see the current temperature only for debugging purposes.
  - **do_core_control()** – Unplugs cores when a threshold is crossed over. KTM prints the following kernel messages when it does core control:

    ```
    msm_thermal:do_core_control: Set Offline: CPU$ Temp: β
    msm_thermal:do_core_control: Allow Online CPU$ Temp: β
    ```

  - **do_vdd_mx()** – For some LINUX ENABLEMENT targets, KTM monitors all of the temperature sensors. If the temperature falls below a certain threshold, it votes for an increased Memory rail voltage. KTM prints the following message when it receives the Memory threshold notification:

    ```
    msm_thermal: vdd_mx_notify: Sensorα trigger received for type
    <threshold_type>
    ```

  - **do_psm()** – For PMIC automodedisablement when a threshold is crossed over and KTM sends a command to make PMIC operate in Pulse Width Modulation (PWM) mode. KTM prints the following messages when it sends the PWM/Auto mode command:

    ```
    msm_thermal:do_psm: Requested PMIC PWM Mode tsens:α. Temp:β
    msm_thermal:do_psm: Requested PMIC AUTO Mode
    ```

  - **do_gfx_phase_cond() and do_cx_phase_cond()** – For multiphase support for DIGITAL/GFX rails, KTM votes for various temperature bands to RPM. RPM takes in the temperature band input for rails and, based on other factors, it decides the number of phases required for the rail.

```
msm_thermal:send_temperature_band: Sending <rail> temperature band
<band_number> where, <rail>: DIGITAL or GFX with multiple BAND definition
depending on chipset
```

- **do_ocr()** – For some targets, KTM monitors the temperature sensors. If the temperature of any sensor exceeds a threshold, it sends an optimum current value request to a set of regulators.

  This request is used in some targets. For example, for MSM8x16 this request is used by the PMIC to decide whether to operate in PWM (high threshold) or PFM (low threshold) mode. KTM prints the following message when it votes for an optimum current request:

  ```
  msm_thermal:request_optimum_current: Requested optimum current mode:
  <opt_curr_mode>
  ```

- **do_vdd_restriction()** – For limiting low voltage/frequency when the temperature goes below a threshold (5ºC). KTM prints the below kernel messages when it does VDD restriction during boot.

  ```
  msm_thermal:vdd_restriction_notify: sensor:α reached high thresh for
  Vddrestriction
  ```

  ```
  msm_thermal:vdd_restriction_notify: sensor:α reached low thresh for
  Vddrestriction
  ```

  KTM exposes a sysfsnode, through which thermal engine can vote for VDD restriction.

  ```
  cat /sys/module/msm_thermal/vdd_restriction/enabled
  ```

- **do_freq_control()** – For CPU frequency control when a threshold is crossed over. KTM prints the below kernel messages when it mitigates the CPU frequency.

  ```
  msm_thermal:do_freq_control: Limiting CPU$ max frequency to 1958400.
  Temp:β
  ```

2.2 KTM configuration

Verify various parameter sets in qcom,msm-thermal (the following example, `/arch/arm64/boot/dts/qcom/msm8916.dtsi`, is MSM8916-specific).

```
qcom,msm-thermal {
       qcom,msm-thermal {
                compatible = "qcom,msm-thermal";
                qcom,sensor-id = <5>;
                qcom,poll-ms = <250>;
                qcom,limit-temp = <60>;
                qcom,temp-hysteresis = <10>;
                qcom,freq-step = <2>;
                qcom,freq-control-mask = <0xf>;
                qcom,core-limit-temp = <80>;
                qcom,core-temp-hysteresis = <10>;
                qcom,core-control-mask = <0xe>;
                qcom,hotplug-temp = <94>;
                qcom,hotplug-temp-hysteresis = <15>;
                qcom,cpu-sensors = "tsens_tz_sensor5", "tsens_tz_sensor5",
                                "tsens_tz_sensor4", "tsens_tz_sensor4";
```

```
                qcom,freq-mitigation-temp = <94>;
                qcom,freq-mitigation-temp-hysteresis = <10>;
                qcom,freq-mitigation-value = <400000>;
                qcom,freq-mitigation-control-mask = <0x01>;
                qcom,online-hotplug-core;
                qcom,vdd-restriction-temp = <5>;
                qcom,vdd-restriction-temp-hysteresis = <10>;
                vdd-dig-supply = <&pm8916_s1_floor_corner>;

                qcom,vdd-dig-rstr{
                        qcom,vdd-rstr-reg = "vdd-dig";
                        qcom,levels = <5 7 7>; /* Nominal, Super Turbo, Super
Turbo */
                        qcom,min-level = <1>; /* No Request */
                };

                qcom,vdd-apps-rstr{
                        qcom,vdd-rstr-reg = "vdd-apps";
                        qcom,levels = <533330 800000 998400>;
                        qcom,freq-req;
                };
        };
```

An embedded temperature sensor used to control the algorithm is defined as CPU0 sensor. If the temperature exceeds the specified level in limit-temp, the maximum allowed CPU frequency will be reduced. The reduction continues while temperature is above the limit at each polling interval. The polling interval is defined in the poll-ms field. If the temperature drops below the sum of the limit-temp plus the temp-hysteresis, then the maximum allowed CPU frequency will be increased. The changes in CPU frequency up or down are one step in the DCVS table frequencies per sample period.

In addition to CPU frequency scaling, a secondary temperature threshold, core-limit-temp, defines the limit at which the CPU hotplug is invoked to take CPUs offline. This feature works the same as frequency scaling in that each sample period another decision is made to either hotplug cores on or offline based on the temperature readings in relation to the limits.

There are two fields in the device tree, freq-control-mask and core-control-mask, that define which cores are acted upon by the feature. Bit 0 of the mask corresponds to CPU0. Note that by default the core-control-mask does not include CPU0 as this core cannot be hotplugged.

The algorithm is contained in /drivers/thermal/msm_thermal.c and the associated parameters are defined in arch/arm64/boot/dts/qcom/msm8916.dtsi.

Verifying KTM frequency mitigation and KTM handoff to thermal engine.

- Kernel log will display actions taken.

```
<6>[ 1.243247] msm_thermal: Limiting cpu0 max frequency to 1209600
<6>[ 1.243277] msm_thermal: Limiting cpu1 max frequency to 1209600
<6>[ 1.243277] msm_thermal: Limiting cpu2 max frequency to 1209600
<6>[ 1.243277] msm_thermal: Limiting cpu3 max frequency to 1209600
```

```
<6>[ 1.493301] msm_thermal: Limiting cpu0 max frequency to 1152000
<6>[ 1.493331] msm_thermal: Limiting cpu1 max frequency to 1152000
<6>[ 1.493331] msm_thermal: Limiting cpu2 max frequency to 1152000
<6>[ 1.493331] msm_thermal: Limiting cpu3 max frequency to 1152000
<6>[ 1.743384] msm_thermal: Limiting cpu0 max frequency to 1094400
<6>[ 1.743415] msm_thermal: Limiting cpu1 max frequency to 1094400
<6>[ 1.743415] msm_thermal: Limiting cpu2 max frequency to 1094400
<6>[ 1.743415] msm_thermal: Limiting cpu3 max frequency to 1094400
<6>[ 1.993438] msm_thermal: Limiting cpu0 max frequency to 998400
<6>[ 1.993468] msm_thermal: Limiting cpu1 max frequency to 998400
<6>[ 1.993468] msm_thermal: Limiting cpu2 max frequency to 998400
<6>[ 1.993468] msm_thermal: Limiting cpu3 max frequency to 998400
<6>[ 2.243491] msm_thermal: Limiting cpu0 max frequency to 800000
<6>[ 2.243491] msm_thermal: Limiting cpu1 max frequency to 800000
<6>[ 2.243491] msm_thermal: Limiting cpu2 max frequency to 800000
<6>[ 2.243491] msm_thermal: Limiting cpu3 max frequency to 800000
      …………………………………………………………………………………………….
<6>[ 13.025240] msm_thermal: Max frequency reset for cpu0
<6>[ 13.029940] msm_thermal: Max frequency reset for cpu1
<6>[ 13.030337] msm_thermal: Max frequency reset for cpu2
<6>[ 13.047855] msm_thermal: Max frequency reset for cpu3
<6>[ 13.052891] msm_thermal: enabled = 0// thermal engine kicked in
```

3 Thermal Engine

This daemon monitors thermal/temperature sensor data and performs actions based on a configuration file (default is /etc/thermal-engine.conf).

3.1 Architecture

The thermal engine is the main thermal monitor running during device operation. It resides in the Linux user space as a daemon process and performs two main functions:

- Monitoring temperature
- Controlling management devices

Two components provide these functions:

- The sensor manager, which monitors temperature
- The device manager, which sends commands to management devices to control performance

In addition to the two managers, the thermal engine also runs two thermal algorithms:

- Threshold control
- Dynamic control

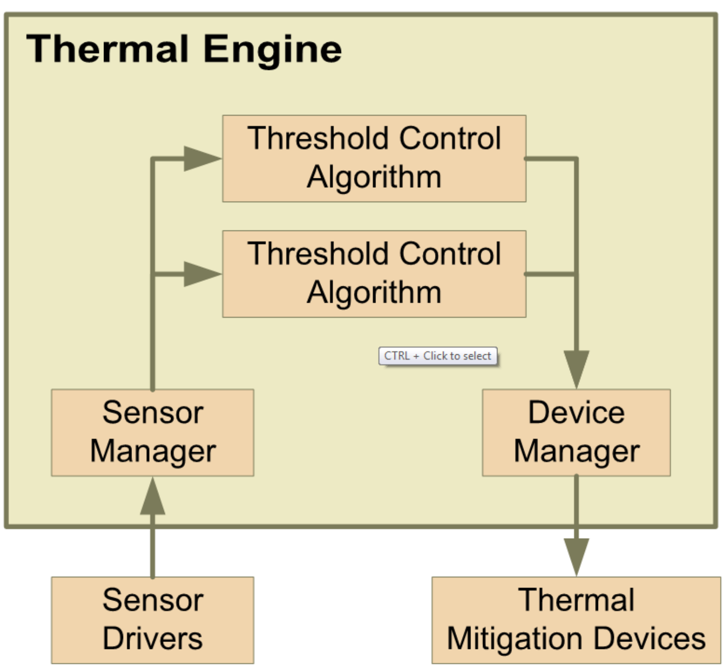

Figure 3-1 Thermal engine

3.2 Thermal engine rules – embedded vs. configuration file

1. Start three different algorithms
   - SS (Single Step), PID (Proportional-Integral-Derivative), and monitor.
   - All the rules are defined either in the thermal engine config file or hardcoded in thermal engine data files.
     - Embedded (hardcoded) rules
       - Required

         Voltage restriction

         PSM control
       - Optional (can be disabled or overridden granted junction limit is not violated)

         SS/PID control on all the CPUs
     - User-defined rules in thermal-engine.conf
       - Rules to maintain junction temperature limit generally through SS algorithm
       - Rules to maintain pop memory case temperature limit generally through either monitor or SS algorithm
       - Rules to maintain skin temperature limit generally through either monitor or SS algorithm
   - Thermal engine initializes these algorithms with predefined configurations.
2. Sensor management
   a. Tsens/PMIC ADC sensors are fully interrupt driven.
   b. Can add external thermistor either interrupt driven or polling based.

3.3 Example of SS/PID/MONITOR algorithms

3.3.1 SS algorithm modem control instance

In this example, the label is surface_control_dtm. This name must be unique from other instance labels. The SS algorithm (algo_type = ss) controls the temperature by sampling sensor ID 3 (sensor = tsens_tz_sensor3) once per second (sampling = 1000). The DTM control adjusts the maximum allowed CPU frequency of all CPUs together (device = cpu). Sensor ID 3 is by the display subsystem, where the thermal response of the sensor was determined to be offset from the systems surface temperature by 25°C. Thus, the controlling set point to maintain the surface temperature at 45°C is set at 70°C. The clearing set point is set at 55°C, which is the temperature where the DTM controller will stop adjusting the maximum allowed frequency and let it return to maximum.

```
 [surface_control_dtm]
algo_type ss
sensor tsens_tz_sensor3
device cpu
sampling 1000
```

```
set_point 70000
set_point_clr 55000
```

3.3.2 PID algorithm instance example

In this example, the label is surface_control_pid. This name must be unique from other instance labels. The PID algorithm (algo_type = pid) controls the temperature by sampling sensor ID 5 (sensor = tsens_tz_sensor5) at 65 ms (sampling = 65). The PID control adjusts the maximum allowed CPU frequency of CPU0 (device = cpu0). Sensor ID 5 is at the hotspot of CPU0, thus, the controlling set point maintains the CPU0 junction temperature at 95°C. The clearing set point is set at 55°C, which is the temperature where the PID controller will stop adjusting the maximum allowed frequency based on this sensor and let it return to maximum.

```
[CPU0_control_pid]
algo_type pid
sensor tsens_tz_sensor5
device cpu0
sampling 65
set_point 95000
set_point_clr 55000
```

3.3.3 Monitor algorithm modem control instance

This example defines a modem control instance using a thermistor as the sensor. The monitor algorithm (algo_type = monitor) controls the temperature by sampling the PA thermistor 0 (sensor = cpu0-1) once per second (sampling = 1000). The thermistor 0 was determined to be offset from the systems surface temperature by 30°C. Thus, thresholds are configured to reduce the modem throughput and maximum allowed Tx power. The first threshold group is configured at 70°C. In this group, the modem data throughput reduction algorithm is triggered (action_info = 1). The second threshold group is configured at 80°C. In this group, the modem Tx power reduction algorithm is triggered (action_info = 2).

```
[modem]
algo_type monitor
sensor pa_therm0
sampling 1000
thresholds 70000 80000
thresholds_clr 65000 75000
actions modem mode
```

3.4 Thermal engine debugging

Thermal engine generates detailed logging of its state and actions when debug mode is enabled. This logging is output to the main Android logcat.

To enable debug mode:

1. Find the current thermal engine Configuration.

   ```
   adb shell thermal-engine –o > thermal-engine.conf
   ```

2. Modify the thermal-engine.conf file and push the updated configuration file to the device.

3. Put “debug” in the first line of thermal-engine.conf and restart thermal-engine service.

```
adb shell stop thermal-engine
adb root
adb remount
adb push thermal-engine.conf /system/etc/thermal-engine.conf
adb shell sync .
adb shell strat thermal-engine --debug &
```

4. Logcat log and temperature logging

```
adb logcat –v time –s ThermalEngine

I/ThermalEngine( 4555): Thermal daemon started
I/ThermalEngine( 4555): No target config file, falling back to
'/system/etc/thermal-engine.conf'
I/ThermalEngine( 4555): devices_manager_init: Init
I/ThermalEngine( 4555): Number of gpus :1
I/Thermal-IOCTL( 4555): KTM IOCTL interface "/dev/msm_thermal_query" opened
E/ThermalEngine( 4555): update_cpu_topology: Cluster Info[0]. Cluster Id: 0
cpu_bits:0xf sync:0
I/ThermalEngine( 4555): vdd_rstr_init: Init KTM VDD RSTR enabled: 0
I/ThermalEngine( 4555): sensors_manager_init: Init
I/ThermalEngine( 4555): qmi: Instance id 157 for fusion TS
I/ThermalEngine( 4555): MODEM thermal mitigation available.
I/ThermalEngine( 4555): ACTION: MODEM - Pending request: pa mitigation
succeeded for level 0.
I/ThermalEngine( 4555): Mitigation:Modem:0
E/ThermalEngine( 4555): bcl_setup: Unexpected node error
E/ThermalEngine( 4555): add_tgt_sensors_set: Error adding bcl
E/ThermalEngine( 4555): sensors_init: Error adding BCL TS
I/ThermalEngine( 4555): ACTION: MODEM - Pending request:
cpuv_restriction_cold mitigation succeeded for level 0.
I/ThermalEngine( 4555): Mitigation:VDD[MODEM-cpuv_restriction_cold]:0
I/ThermalEngine( 4555): Loading config file for virtual sensor
I/ThermalEngine( 4555): Loading configuration file /system/etc/thermal-
engine.conf
I/ThermalEngine( 4555): Parsing section global
I/ThermalEngine( 4555): Found field 'debug'
I/ThermalEngine( 4555): Debug output enabled from config
I/ThermalEngine( 4555): Found field 'sampling'
```

3.5 Temperature logging

- Internal sensors logging.
  - Mostly exposed to sysfs node and read values through regular file accesses.
  - A logging script or logging program to periodically read sensor values and store these on external storage to retrieve later.
  - All the frequency information (current frequency and maximum frequency) needs to be logged as well.

```
// checking for temp zone 0 value if sensor available
          if (tz_flags[0]) {
                 tz_temp= 0;
                 tzs=
fopen("/sys/devices/virtual/thermal/thermal_zone0/temp","r");
                 if(tzs) {
                        fscanf(tzs,"%d",&tz_temp);
                        if (debug) {
                                      printf("\nReadTEMPZONE0
file %d\n",tz_temp);
                        }
                        fclose(tzs);
                 }
                 fprintf(out_fd,"%d,",tz_temp);
          }
```

- POP memory/skin temperature logging.
  - Case temperature and/or skin temperature can be monitored using thermocouples or IR camera.
- Read the current Temperature.

```
adb shell cat /sys/devices/virtual/thermal/thermal_*/temp
```

3.6 tsens_reset

tsens_reset is a hardware-triggered reset designed to prevent device damage. The reset occurs when TM fails to properly react to thermal behavior.

How it can happen:

- Thermal engine is a user space daemon and started as system service.
- Thermal engine is the process with the highest priority.
- Thermal engine cannot run properly under certain situations.
  - Gets starved because there are too many high-priority processes or no chance to get scheduled due to any scheduling issues.
    - Case 1 – Too many high-priority processes and thermal engine lost the chance to kick in timely. tsens_reset happens as thermal engine cannot get scheduled.

- Case 2 – A specific debug-only service is enabled unnecessarily and it used up all the CPU resources for quite a long time and then tsens_reset happens.

- Legacy priority inversion problem.
  - Case 3 – A low-priority process holds a mutex but thermal engine kicks in and tries to lock the same mutex, which is already held. A mid-priority process (long lasting one) gets scheduled and thermal engine had no chance to proceed.

Consequently, temperature rises continuously. The issue can be resolved when the low-priority process is boosted (through priority inheritance or priority ceiling), removing any possibility of user space mutex causing this issue.

Fundamental solution:

- Emergency throttling in kernel space.

3.7 Kernel emergency throttling – no more tsens_reset

No tsens_reset is expected. However, if it happens:

1. Identify why thermal engine had not been properly mitigating CPUs.

- Go back to thermal engine debugging.

2. Decouple thermal engine and KTM.
   a. Try to reproduce the issue by stopping thermal engine.
   b. If reproducible, debug KTM.
3. Else, the issue mostly happens due to interaction between thermal engine and KTM.
   a. Need to look at TSENS threshold values and enable/disable status.
   b. IOCTL interface.
   c. Use hot chamber to expedite the reproduction of the issue.

3.8 Thermal engine client/server issues

Other user space processes make the following requests to the thermal engine through the thermal engine client interface:

- **Override** – Override threshold values
- **Speaker** – Speaker coil calibration
- **Dynamic parameter update** – Updates the thermal engine parameter at runtime

However, the calling processes should have proper privileges to request thermal engine to proceed.

- Currently, only processes with root or system permissions are allowed.
- Otherwise, the request will be rejected from thermal engine client code.
- To debug, thermal engine needs to run in debug mode and search “thermal” keyword on logcat log.

EXHIBIT 1

PLEASE READ THIS LICENSE AGREEMENT ("AGREEMENT") CAREFULLY. THIS AGREEMENT IS A BINDING LEGAL AGREEMENT ENTERED INTO BY AND BETWEEN YOU (OR IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF AN ENTITY, THEN THE ENTITY THAT YOU REPRESENT) AND QUALCOMM TECHNOLOGIES, INC. ("QTI" "WE" "OUR" OR "US"). THIS IS THE AGREEMENT THAT APPLIES TO YOUR USE OF THE DESIGNATED AND/OR ATTACHED DOCUMENTATION AND ANY UPDATES OR IMPROVEMENTS THEREOF (COLLECTIVELY, "MATERIALS"). BY USING OR COMPLETING THE INSTALLATION OF THE MATERIALS, YOU ARE ACCEPTING THIS AGREEMENT AND YOU AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THESE TERMS, QTI IS UNWILLING TO AND DOES NOT LICENSE THE MATERIALS TO YOU. IF YOU DO NOT AGREE TO THESE TERMS YOU MUST DISCONTINUE AND YOU MAY NOT USE THE MATERIALS OR RETAIN ANY COPIES OF THE MATERIALS. ANY USE OR POSSESSION OF THE MATERIALS BY YOU IS SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.

1.1 **License.** Subject to the terms and conditions of this Agreement, including, without limitation, the restrictions, conditions, limitations and exclusions set forth in this Agreement, Qualcomm Technologies, Inc. ("QTI") hereby grants to you a nonexclusive, limited license under QTI's copyrights to use the attached Materials; and to reproduce and redistribute a reasonable number of copies of the Materials. You may not use Qualcomm Technologies or its affiliates or subsidiaries name, logo or trademarks; and copyright, trademark, patent and any other notices that appear on the Materials may not be removed or obscured. QTI shall be free to use suggestions, feedback or other information received from You, without obligation of any kind to You. QTI may immediately terminate this Agreement upon your breach. Upon termination of this Agreement, Sections 1.2-4 shall survive.

1.2 **Indemnification.** You agree to indemnify and hold harmless QTI and its officers, directors, employees and successors and assigns against any and all third party claims, demands, causes of action, losses, liabilities, damages, costs and expenses, incurred by QTI (including but not limited to costs of defense, investigation and reasonable attorney's fees) arising out of, resulting from or related to: (i) any breach of this Agreement by You; and (ii) your acts, omissions, products and services. If requested by QTI, You agree to defend QTI in connection with any third party claims, demands, or causes of action resulting from, arising out of or in connection with any of the foregoing.

1.3 **Ownership.** QTI (or its licensors) shall retain title and all ownership rights in and to the Materials and all copies thereof, and nothing herein shall be deemed to grant any right to You under any of QTI's or its affiliates' patents. You shall not subject the Materials to any third party license terms (e.g., open source license terms). You shall not use the Materials for the purpose of identifying or providing evidence to support any potential patent infringement claim against QTI, its affiliates, or any of QTI's or QTI's affiliates' suppliers and/or direct or indirect customers. QTI hereby reserves all rights not expressly granted herein.

1.4 **WARRANTY DISCLAIMER.** YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT THE USE OF THE MATERIALS IS AT YOUR SOLE RISK. THE MATERIALS AND TECHNICAL SUPPORT, IF ANY, ARE PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED. QTI ITS LICENSORS AND AFFILIATES MAKE NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE MATERIALS OR ANY OTHER INFORMATION OR DOCUMENTATION PROVIDED UNDER THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR AGAINST INFRINGEMENT, OR ANY EXPRESS OR IMPLIED WARRANTY ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. NOTHING CONTAINED IN THIS AGREEMENT SHALL BE CONSTRUED AS (I) A WARRANTY OR REPRESENTATION BY QTI, ITS LICENSORS OR AFFILIATES AS TO THE VALIDITY OR SCOPE OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT OR (II) A WARRANTY OR REPRESENTATION BY QTI THAT ANY MANUFACTURE OR USE WILL BE FREE FROM INFRINGEMENT OF PATENTS, COPYRIGHTS OR OTHER INTELLECTUAL PROPERTY RIGHTS OF OTHERS, AND IT SHALL BE THE SOLE RESPONSIBILITY OF YOU TO MAKE SUCH DETERMINATION AS IS NECESSARY WITH RESPECT TO THE ACQUISITION OF LICENSES UNDER PATENTS AND OTHER INTELLECTUAL PROPERTY OF THIRD PARTIES.

1.5 **LIMITATION OF LIABILITY.** IN NO EVENT SHALL QTI, QTI'S AFFILIATES OR ITS LICENSORS BE LIABLE TO YOU FOR ANY INCIDENTAL, CONSEQUENTIAL OR SPECIAL DAMAGES, INCLUDING BUT NOT LIMITED TO ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE, OR THE DELIVERY OR FAILURE TO DELIVER, ANY OF THE MATERIALS, OR ANY BREACH OF ANY OBLIGATION UNDER THIS AGREEMENT, EVEN IF QTI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING LIMITATION OF LIABILITY SHALL REMAIN IN FULL FORCE AND EFFECT REGARDLESS OF WHETHER YOUR REMEDIES HEREUNDER ARE DETERMINED TO HAVE FAILED OF THEIR ESSENTIAL PURPOSE. THE ENTIRE LIABILITY OF QTI, QTI's AFFILIATES AND ITS LICENSORS, AND THE SOLE AND EXCLUSIVE REMEDY OF YOU, FOR ANY CLAIM OR CAUSE OF ACTION ARISING HEREUNDER (WHETHER IN CONTRACT, TORT, OR OTHERWISE) SHALL NOT EXCEED US$10.

2. **COMPLIANCE WITH LAWS; APPLICABLE LAW.** You agree to comply with all applicable local, international and national laws and regulations and with U.S. Export Administration Regulations, as they apply to the subject matter of this Agreement. This Agreement is governed by the laws of the State of California, excluding California's choice of law rules.

3. **CONTRACTING PARTIES.** If the Materials are downloaded on any computer owned by a corporation or other legal entity, then this Agreement is formed by and between QTI and such entity. The individual accepting the terms of this Agreement represents and warrants to QTI that they have the authority to bind such entity to the terms and conditions of this Agreement.

4. **MISCELLANEOUS PROVISIONS.** This Agreement, together with all exhibits attached hereto, which are incorporated herein by this reference, constitutes the entire agreement between QTI and You and supersedes all prior negotiations, representations and agreements between the parties with respect to the subject matter hereof. No addition or modification of this Agreement shall be effective unless made in writing and signed by the respective representatives of QTI and You. The restrictions, limitations, exclusions and conditions set forth in this Agreement shall apply even if QTI or any of its affiliates becomes aware of or fails to act in a manner to address any violation or failure to comply therewith. You hereby acknowledge and agree that the restrictions, limitations, conditions and exclusions imposed in this Agreement on the rights granted in this Agreement are not a derogation of the benefits of such rights. You further acknowledges that, in the absence of such restrictions, limitations, conditions and exclusions, QTI would not have entered into this Agreement with You. Each party shall be responsible for and shall bear its own expenses in connection with this Agreement. If any of the provisions of this Agreement are determined to be invalid, illegal, or otherwise unenforceable, the remaining provisions shall remain in full force and effect. This Agreement is entered into solely in the English language, and if for any reason any other language version is prepared by any party, it shall be solely for convenience and the English version shall govern and control all aspects. If You are located in the province of Quebec, Canada, the following applies: The Parties hereby confirm they have requested this Agreement and all related documents be prepared in English.