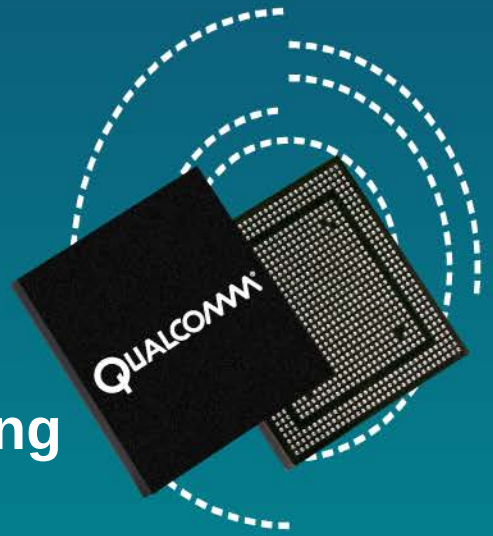
Qualcomm
TECHNOLOGIES, INC



Linux Android™ Software Thermal Debugging Guide

80-NM998-1 B

Confidential and Proprietary – Qualcomm Technologies, Inc.

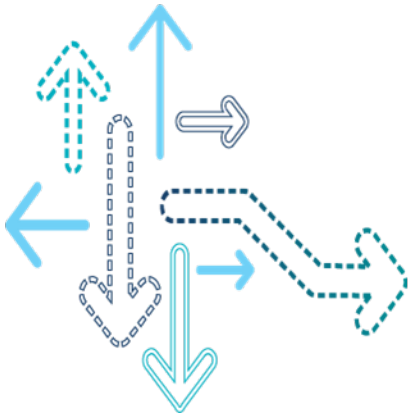
Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	Mar 2014	Initial release
B	Aug 2014	Added BTM/KTM case studies and description of KTM's full functionality

Contents

Introduction



Document Scope

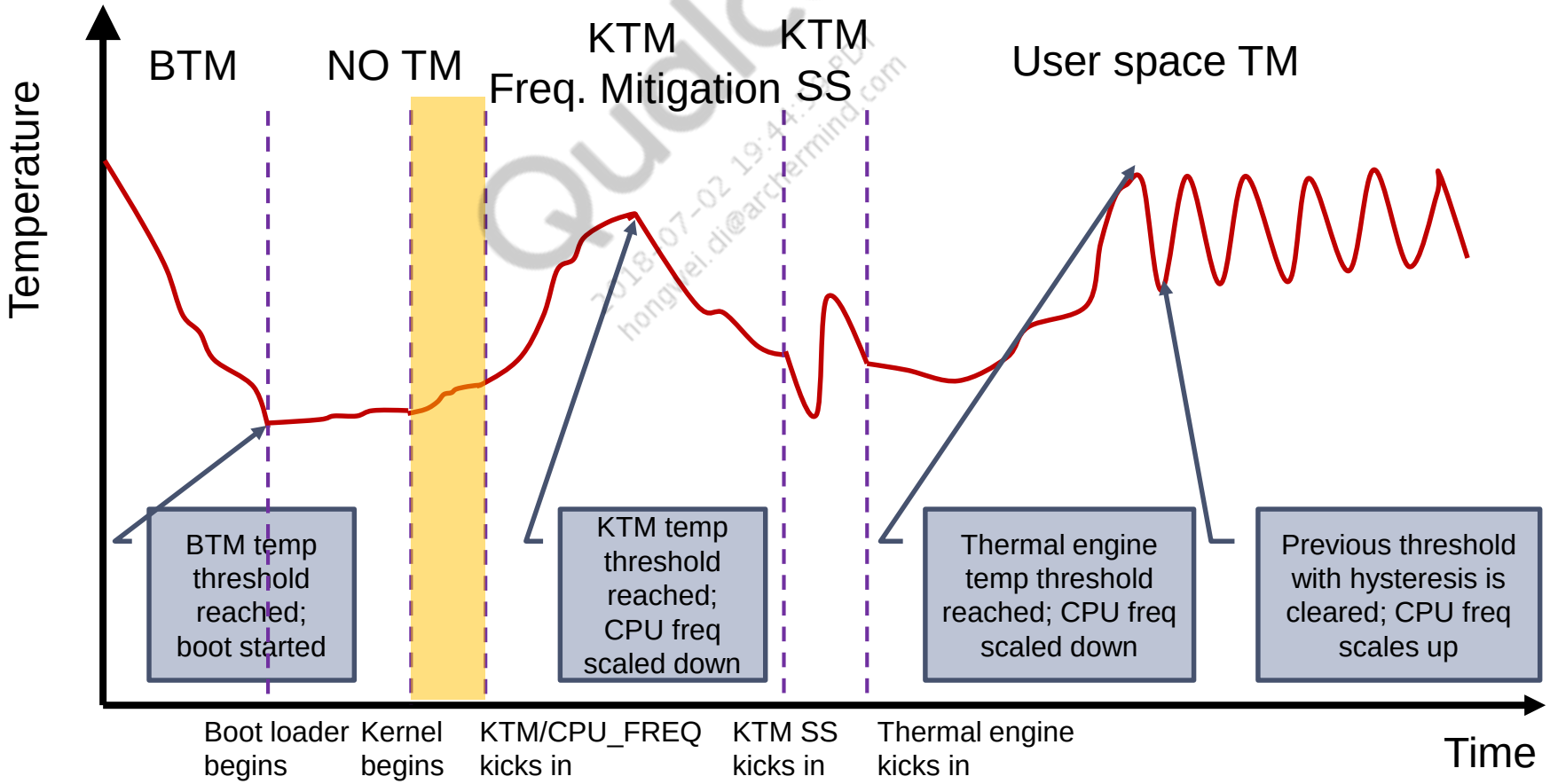
- This document provides guidance for debugging some of the most common thermal-related issues from the software perspective. In order to provide sufficient context for the discussion on thermal debugging, the document also includes information on the three different thermal instantiations, Boot Thermal Monitor (BTM), Kernel Thermal Monitor (KTM), and Thermal Engine.

Android™ Thermal Limits and Features

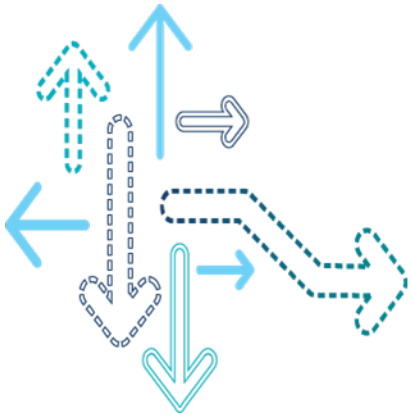
- Thermal limits protect the system
  - CPU/GPU mitigation threshold – Prevents violation of junction-level thermal limits in order to guarantee reliable operation; see the Thermal Engine config file
  - CPU hotplug on core 1/2/3 – Defined in device tree (MSM-thermal) to prevent tsens_reset
  - TSENS_RESET (hardware triggered) – Hardcoded in SBL1 to protect the device from damage
- BTM
  - Protects system from unexpected behaviors during boot loading such as memory corruption
  - Debug – Similar to stability debugging when any type of crash happens while booting from high/low temperature ambient environment
- KTM
  - Protects system during kernel boot time
  - Sets a single 110ºC threshold for emergency CPU mitigation and CPU hotplug
  - Hands control over to Thermal Engine
  - Debug – See kernel log and various other temperature information
- Thermal Engine
  - Full-fledged thermal protection
  - Must be tuned for specific OEM targets according to [Q7]
  - Debug – Review Thermal Engine logcat log and temperature/frequency/hotplug information
- Thermal Reset
  - Happens unexpectedly
  - Debug – Similar to stability debugging; identify the root cause, i.e., understand why KTM/Thermal Engine is not responding correctly or in time

Thermal Management (TM) Handover Timeline

- Three distinct TMs are provided
- No TM for SBLs and first few seconds after kernel started
- Graceful handover between KTM and Thermal Engine



BTM



BTM Configurations

- The device needs to be tested above the temperature defined as critical low (.nCriticalMin) and below the temperature defined as critical high (.nCriticalMax), otherwise the device will reset. In reality, consider the tsens accuracy of ±1.5ºC and other associated inaccuracies.
- If .nLowerThresholdDegC is defined (by default, -150ºC, and virtually disabled), the boot can be deferred or failed if the predefined number of retries are exceeded.

```
const BootTempCheckBspType BootTempCheckBsp[] = {
  {
    /* .nUpperThresholdDegC */ 150,
    /* .nLowerThresholdDegC */ -150
  }
};
```

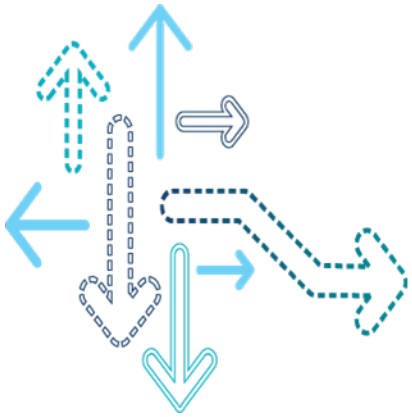
on_boot_images\core\hwengines\tsens\config\8x26\TsensBootBsp.c by Tsens_Init() function call. (see below)

```
const static TsensBootSensorType aSensors[] =
{
  /* Sensor 0 */
  {
    /* .uTsensConfig  */ 0x1C3,
    /* .eCal          */ TSENS_BSP_SENSOR_CAL_NORMAL,
    /* .nX1  default  */ 595,
    /* .nM  default   */ 11295,  /* TSENS_FACTOR * (1 / 2.901) */
    /* .nCriticalMin  */ -35,
    /* .nCriticalMax  */ 120
  },

  /* Sensor 1 */
  {
    /* .uTsensConfig  */ 0x11C3,
    /* .eCal          */ TSENS_BSP_SENSOR_CAL_NORMAL,
    /* .nX1  default  */ 629,
    /* .nM  default   */ 11502,  /* TSENS_FACTOR * (1 / 2.849) */
    /* .nCriticalMin  */ -35,
    /* .nCriticalMax  */ 120
  },
}
```

BTM – Hot/Cold Temperature Boot Failures

- Case 1 – Back-to-back boot test
  - Repeat {boot the device and reset once the UI is up and running}
  - Temperature gradually rises due to high power consumption during boot and eventually the temperature gets too high and crashes the device
  - To avoid the crash
    1. Reduce the power consumption on boot (especially kernel initialization) – Limit the maximum frequency of the CPUs or the number of active cores.
    2. Limit the BTM upper threshold to cool down the temperature when the initial temperature is too high.
- Case 2 – Device boots in extremely low temperature
  - Try to boot the device at -40ºC, and a reset happens
  - .nCriticalMin is defined as -35ºC and this is the allowed minimum temperature, so the test needs to be performed within operating temperature ranges, considering both tsens accuracy and thermal chamber accuracy. Consequently, ~30°C seems to be the lowest possible test point.

KTM



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 perioddefined 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 below 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

KTM Functions and Associated Debugging (cont.)

- Continuously checks current temperature from specified tsens and performs the following actions (cont.)
  - do_vdd_mx() – For some LE targets, KTM monitors all of the temperature sensors and if the temperature falls below a certain threshold, it votes for an increased MX rail voltage. KTM prints the below message when it receives the MX threshold notification.

    msm_thermal:vdd_mx_notify: Sensorα trigger received for type <threshold_type>
  - do_psm() – For PMIC automode disablement when a threshold is crossed over and KTM sends a command to make PMIC operate in Pulse Width Modulation (PWM) mode. KTM prints the below messages when it send 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 CX/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>: CX or GFX with multiple BAND definition depending on chipset

KTM Functions and Associated Debugging (cont.)

- Continuously checks current temperature from specified tsens and performs the following actions (cont.)
  - do_ocr() – For some targets, KTM monitors the temperature sensors and 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, e.g., MSM8974Pro, for deciding the number of phases on a power rail. In other targets, e.g., 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 below message when it votes for a 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 Vdd restriction
    msm_thermal:vdd_restriction_notify: sensor:α reached low thresh for Vdd restriction
    - KTM exposes a sysfs node, 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

KTM Configuration

- Verify various parameter sets in qcom,msm-thermal (the following example, /arch/arm/boot/dts/msm8974.dtsi, is MSM8974-specific)

```
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 = <110>;
                        qcom,hotplug-temp-hysteresis = <20>;
                        qcom,cpu-sensors = "tsens_tz_sensor5", "tsens_tz_sensor6",
                                           "tsens_tz_sensor7", "tsens_tz_sensor8";
                        qcom,vdd-restriction-temp = <5>;
                        qcom,vdd-restriction-temp-hysteresis = <10>;
                        qcom,pmic-sw-mode-temp = <85>;
                        qcom,pmic-sw-mode-temp-hysteresis = <75>;
                        qcom,pmic-sw-mode-regs = "vdd-dig";
                        vdd-dig-supply = <&pm8841_s2_floor_corner>;
                        vdd-gfx-supply = <&pm8841_s4_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-gfx-rstr{
                                    qcom,vdd-rstr-reg = "vdd-gfx";
                                    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 = <1881600 1958400 2265600>;
                                    qcom,freq-req;
                        };
            }
```

Verifying KTM Frequency Mitigation and KTM Handoff to Thermal Engine

- Kernel log will display actions taken

```
<6>[    0.743018] msm_thermal: Limiting cpu0 max frequency to 1674000
<6>[    0.743018] msm_thermal: Limiting cpu1 max frequency to 1674000
<6>[    0.743048] msm_thermal: Limiting cpu2 max frequency to 1674000
<6>[    0.743048] msm_thermal: Limiting cpu3 max frequency to 1674000
<6>[    0.992888] msm_thermal: Limiting cpu0 max frequency to 1458000
<6>[    0.992888] msm_thermal: Limiting cpu1 max frequency to 1458000
<6>[    0.992919] msm_thermal: Limiting cpu2 max frequency to 1458000
<6>[    0.992919] msm_thermal: Limiting cpu3 max frequency to 1458000
<6>[    1.243247] msm_thermal: Limiting cpu0 max frequency to 1242000
<6>[    1.243277] msm_thermal: Limiting cpu1 max frequency to 1242000
<6>[    1.243277] msm_thermal: Limiting cpu2 max frequency to 1242000
<6>[    1.243277] msm_thermal: Limiting cpu3 max frequency to 1242000
<6>[    1.493300] msm_thermal: Limiting cpu0 max frequency to 1026000
<6>[    1.493331] msm_thermal: Limiting cpu1 max frequency to 1026000
<6>[    1.493331] msm_thermal: Limiting cpu2 max frequency to 1026000
<6>[    1.493331] msm_thermal: Limiting cpu3 max frequency to 1026000
<6>[    1.743384] msm_thermal: Limiting cpu0 max frequency to 810000
<6>[    1.743415] msm_thermal: Limiting cpu1 max frequency to 810000
<6>[    1.743415] msm_thermal: Limiting cpu2 max frequency to 810000
<6>[    1.743415] msm_thermal: Limiting cpu3 max frequency to 810000
<6>[    1.993438] msm_thermal: Limiting cpu0 max frequency to 594000
<6>[    1.993468] msm_thermal: Limiting cpu1 max frequency to 594000
<6>[    1.993468] msm_thermal: Limiting cpu2 max frequency to 594000
<6>[    1.993468] msm_thermal: Limiting cpu3 max frequency to 594000
<6>[    2.243491] msm_thermal: Limiting cpu0 max frequency to 384000
<6>[    2.243491] msm_thermal: Limiting cpu1 max frequency to 384000
<6>[    2.243491] msm_thermal: Limiting cpu2 max frequency to 384000
<6>[    2.243522] msm_thermal: Limiting cpu3 max frequency to 384000
<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
```


VDD Restriction – Cold Temperature-Related Stability Issues

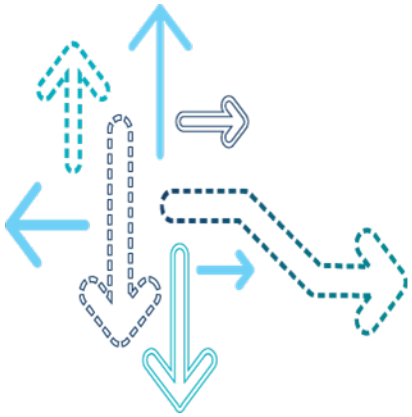
- KTM or Thermal Engine monitors all tsens for temperatures that dip below 5ºC, at which point the VDD restriction rule will be employed.
- VDD restriction ensures that proper voltage levels are maintained when ambient temperatures are low.
  - Minimum voltage levels on predefined power rails – Nominal on CX and GFX for the below example
  - Minimum, frequency level on apps core – 1881600 Hz for the given example
- If any stability concern is raised on low temperatures, e.g., <0ºC, make sure that VDD restriction has properly kicked in and is maintaining proper voltages and performance levels.

```
qcom,msm-thermal {
                    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-gfx-rstr{
                              qcom,vdd-rstr-reg = "vdd-gfx";
                              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 = <1881600 1958400 2265600>;
                              qcom,freq-req;
                    };
          }
```


VDD/MX Restriction – Cold Temperature-Related LPM Issue

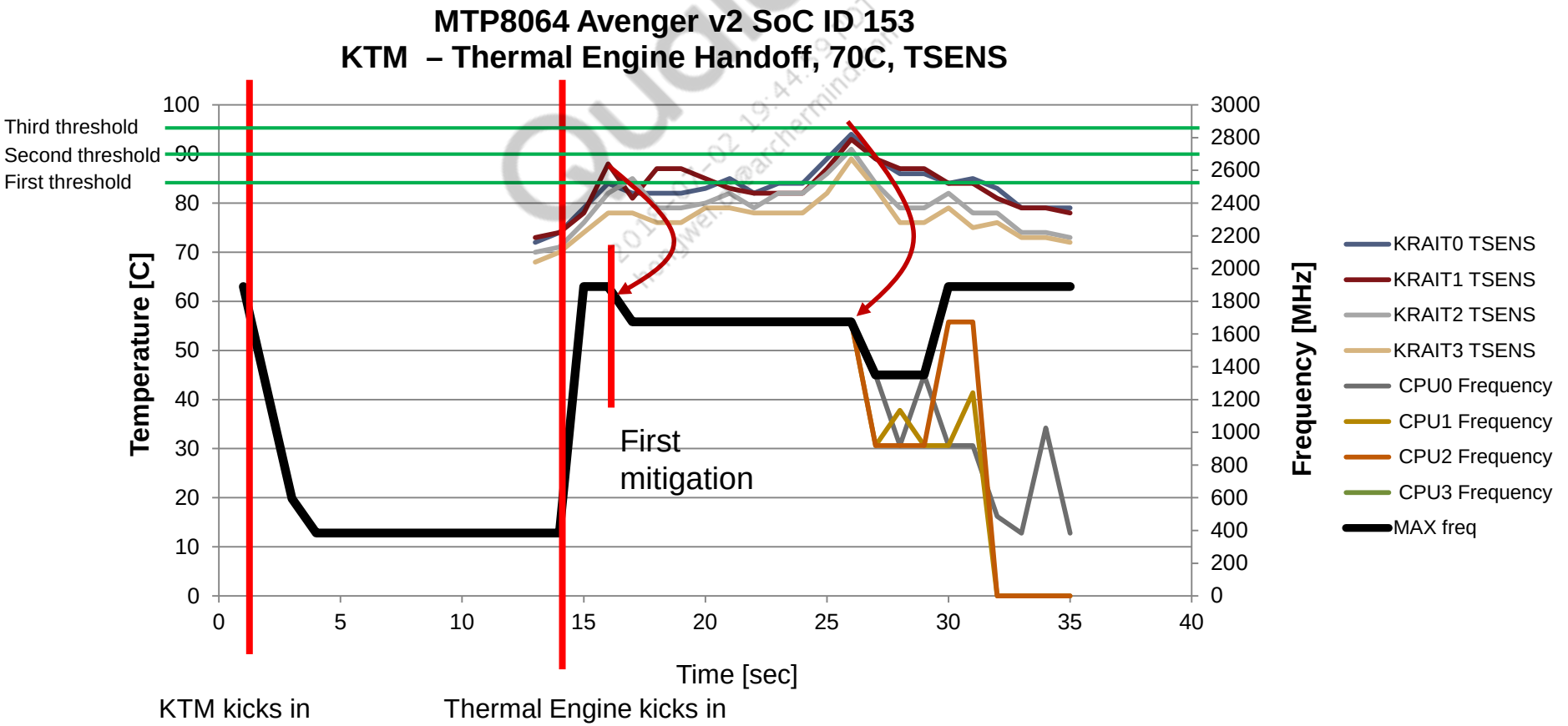
- KTM or Thermal Engine monitors all tsens for temperatures that dip below 5ºC, at which point the VDD restriction rule will be employed.
  - Active set CX voting, which will not impact VDD min voltage on CX
- For devices with 20nm or onward chipsets, MX restriction is also applied.
  - Required MX voting, which impacts VDD min voltage on MX
  - Actual VDD min happens when both CX and MX are requested in retention; otherwise, if MX is requested to be in SVS, CX cannot be in retention; instead, it will stay at the same level when it enters XO shutdown.

KTM/Thermal Engine



KTM/Thermal Engine Handoff Timeline Analysis

- The handover from KTM and Thermal Engine executed as expected
  - CPU1/2/3 temperature changes follow the similar trend as CPU0
  - Temperature rises above 85°C during the handover period, but temperature is well maintained
- This graph is based on sampled data (1 sec) and there may exist a slight accuracy and synchronization (between power and frequency/temperature) error



KTM/Thermal Engine Handoff

- Improvement on MSM8974 and newer chipsets
  - Issue – Thermal Engine kicks in and KTM is disabled by resetting CPU frequencies to max, which causes rapid temperature rises
  - Resolution – Allow overlap between these transitions to avoid a period where no thermal mitigation exists
  - Post boot KTM functionality runs in parallel for better thermal stability
  - Following code snippet is from /vendor/qcom/proprietary/thermal-engine/thermal.c

```
/* Vote to keep kernel mitigation enabled until init is done */
        kernel_dev = devices_manager_reg_clnt("kernel");
        if (kernel_dev == NULL) {
                msg("%s Failed to create kernel device handle\n", __func__);
        }
        req.value = 1;
        device_clnt_request(kernel_dev, &req);
….
        thermal_server_init();
        pid_algo_init(&thermal_settings);
        thermal_monitor(&thermal_settings);
        ss_algo_init(&thermal_settings);
        speaker_cal_init(&thermal_settings);

        if (kernel_dev)
                device_clnt_cancel_request(kernel_dev);
```

Thermal Engine Rules – Embedded vs Configuration File

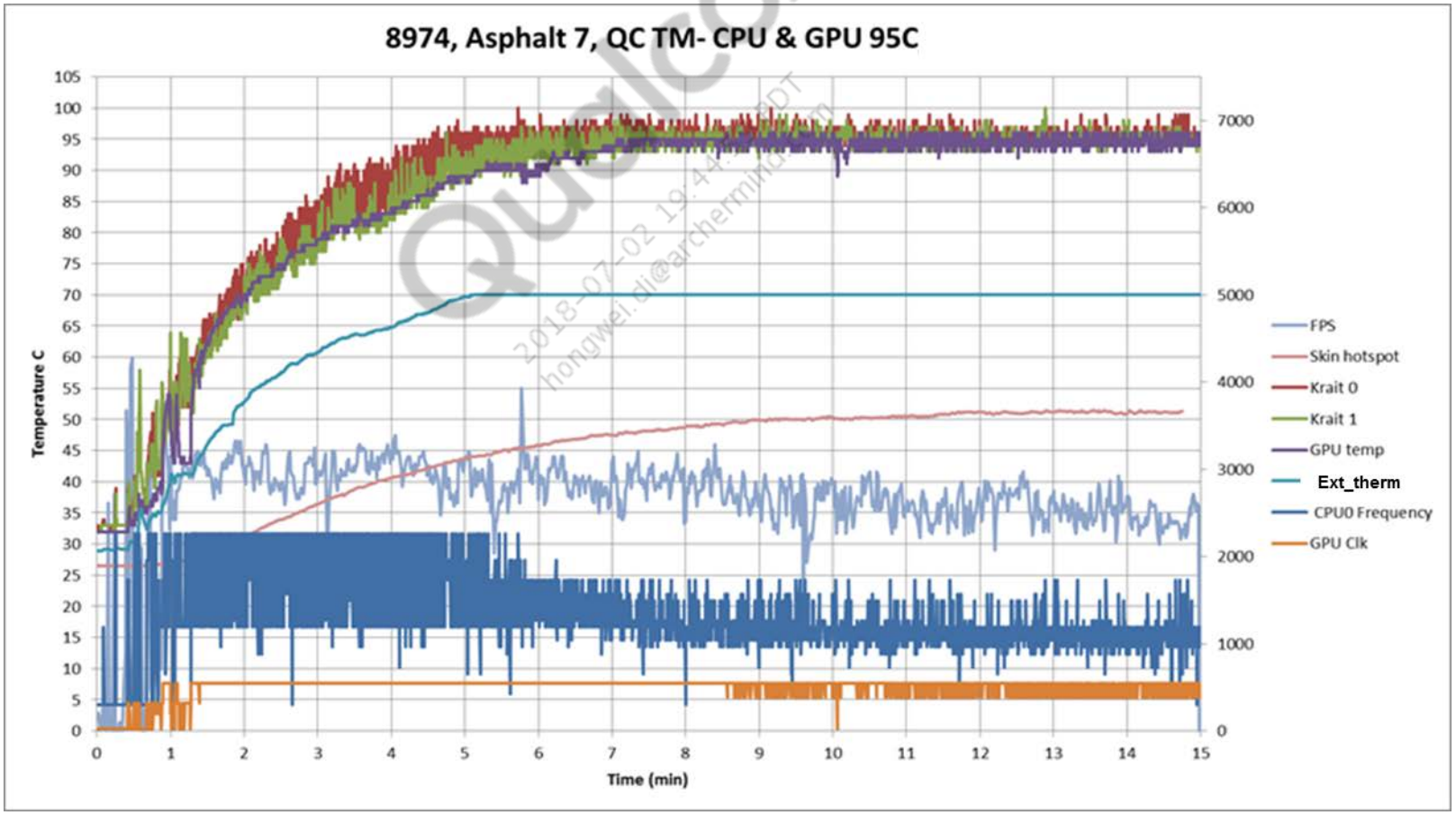
- Start three different algorithms
  - SS, PID, and monitor
  - All the rules are defined either in the Thermal Engine config file or hardcoded in Thermal Engine data files
    - Embedded (hardcoded) rules
      - Required
        - Voltage restriction
        - PSM control (PMIC auto-mode)
      - Optional (can be disabled or overridden granted junction limit is not violated)
        - SS/PID control on all the CPUs
    - OEM-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

```
thermal_server_init();
     pid_algo_init(&thermal_settings);
     thermal_monitor(&thermal_settings);
     ss_algo_init(&thermal_settings);
     speaker_cal_init(&thermal_settings);
```

- Sensor management
  - Tsens/PMIC ADC sensors are fully interrupt driven
  - Can add OEM's external thermistor either interrupt driven or polling based

Thermal Profiling and Logging

- To gain more insight on a thermal-related issue through various temperature and CPU/GPU frequency data



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. Put “debug” in the first line of thermal-engine.conf and restart thermal-engine service
  2. Stop thermal-engine and /sbin/thermal-engine –-debug &
- Logcat log and temperature logging
  - logcat –v time –s ThermalEngine

```
01-21 21:30:01.948 E/ThermalEngine(  286): Setting up TSENS thresholds low: 90
01-21 21:30:01.948 E/ThermalEngine(  286): j=0 i=0 TM Id HOTPLUG-CPU1 Sensor cpu1: Action hotplug_1 value 1
01-21 21:30:01.948 E/ThermalEngine(  286): devices_manager_set_lvl: DEV hotplug_1, lvl 1
01-21 21:30:01.948 I/ThermalEngine(  286): hotplug_ktm_request: write out 2
01-21 21:30:01.958 I/ThermalEngine(  286): ACTION: Hotplugged OFF CPU[1]
01-21 21:30:01.958 E/ThermalEngine(  286): TM Id HOTPLUG-CPU1 Sensor cpu1 Reading 105.0
01-21 21:30:01.958 E/ThermalEngine(  286): handle_thresh_sig: TM Id HOTPLUG-CPU1 Sensor cpu1 Temp 105000
01-21 21:30:01.958 E/ThermalEngine(  286): TM Id 'HOTPLUG-CPU1' Sensor 'cpu1' - alarm raised 1 at 105.0 degC
01-21 21:30:01.958 E/ThermalEngine(  286): sensor_update_thresholds: TM Id HOTPLUG-CPU1 Sensor cpu1 threshold_type 1, level 1
01-21 21:30:01.958 E/ThermalEngine(  286): sensors_manager_set_thresh_lvl: tsens_tz_sensor6 Hi(0) 105000, Lo(1) 90000, Interval(1) 65
01-21 21:30:01.958 E/ThermalEngine(  286): update_active_thresh: tsens_tz_sensor6 Active(1), Hi(0) 2147483647, Lo(1) 90000, Interval(1) 65
01-21 21:30:01.958 E/ThermalEngine(  286): enable_threshold: tsens_tz_sensor6 (/sys/devices/virtual/thermal/thermal_zone6/trip_point_0_type)
01-21 21:30:01.958 E/ThermalEngine(  286): TSENS threshold at 0 enabled: 0
01-21 21:30:01.958 E/ThermalEngine(  286): enable_threshold: tsens_tz_sensor6 (/sys/devices/virtual/thermal/thermal_zone6/trip_point_1_type)
01-21 21:30:01.958 E/ThermalEngine(  286): TSENS threshold at 1 enabled: 1
01-21 21:30:01.958 E/ThermalEngine(  286): Setting up TSENS thresholds low: 90
```

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 on
  - 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("\nRead TEMPZONE0 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
  - See [Q7] and [Q8]

Tsens Reset – Are We Really Seeing This?

- Tsens reset is a hardware-triggered reset that is 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
  - However, Thermal Engine cannot run properly under certain situations
    - Just 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. Then, a mid-priority process (long lasting one) gets scheduled and Thermal Engine had no chance to proceed. Consequently, temperature had risen continuously. The issue can be resolved when the low-priority process is boosted (through priority inheritance or priority ceiling), but more fundamentally, remove any possibility of user space mutex causing this issue.
- Fundamental solution
  - Emergency throttling in kernel space – Implemented and incorporated in main branch for MSM8974 and newer chipsets

Case Study – A Priority Inversion Leading Tsens Reset

- SS loop is blocked due to a user space mutex (identified by looking at crash RAM dump)
- Other algorithms are still running
  - Hotplug (monitor) on CPU1/2/3 still happens
  - However, CPU0's frequency still remains at 1728000 Hz and eventually CPU0's temperature exceeds 120ºC, which causes tsens reset

08-24 16:11:30.968 15241 15283 D ThermalEngine: ACTION: CPU - Setting CPU[0] to 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: CPU[0] frequency limited to 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: devices_manager_set_op_value: DEV cpu1, op_value 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: ACTION: CPU - Setting CPU[1] to 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: CPU[1] frequency limited to 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: devices_manager_set_op_value: DEV cpu2, op_value 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: ACTION: CPU - Setting CPU[2] to 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: CPU[2] frequency limited to 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: devices_manager_set_op_value: DEV cpu3, op_value 1728000
08-24 16:11:30.968 15241 15283 D ThermalEngine: ACTION: CPU - Setting CPU[3] to 1728000
-----Should see "ThermalEngine: CPU[3] frequency limited to 1728000" following this. ----
08-24 16:11:31.288 15241 15260 D ThermalEngine: tsens_uevent: tsens_tz_sensor6
08-24 16:11:31.288 15241 15261 D ThermalEngine: sensor_monitor: tsens_tz_sensor6 Reading 105000 .
08-24 16:11:31.288 15241 15261 D ThermalEngine: sensor_thresh_notify: Update recieved HOTPLUG-CPU1 105000
08-24 16:11:31.288 15241 15261 D ThermalEngine: update_active_thresh: tsens_tz_sensor6 Active(1), Hi(0) 2147483647, Lo(1) 75000, Interval(1) 1000

Kernel Emergency Throttling – No More tsens_reset

- No tsens_reset is expected, however, if it happens:
  - First, identify why Thermal Engine had not been properly mitigating CPUs
    - Go back to Thermal Engine debugging
  - Secondly, decouple Thermal Engine and KTM
    - Try to reproduce the issue by stopping Thermal Engine
      - If reproducible, debug KTM
      - Else the issue mostly happens due to interaction between Thermal Engine and KTM
        - Need to look at tsens threshold values and enable/disable status
        - IOCTL interface
    - Use hot chamber to expedite the reproduction of the issue

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
    - thermal_client_config_query
    - thermal_client_config_set
    - thermal_client_config_cleanup
- However, the calling processes should have proper privileges to request Thermal Engine to proceed.
  - Currently, only processes with root or system 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

References

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Thermal Design Checklist*	80-VU794-21
Q3	*Design For Thermal: Key Requirements Why What Where When*	80-VU794-24
Q5	*Presentation: Thermal Protection Algorithm Overview*	80-VT344-1
Q6	*MSM8974 Thermal Mitigation Algorithm*	80-N8633-6
Q7	*Presentation: Thermal Tuning Procedure*	80-N9649-1
Q8	*Application Note: Skin Temperature Measurement Procedure Using IR Camera*	80-VU794-15

Questions?

https://support.cdmatech.com