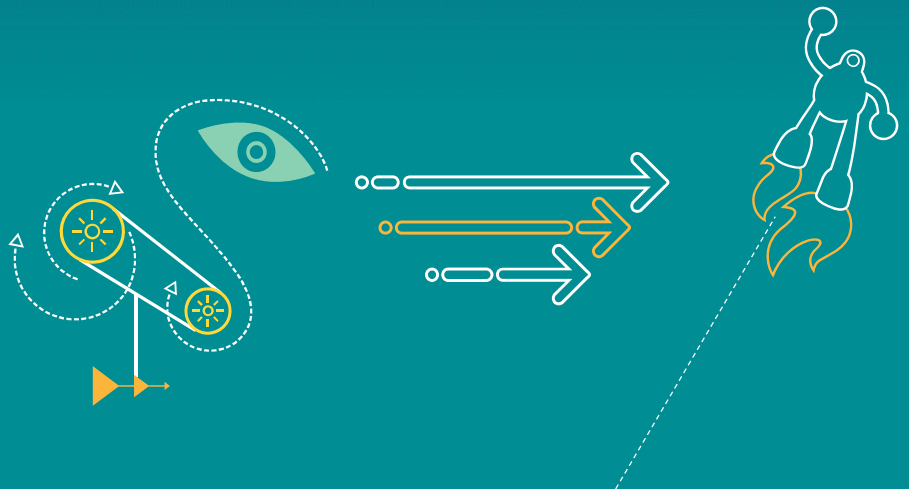
MSM8917, MSM8937, MSM8940 Linux Android Thermal Management Overview

Qualcomm Technologies, Inc.

80-P2485-13 Rev. F



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

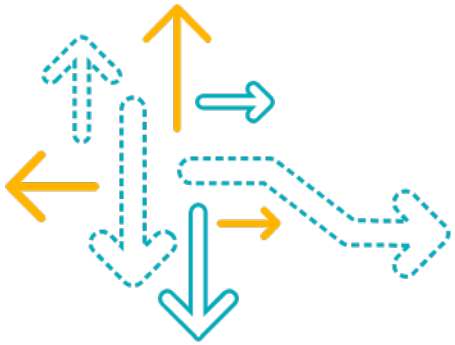
Revision History

Revision	Date	Description
A	November 2015	Initial release
B	February 2016	Numerous changes were made in this document; to be read in its entirety.
C	May 2016	Added slides 24, 25, and 26. Updated slides 9, 10, 11, and 29.
D	June 2016	Updated the document title; updated slides 9-11
E	August 2016	Added slides 23-26, 30, 31, 33-38, 44, 47, and 48. Updated slides 7, 10-13, 17-22, 32, and 45.
F	October 2017	Numerous changes were made in the section *Modem Thermal Management*; to be read in its entirety.

Contents

Objectives

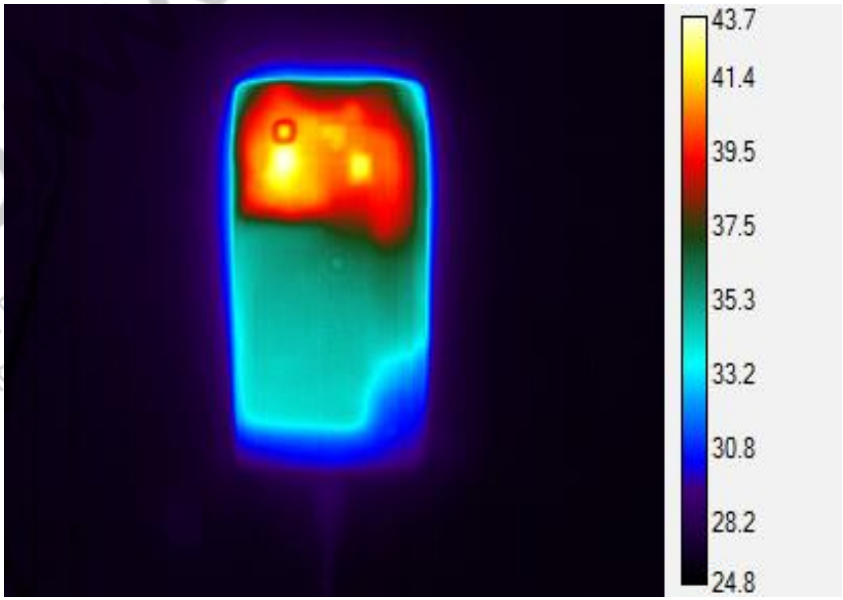
- At the end of this presentation, you will understand the Linux Android thermal management features for the MSM8917, MSM8937, MSM8940 chipset.



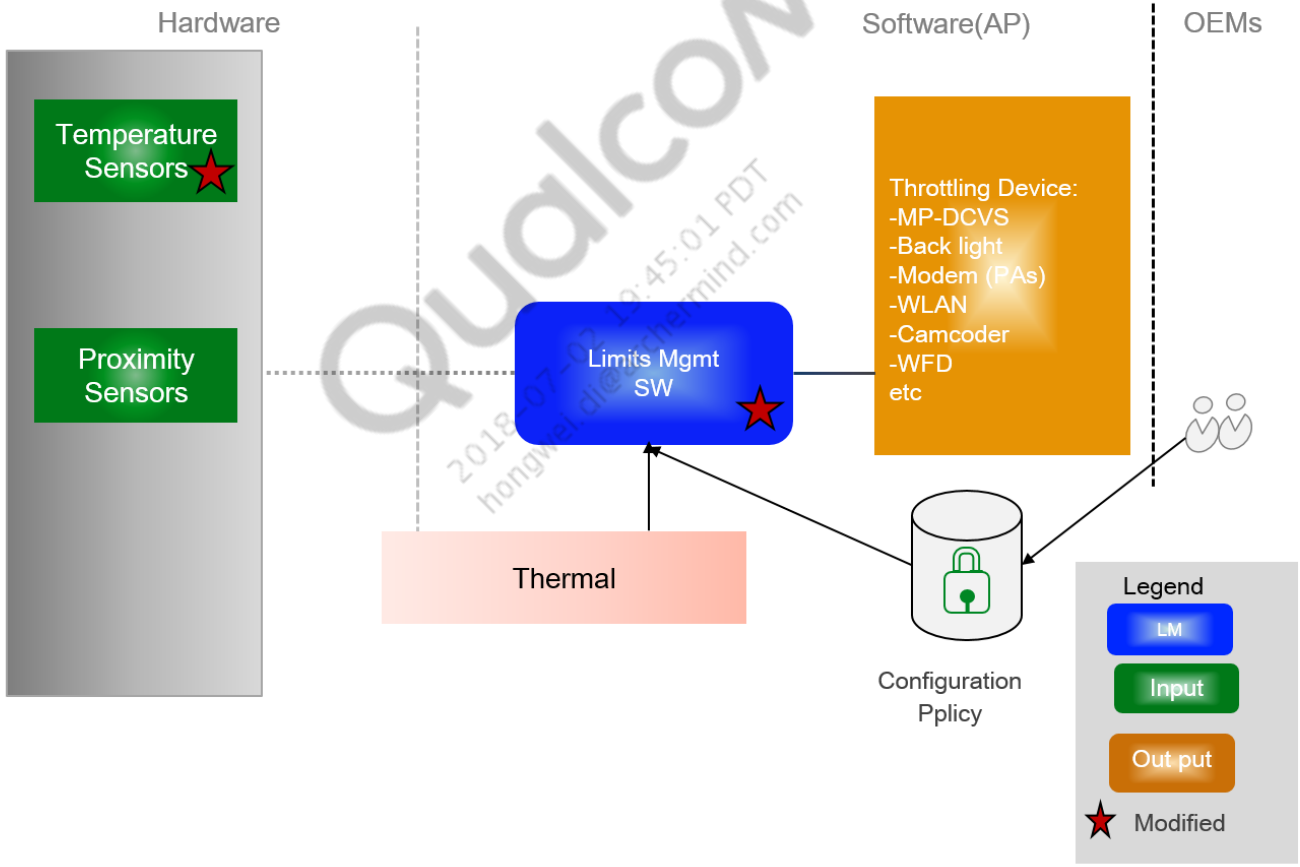
Thermal Management

Overview

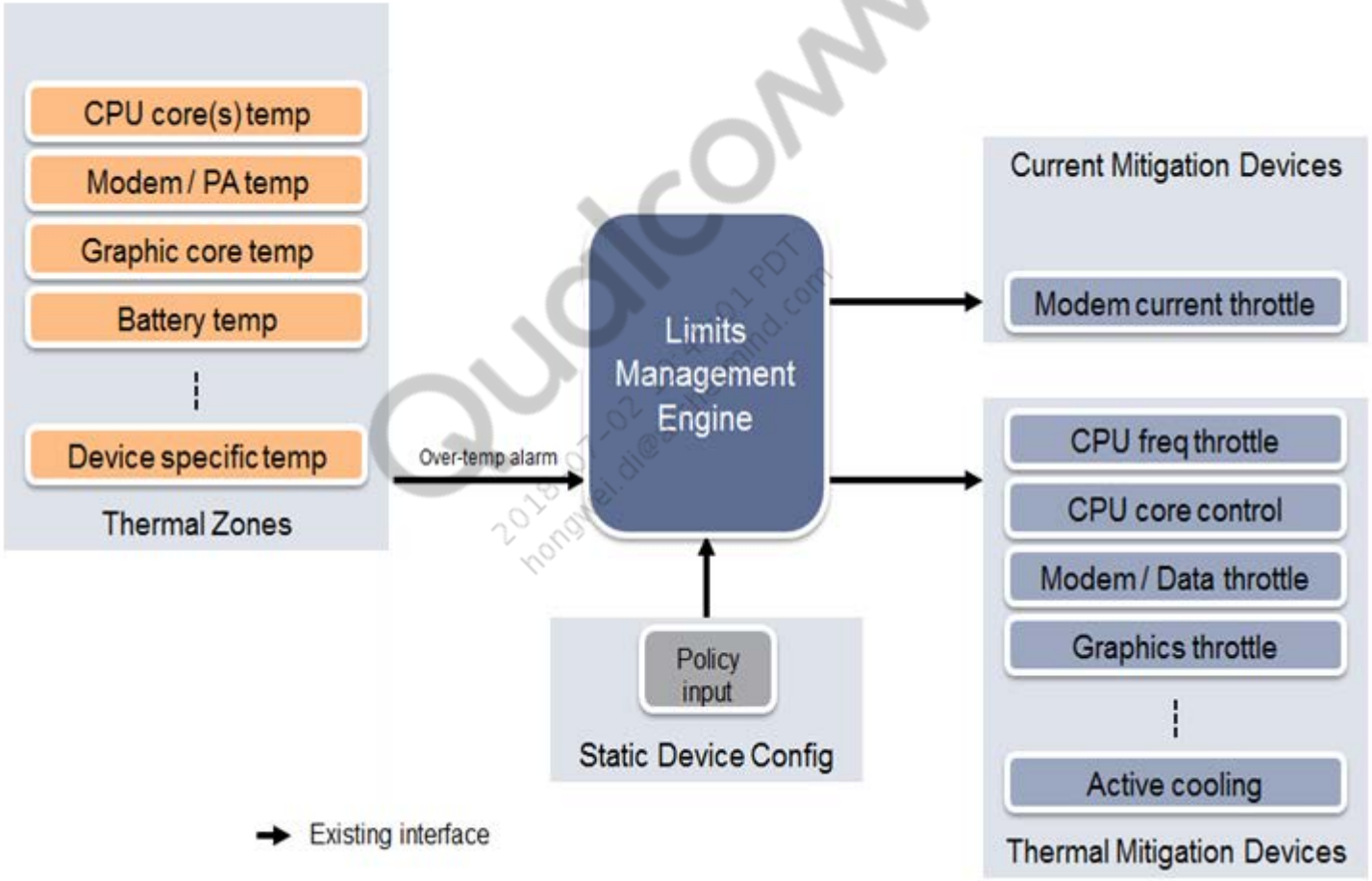
- The purpose of thermal management is to manage:
  - Silicon junction temperature limits; 85ºC for logic
  - External surface temperature limits, typically 45ºC for the handheld device
- Simulation of mechanical design is the most important step in achieving the best performance.
- Thermal management software controls thermal response.
- Temperature sensors
  - There are temperature sensors on the logic die
  - PCB sensors – Power management integrated circuit (PMIC), Power amplifier (PA), surface temperature estimation, and so on.
- Management devices
  - Passive cooling applied by reduced performance and power consumption
  - Active cooling – Fan
  - Set points are configurable by licensees to tune for industrial design capability



MSM Thermal Architecture



MSM LM Thermal Software Architecture



Deltas Between MSM8917, MSM8937, MSM8940 and MSM8952 Software Impacts

Area	Functionality	MSM8952	MSM8937, MSM8940	MSM8917	Thermal software impact	Comments
TSENS	Temperature sensors supported	11	11	10	No	–
	Sensor placements	A53 4 – Performance 1 – L2 MHM 1 – Power cluster 1 – GPU 1 – Camera 1 – Modem 1 – Qualcomm® Hexagon™ DSP 1 – Main sensor	A53 4 – Performance 1 – Power Cluster 1 – GPU 1 – Main sensor 1 – Camera 1 – Modem 1 – Hexagon 1 – L2 MHM	A53 4 – 1 sensor for each CPU core 1 – GPU 1 – Main sensor 1 – Camera 1 – Modem 1 – Hexagon 1 – L2 MHM	Yes (configuration mapping changes)	Device-specific sensor mapping changes in the software
Thermal management	Hotspot monitoring	Mitigation triggered by TSENS interrupt invokes throttling mitigation	Same as MSM8952	Same as MSM8952	No (only porting for cluster level)	Throttling mechanisms: CPU, GPU – DCVS Modem – UL flow control, Tx power limiting, WLAN data throttling, display backlight adjustment Battery charging rate Camcorder fps restriction
	Coldspot monitoring	Invokes voltage restriction	Same as MSM8952	Same as MSM8952	No (only porting effort)	–

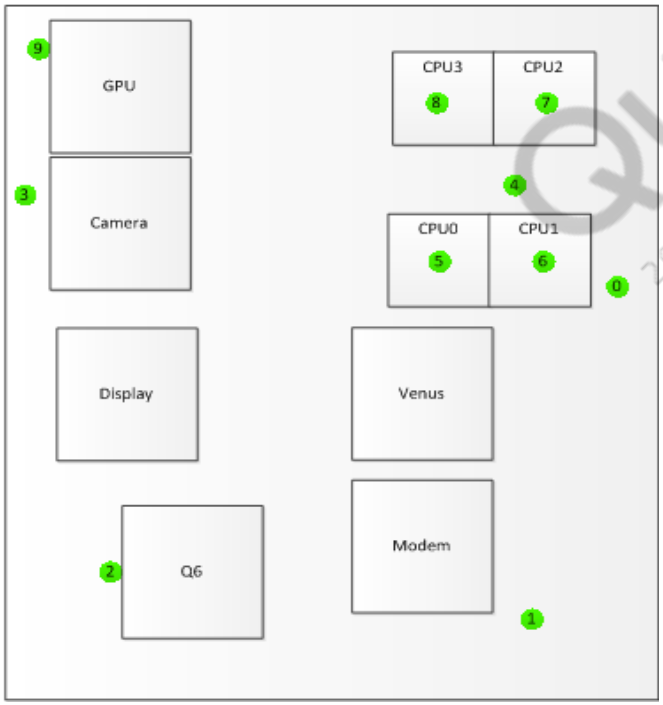
Deltas Between MSM8917, MSM8937, MSM8940 and MSM8952 Software Impacts (cont.)

Area	Functionality	MSM8952	MSM8937, MSM8940	MSM8917	Thermal software impact	Comments
Battery current limiting (BCL)	Triggered mitigation based on current load	PMIC supports new BCL peripheral to monitor ibat and Vbatt and to trigger an interrupt when threshold is reached	Leverage from MSM8952, MSM8976	Leverage from MSM8952, MSM8976	Some changes from PMIC side	BCL kernel module monitors battery discharge current and voltage through BCL peripheral and applies CPU mitigation when required. BCL implementation uses PM89xx fuel gauge hardware.
MTC	Multizone temperature control	MTC issues commands to pulse swallower based on sudden transient	Same as MSM8952	Same as MSM8952	No	MTC configuration is taken care of by the TSESE driver based on profiled PTE data
Modem	UL throttling	Supported	Supported	Supported	No	–
	Tx power backoff and/or PUCCH backoff	Supported	Supported	Supported	No	–
	Hexagon Silver core thermal mitigation (LTE and WCDMA)	MSM8952 is with TA modem, which introduced with Hexagon Silver core mitigation	MSM8937 is not supported, since it is with MPSS.JO (Jolokia) 1.2 modem. MSM8940 is supported, it is based on TA	Not Supported, since it is based Jolokia Cat4 modem	Leveraged source code from MSM8952 for supporting Cat6 TA modem on MSM8940	MSM8940 is based on TA – MPSS.TA (Tabasco) modem

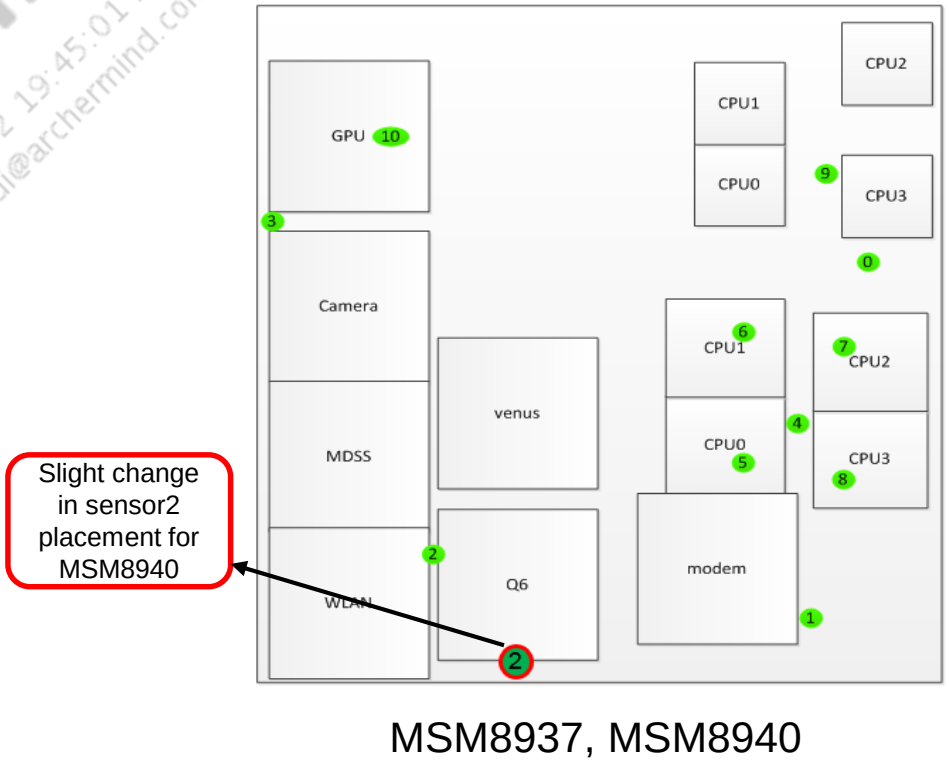
On-Die Temperature Sensors

- MSM8917 chipset has 10 and MSM8937, MSM8940 chipset has 11 on-die sensors. More sensors (thermistors) are needed on devices (PA, XO, BATT, PMIC, and case thermistors) to tune thermal configuration.
- Hardware tsens reset occurs to protect the device when predefined critical high or low threshold in SBL1 exceeds
- boot_images\core\hwengines\tsens\config\<Target>\TsensBootBsp.c

```
/* .nCriticalMin   */ -35,
/*.nCriticalMax   */ 120
```



MSM8917



MSM8937, MSM8940

Note: Sensors are represented by green dots with the sensor ID

Software Thermal Management in Linux Android

- Manages thermal start-up and normal operation
  - SBL temperature checks to confirm whether the initial temperature is within operational limits.
  - Kernel thermal monitor (KTM) keeps the die temperatures within limits under all ambient conditions when the full thermal engine is initialized.
  - Full thermal engine monitors temperature limits across the whole system

	Linux Android boot timeline					
Initialization phase	RPM PBL	CPU PBL	CPU SBL	Application boot loader	Kernel initialization	User space initialization
Thermal management	–	–	Check CPU0 temperature delay if outside of limits	–	Monitor CPU temperature reduces frequency of all cores above set point. Power collapse cores if above second set point.	Thermal engine
CPU configuration	Power collapsed	1 core nominal voltage 960 MHz	1 core nominal voltage 960 MHz	1 core nominal voltage 960 MHz	Initially two or more cores nominal voltage 960 MHz. Transition to multicore with DCVS	Multicore with DCVS

Boot Thermal Management (BTM) Algorithm

KTM Algorithm

Thermal Engine

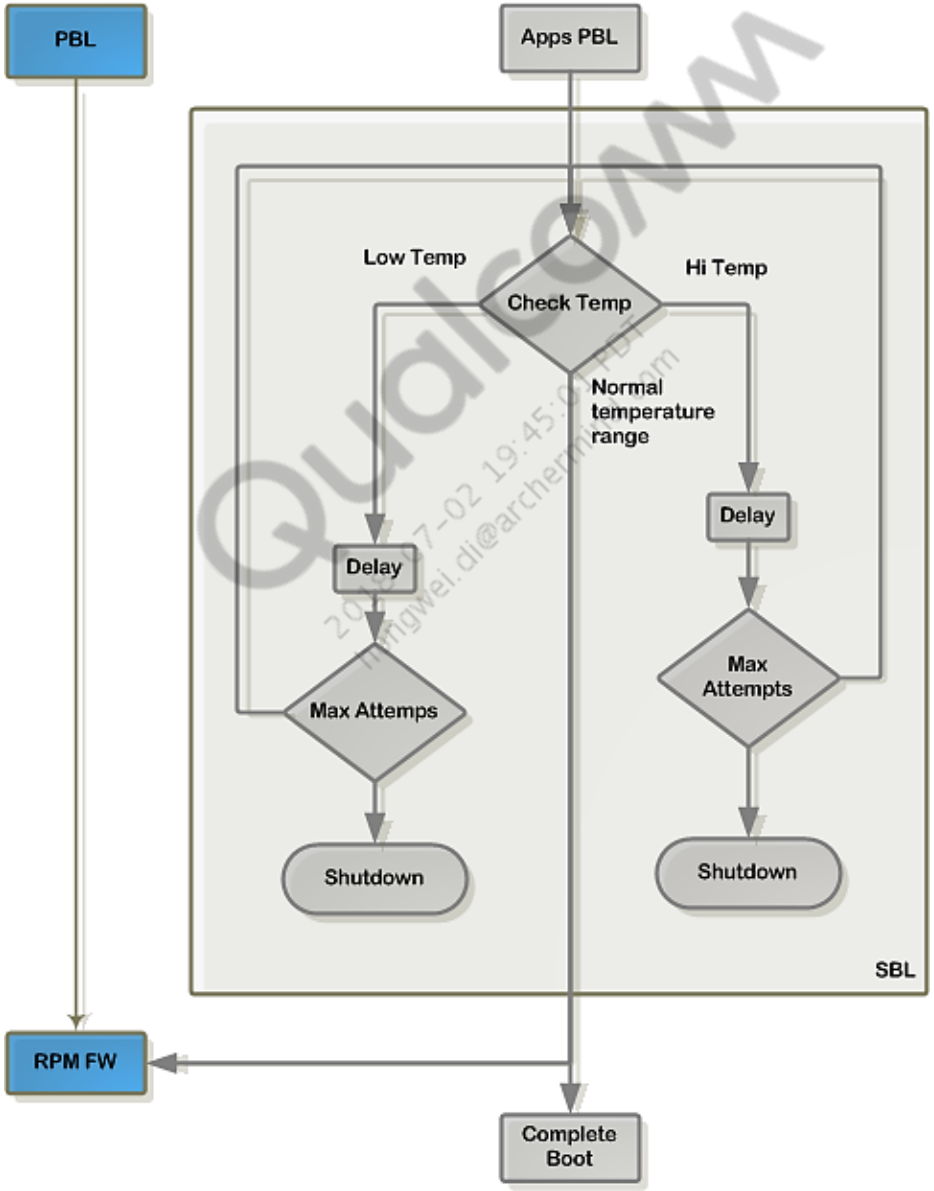
BTM Algorithm

- Early boot mitigation
  - Ensures that temperature is in a valid operating range before allowing the device to boot.
  - The high temperature bound is set to ensure that the device is operational until the next available mitigation algorithm (kernel thermal driver) can monitor.
  - The low temperature bound is set to ensure that the device maintains timing closure at the operational voltages set in the boot code.
  - Temperature thresholds, delays, and maximum attempts are configured in the boot loader build
- BTM configuration hard coded
- nUpperThresholdDegC or LowerThresholdDegC is set to max value + or -150°C by default and BTM is virtually disabled.
- If nUpperThresholdDegC or LowerThresholdDegC is defined by OEMs with correct temperature threshold, boot can be deferred if the current temperature is higher than the threshold and retries; however, boot can fail if predefined number of retries is exceeded.

  On boot_images/core/hwengines/tsens/config/89xx/BootTempCheckBsp.c

```
Const BootTempCheckBspTypeBootTempChecksBsp[]={
  {
    /* .nUpperThesholdDegC*/150,
    /* .nLowerThresholdDegC*/-150
   }
  };
```

BTM Algorithm (cont.)



KTM

- Kernel bootup mitigation
  - KTM is one of the kernel drivers that is initialized early to guarantee correct operation and protect the device from thermal damage.
  - After the driver is initialized, it starts polling for the temperature and mitigating the CPU (or other devices) when the temperature is above a certain threshold.
    - Mitigations or monitoring include:
      - CPU frequency mitigation
      - CPU core control
      - Thermal reset
      - Vdd restriction – Cx, MSS, and APC
  - KTM eventually switches to the Interrupt mode late in the initialization phase. Before performing this transition to Interrupt mode, the previous mitigations posed during bootup are removed.

KTM (cont.)

- Postbootup mitigation
  - After the system enters the late initialization phase, the KTM clears the current mitigation or monitor thread and switches to an Interrupt mode type mitigation. During this phase, the thermal-sys framework is initialized and thermal can use this framework to set the threshold and receive notifications. This switch to Interrupt mode does not depend on or wait for the thermal engine.
    - Mitigations or monitoring include:
      - Emergency frequency mitigation
      - Emergency hotplug
      - Thermal reset
      - Vdd restriction – CX, Mss, and APC
  - These KTM features permanently coexist with the thermal engine for better device protection.

KTM Configuration (Kernel Device Tree Example)

- The algorithm is present in the /drivers/thermal/msm_thermal.c file and the associated parameters are defined in the arch/arm/boot/dts/<chipset>.dtsi file.

  KTM device tree example:

```
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>;
                        …
    };
```

User Space Thermal Engine

- Thermal daemon was initially commercialized on the MSM8660 chipset and later enhanced and reworked.
- The reworked thermal daemon enables integration with sensor manager and allows for multiple algorithms.
  - Thermal engine: The thermal engine supports legacy and advanced dynamic algorithms running in parallel. OEMs will be able to choose the existing algorithm or the new algorithm in the thermal engine configuration file.
  - Dynamic algorithm exhibits significantly reduced tuning effort and improved average DMIPS.
  - Virtual sensor: A combination of more than two sensor readings with associated weights to accurately correlate skin temperature.
  - Dynamic parameter update: Important set of thermal engine configuration parameters can be updated at runtime for better OEM-specific dynamic thermal management.
- Two algorithm types are used in the thermal configuration file:
  - Monitor or threshold algorithm – algo_type monitor
  - Dynamic thermal management (DTM) – algo_type_ss

Threshold or Monitor Algorithm

- Threshold or Monitor algorithm performs mitigation based on preconfigured set points. When a temperature threshold is reached, a management device is set to a predetermined performance level.
- OEM determines a series of temperature thresholds and a precise corresponding action per threshold.
- Monitor algorithm is used for LCD, modem, camcorder, and battery mitigation; not recommended for CPU and GPU mitigation, as it requires extensive tuning to find each set point.

Threshold or Monitor configuration example:

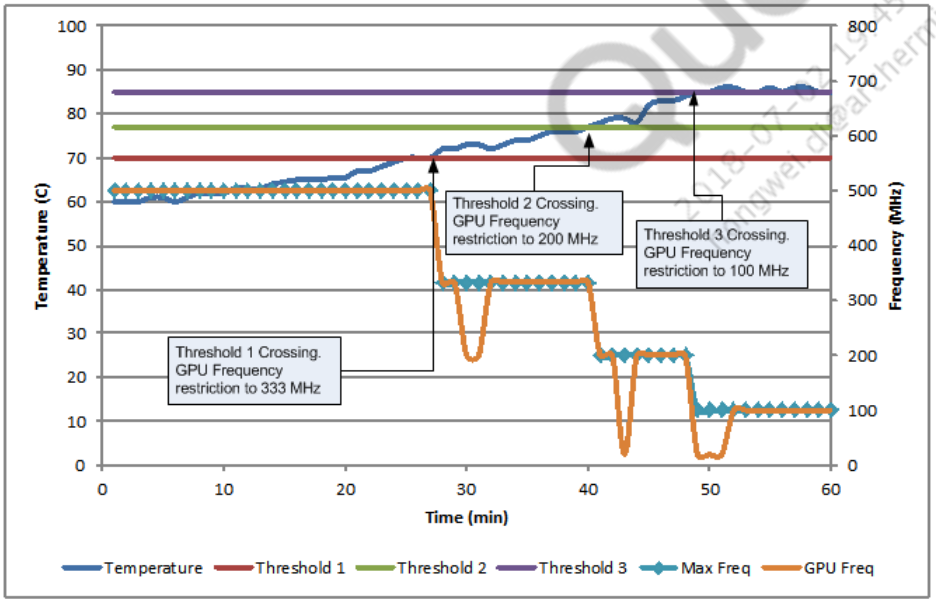
```
sampling         1000     : Default sampling period of 1000msec

[MSS_TM]                                 : Thermal Rule name
algo_type         monitor                : Algorithm Type
sensor            tz_sensor_zone*/Thermistor  : Sensor Type: TSENS/Thermistor
sampling          1000                   : Sampling period of 1000msec
thresholds        95000     100000  105000    : thresholds set in C, 95C, 100C and 105C
thresholds_clr    90000     950000  100000    : clear points set 90C, 95C and 100C
actions          <device>  <device> <device>  : Mitigation Device Type
action_info       1            2        3     : Mitigation Levels
```

Note: * <device> Refer thermal read me file for types mitigation devices supported
/vendor/qcom/proprietary/thermal-engine/readme.txt

Threshold or Monitor Algorithm (cont.)

- Measured temperature crosses a predefined threshold and then sets a predefined mitigation level.
- When sensor temperature reaches 70°C, the GPU frequency is reduced from 500 MHz to 333 MHz.
- Final sensor temperature is maintained at 85°C.



	Level 1	Level 2	Level 3
Threshold set	70°C	77°C	85°C
Threshold clear	65°C	72°C	80°C
GPU frequency	333 MHz	200 MHz	100 MHz

DTM

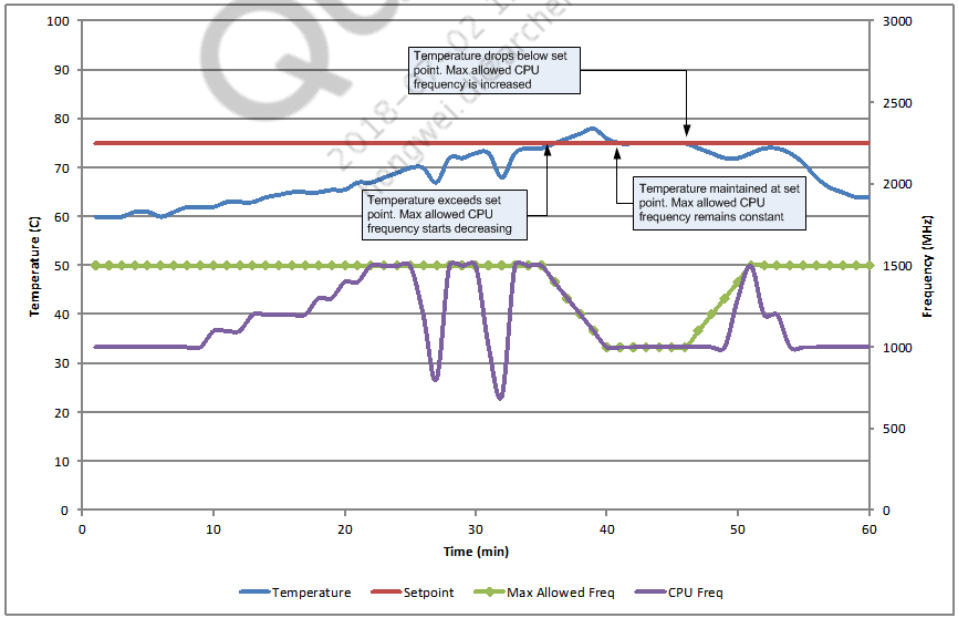
- DTM is recommended over monitor for CPU and GPU mitigation because it greatly decreases tuning effort, boosts performance, and more strictly maintains temperature to OEM set point.
- The dynamic algorithm adjusts performance based on the difference between a sensor measurement and a set point temperature.
- If the measured temperature is above the set point the algorithm steps the maximum allowed frequency of the CPU and/or GPU down. The algorithm continues to monitor the temperature and adjust the frequency maximum down until the measured temperature is at or below the set point.

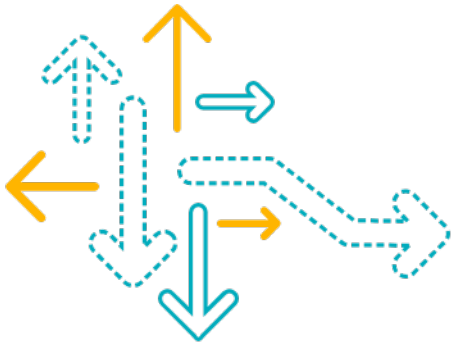
DTM configuration example:

```
[DTM_CONF]                                  : Thermal Rule name
algo_type    SS                             : Algorithm Type
sampling     65                             : Sampling period of 65 msec
sensor       tz_sensor_zone*/Thermistor     : Sensor Type: TSENS/Thermistor
device       cpu/gpu                        : Device(cpu/gpu)to be mitigated
set_point     95000                         : Thresholds set point in degree Celsius - 95˚C
set_point_clr 90000                         : Threshold clear set points – 90 ˚C
```


DTM (cont.)

- When the temperature crosses a set point (75°C), reduce performance until temperature stabilizes. The Polling mode is enabled based on the sampling parameter defined in the rule.
- By reducing performance while above the threshold and as the temperature falls below the set point (75°C), the maximum allowed frequency is allowed to ramp back up.
- If the temperature drops further below the set point clear (50°C), the Interrupt mode is re-enabled.
- This is used only for CPU and/or GPU control.





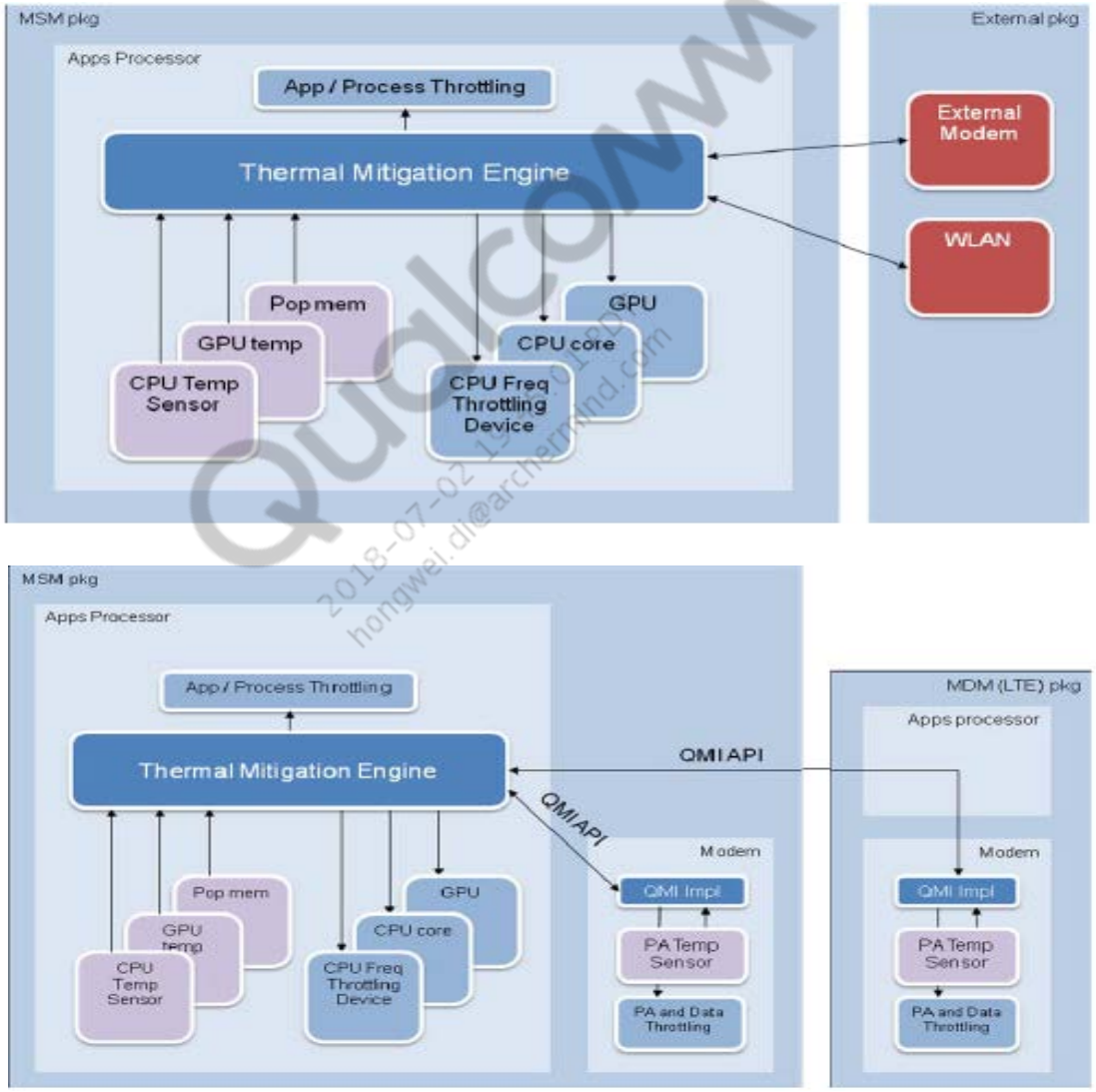
Modem Thermal Management

Overview

- Managed by the thermal engine
- Temperature is reported to the thermal engine by house keeping analog-to-digital converter (HKADC) via Qualcomm modem interface (QMI).
- The thermal engine receives temperature reading from thermistors and temperature sensor (TSENS)
- PA is the hottest component in modem-centric scenario
- Actions and thresholds are defined in the thermal engine configuration file as follows:
  - Thermal-engine.conf – TSENS
  - Modem embedded file system (EFS) – PA
- Perform one of the following methods to control the temperature:
  - Preferred or first method – Keep the original power class and limit the uplink (UL) data throughput, while performing duty-cycling (DTx).
  - Not a preferred method; however, it is required as another tool to reduce the probability of reaching the Emergency state – Reduce the power class of the device and lower the maximum Tx power, to limit the power dissipation of the power amplifier
  - DL data traffic is controlled by user equipment (UE) acknowledgment (ACK) and/or no acknowledgment (NACK) packets to network (physical uplink common control channel (PUCCH) backoff)
  - Limit the downlink (DL) throughput by dropping carrier components (CC) and fallback to 2Rx from 4Rx. This can be supported for MSM™ chipsets with Tabasco modem only.
  - Emergency state – Call shutdown; device goes to limited service and allows only E911 calls

Note: MSM8917, MSM8937 is based on Jolokia Cat4 modem, whereas MSM8940 is based on Tabasco Cat6 modem.

Modem Mitigation Legacy PA Thermal Mitigation Algorithms



Modem Mitigation Legacy PA Thermal Mitigation Algorithms (cont.)

- Legacy PA thermal mitigation algorithm is same as previous targets, which use modem as thermal action device.
- The legacy PA thermal mitigation algorithm supports four levels of thermal adjustment:
  - Level 0 – No restriction, full modem performance
  - Level 1 – Requests the modem to run the data throughput reduction algorithms
  - Level 2 – Maximum transmit power limit (MTPL) backoff and/or PUCCH backoff
  - Level 3 – Puts the modem into Limited Service mode, in which only emergency 911 calls are allowed

Sample configuration:

[pa_therm0]

algo_type	monitor		
sensor	pa_therm0		
sampling	1000		
thresholds	75000	95000	105000
thresholds_clr	65000	85000	100000
actions	modem	modem	modem
action_info	1	2	3

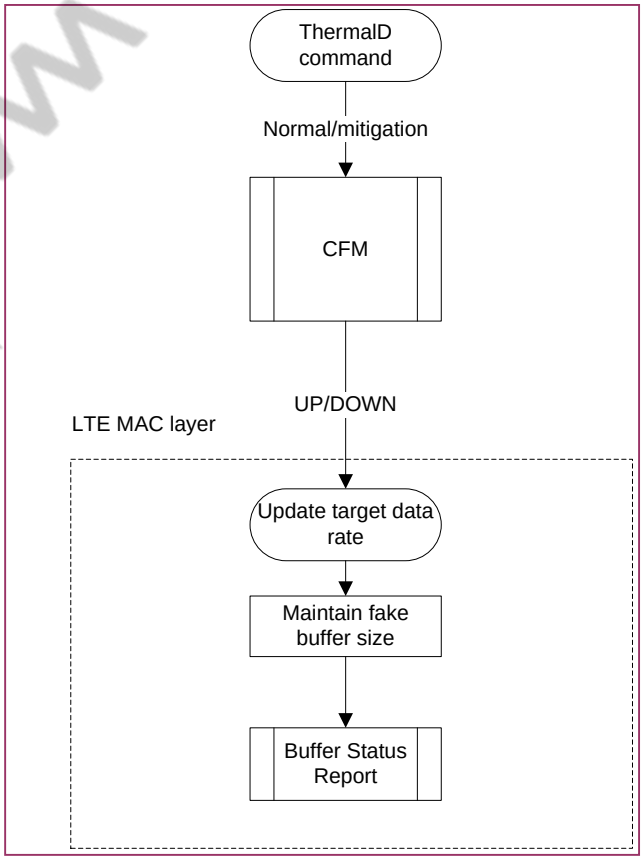
Note: Tx backoff and/or PUCCH backoff power backup is enabled by EFS configuration settings.

Modem Mitigation Legacy PA Thermal Mitigation Algorithms (cont.)

- Modem mitigation level 2 supports DL throttling and PA power backoff throttling.
- DL throttling is enabled by default
  - If tm_mechanism is set to 00 and the EFS file tx_power_backoff is pushed, the MTPL backoff will kick-in along with PUCCH backoff
- Tx power or PUCCH backoff alone can be configured by pushing EFS file
  - If CR 1105869 is reverted (DL throttling at level 2 thermal mitigation)
- The tx_power_backoff and the tm_mechanism EFS files must be located at /nv/item_files/modem/lte/ML1/.

Mitigation Level 1 (PA Sensor) – UL Data Throttle

- NV 65611 defines the flow-control target LTE data rates; these data rates are expressed in number of bytes per millisecond.
- NV 65676 step timer in seconds for changing the UL data rate states; the default value is 15 sec.
- With the centralized flow manager, the UE sends fake buffer status reports to the network based on the target rate; therefore, the network assigns lower grant based on the same.



Mitigation Level 1 (PA Sensor) – UL Data Throttle (cont.)

- The following is the default target MAC-level data rate configuration for UL data throttle in software:

```
Uint8 num_state= 10;
Uint8 default_state= 5;
Uint16 reserved = 0; /* keep this set to 0 */
 /* number of bytes per TTI (ms) */
Uint32 target_rate[0] = 6250 (50 Mb/s)
Uint32 target_rate[1] = 5000 (40 Mb/s)
Uint32 target_rate[2] = 3125 (25 Mb/s)
Uint32 target_rate[3] = 1250 (10 Mb/s)
Uint32 target_rate[4] = 625 (5 Mb/s)
Uint32 target_rate[5] = 125 (1 Mb/s)
Uint32 target_rate[6] = 63 (500 kb/s)
Uint32 target_rate[7] = 13 (100 kb/s)
Uint32 target_rate[8] = 6 (50 kb/s)
Uint32 target_rate[9] = 1 (10 kb/s)
```

Note:

- Thermal management is active only when using a non-GCF SIM, that is, a SIM that is not programmed with MCC-MNC (1-1).
- For LTE flow control to work, network and/or network simulator must support dynamic scheduling, that is, scheduling based on buffer status reported by the UE.

Mitigation Level 1 (PA Sensor) – UL Data Throttle (cont.)

- To change or configure the data rate, use the following EFS structure and write num_state, default_state, target_rate[0]…target_rate[num_state-1]. The data rate is expressed in bytes per miliseconds:

```
Uint8 num_state = 10;
Uint8 default_state= 5;
Uint16 reserved = 0; /* keep this set to 0 */
 /* number of bytes per TTI (ms) */
Uint32 target_rate[0] = 6250 (50 Mb/s)  //0x186A ( in EFS write it as 6A180000 )
Uint32 target_rate[1] = 5000 (40 Mb/s)  //0x1388 ( in EFS write it as 88130000 )
Uint32 target_rate[2] = 3125 (25 Mb/s)
Uint32 target_rate[3] = 1250 (10 Mb/s)
Uint32 target_rate[4] = 625 (5 Mb/s)
Uint32 target_rate[5] = 125 (1 Mb/s)  //Default state as configured on EFS
Uint32 target_rate[6] = 63 (500 kb/s)
Uint32 target_rate[7] = 13 (100 kb/s)
Uint32 target_rate[8] = 6 (50 kb/s)
Uint32 target_rate[9] = 1 (10 kb/s)
```

- EFS filename – lte_fc_macul_target_rates
- EFS file location – /nv/item_files/modem/lte/common/
- Sample EFS content

```
6A180000 88130000 350C0000 E2040000
71020000 7D000000 3F000000 0D000000
06000000 01000000
```


Mitigation Level 1 (PA Sensor) – UL Data Throttle (cont.)

- To revert to the default configuration, use the QPST EFS Explorer to remove the /nv/item_files/modem/lte/common/lte_fc_macul_target_rates file.
- For flow control target data rates, the default value of the state is 10. The default rate is currently set to 1 Mbps. The minimum rate can be set to a lower value. However, the value must be set to meet the requirement for control channels and delay sensitive applications. The value must not be set to 0 as it shuts down the UL and causes the call to eventually drop.
- The default state is selected to guarantee a timely response that reduces the temperature. Set the default state to 5 in the default configuration. The data rate is initially reduced to 1 Mbps after the UL flow control is triggered.

```
Uint8 num_state = 10;
Uint8 default_state = 5;
Uint16 reserved = 0; /* keep this set to 0 */
 /* number of bytes per TTI (ms) */
Uint32 target_rate[0] = 6250 (50 Mb/s)
                    ::::
Uint32 target_rate[5] = 125 (1 Mb/s)
                    ::::
Uint32 target_rate[9] = 1 (10 kb/s)
```

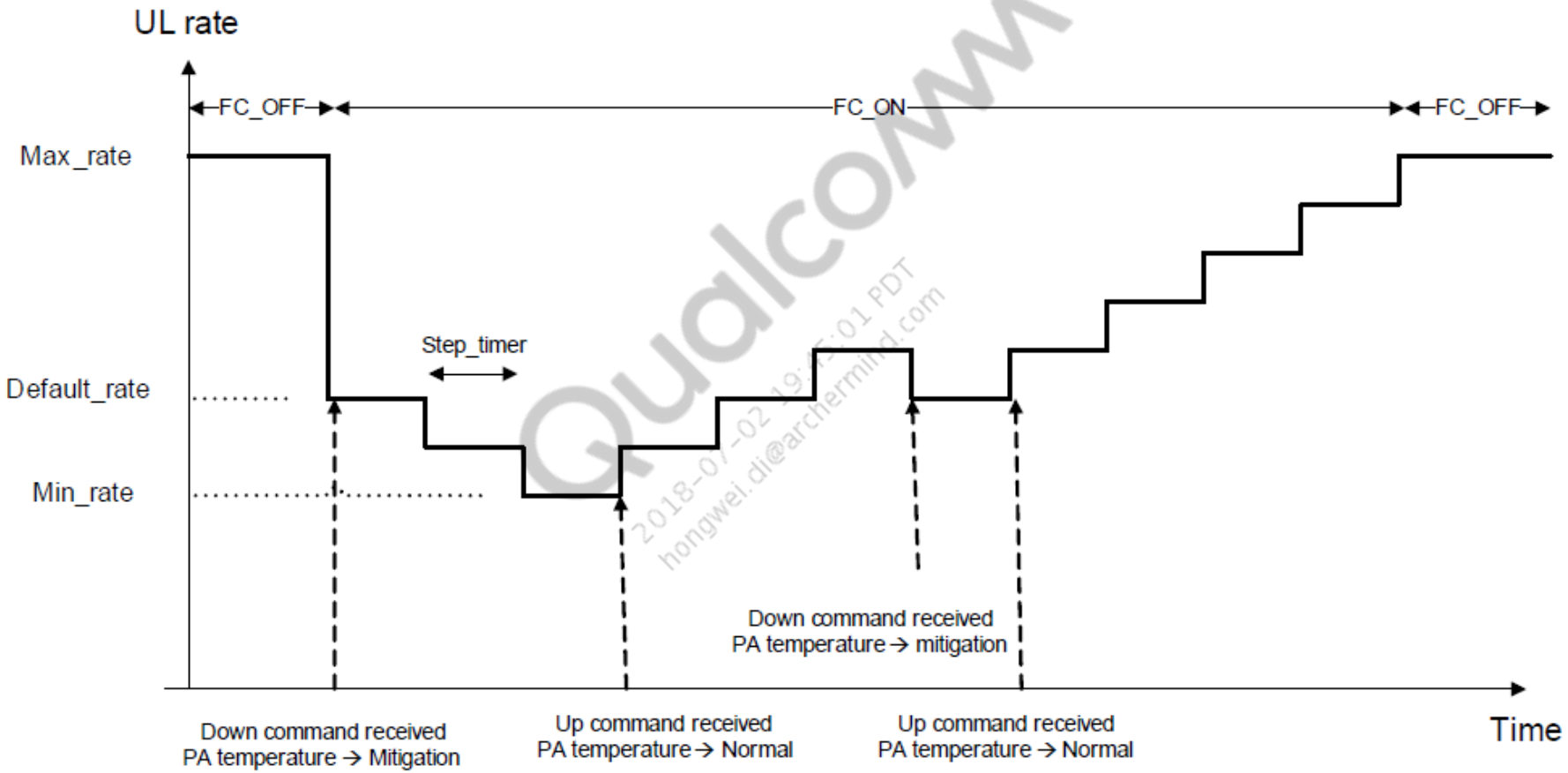
Mitigation Level 1 (PA Sensor) – UL Data Throttle (cont.)

- NV 65676 step timer is in seconds to change the rate states. The default value is 15 seconds.
- With a centralized flow manager, the UE sends fake buffer status reports to the network based on the target rate. Therefore, the network assigns lower grant based on the same.

Note:

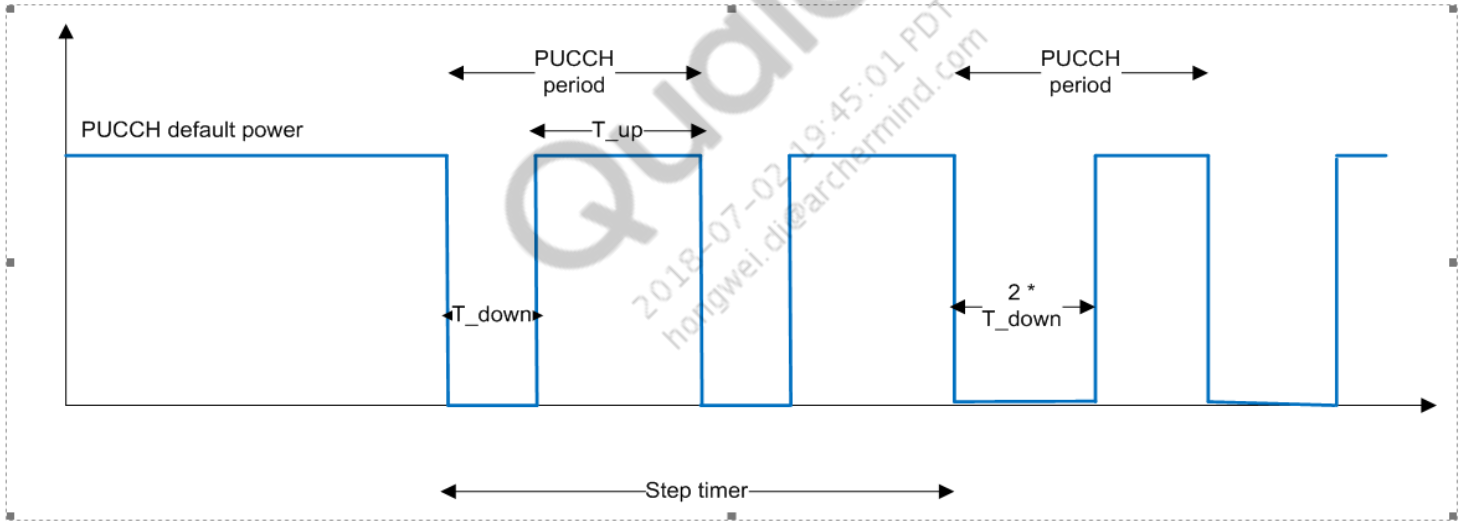
- For LTE flow control to work, network and/or network simulator must support dynamic scheduling, that is, scheduling based on the buffer status reported by the UE.
- NV 65611 cannot be used to configure the data rate at level 1, only the EFS method can be used to do this.

Mitigation Level 1 (PA Sensor) – UL Data Throttle (cont.)

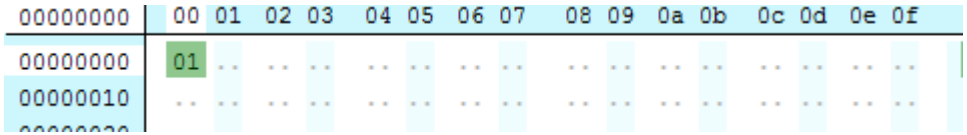


Mitigation Level 2 (PA Sensor) – PUCCH Backoff

- DL throttling is enabled by default
- Data traffic is controlled by UE ACK and/or NACK packets to network.
- During T_down, UE does not transmit any HARQ ACK/NACK on PUCCH.
- T_down + T_up is kept constant, called the PUCCH period. PUCCH duty cycle can be adjusted by changing T_down and T_up.



- To enable DL throttle (PUCCH backoff), the EFS hexadecimal file tm_mechanism (content: 01) must be present at /nv/item_files/modem/lte/ML1/.



Mitigation Level 2 (PA Sensor) – PUCCH Backoff (cont.)

- Default PUCCH backoff settings

```
pucch_cancel_info->default_state_fc = 4;
/* Default state for Thermal mitigation */
pucch_cancel_info->default_state_tm = 4;
pucch_cancel_info->num_states = 6;
pucch_cancel_info->step_timer_fc = 400;

/* Step Timer for each state for thermal  mitigation */
pucch_cancel_info->step_timer_tm = 30000;

pucch_cancel_info->timer_info[0].t_off = 100; /* Off timer */
pucch_cancel_info->timer_info[0].t_on = 100; /* On timer */

pucch_cancel_info->timer_info[1].t_off = 80;
pucch_cancel_info->timer_info[1].t_on = 120;

pucch_cancel_info->timer_info[2].t_off = 60;
pucch_cancel_info->timer_info[2].t_on = 140;

pucch_cancel_info->timer_info[3].t_off = 40;
pucch_cancel_info->timer_info[3].t_on = 160;

pucch_cancel_info->timer_info[4].t_off = 20;
pucch_cancel_info->timer_info[4].t_on = 180;

pucch_cancel_info->timer_info[5].t_off = 10;
pucch_cancel_info->timer_info[5].t_on = 190;
```


Mitigation Level 2 (PA Sensor) – PUCCH Backoff (cont.)

- To change or configure the PUCCH throttle information, use the following EFS structure and write num_states, default_state_tm, default_state_fc, padding, t_on, and t_off as follows:

Structure lte_ml1_nv_cfg_pucch_cancel_info_s

```
struct {
   /* Number of states */
   uint8 num_states;------------------------------------ 6 //0x06

   /* Default state for Thermal mitigation */
   uint8 default_state_tm;------------------------------- 4 //0x04

   /* Default state for CPU based Flow control */
   uint8 default_state_fc;------------------------------- 4 //0x04

  /* Padding */
  uint8 padding;---------------------------------------- 0 //0x00

  struct {
     /* On timer */              ------------------------ 100 0x00 64 ( Write in EFS 64 00)
     uint16 t_on;

     /* Off timer */
     uint16 t_off;               -------------------------100 0x00 64   ( Write in EFS 64 00)

   }timer_info[LTE_ML1_NV_CFG_MAX_PUCCH_CANCEL_STATES];

     /* Step Timer for each state for thermal mitigation */
     uint32 step_timer_tm;--------------------------------3000 //0x00007530(Write in EFS 3075 0000)


     /* Step Timer for each state for CPU flow control */
     uint32 step_timer_fc;--------------------------------400 //0x0000 0190(Write in EFS 9001 0000)
}
```


Mitigation Level 2 (PA Sensor) – PUCCH Backoff (cont.)

- EFS file pucch_cancel
- EFS location: /nv/item_files/modem/lte/ML1/
- Sample EFS content:

```
06 04 04 00  64 00 64 00  6e 00 5a 00  78 00 50 00
82 00 46 00  8c 00 3c 00  96 00 32 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  30 75 00 00
90 01 00 00  .. .. .. ..  .. .. .. ..  .. .. .. ..
```


PUCCH Backoff Log

F3 Logs:

```
2015 Jan  1  00:16:37.791  lte_ml1_common_fc.c  1048   H  PUCCH Backoff down received.
2015 Jan  1  00:16:37.811  lte_ml1_common_fc.c  697    H  Off timer expiry. PUCCH
Backoff OFF
2015 Jan  1  00:16:37.841  lte_ml1_common_fc.c  1167   H  PUCCH FC OFF cmd received.
```

Log Packet: Ml1 intentionally punctures the ACK/NACK information to reduce the CPU usage
1980 Jan 6 00:27:54.827 [27] 0xB173 LTE PDSCH Stat Indication

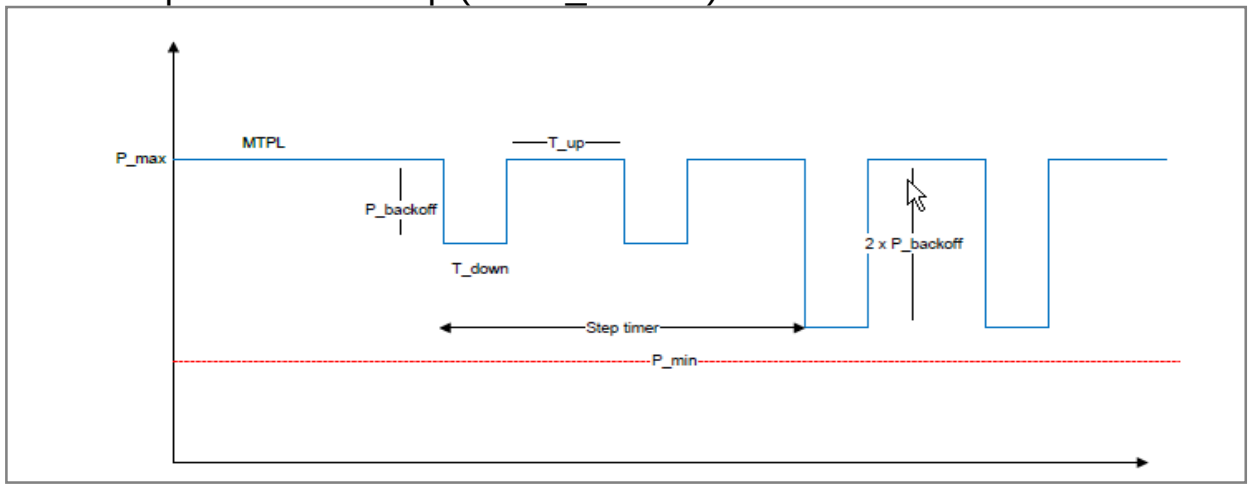
```
| 18|      0|   27| 50|     2|       2|  PCell|   1| 0|  0|  Pass|       C|    0|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
|   |       |     |   |      |        |       |   1| 0|  1|  Pass|       C|    1|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
| 19|      1|   27| 50|     2|       2|  PCell|   2| 0|  1|  Pass|       C|    0|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
|   |       |     |   |      |        |       |   2| 0|  0|  Pass|       C|    1|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
| 20|      2|   27| 50|     2|       2|  PCell|   3| 0|  1|  Pass|       C|    0|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
|   |       |     |   |      |        |       |   3| 0|  0|  Pass|       C|    1|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
| 21|      3|   27| 50|     2|       2|  PCell|   4| 0|  1|  Pass|       C|    0|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
|   |       |     |   |      |        |       |   4| 0|  1|  Pass|       C|    1|     None|       No|   4590| 28|     64QAM| 50|     ACK|    |    |
```

```
2015 Jan  1  00:16:37.823  [93]  0xB173  LTE PDSCH Stat Indication
Version      = 5
Num Records  = 22
Records
```

#	Subframe Num	Frame Num	Num RBs	Num Layers	Transport Blocks Present	Serving Cell Index	HARQ ID	RV	NDI	CRC Result	RNTI Type	TB Index	Discarded reTx	Did Recombining	TB Size (bytes)	MCS	Modulation Type	Num RBs	ACK/NACK Decision	PMCH ID	Area ID
0	4	27	50	2	2	PCell	5	2	0	Fail	C	0	Present	No	4590	28	64QAM	50	ACK		
							5	2	1	Fail	C	1	Present	No	4590	28	64QAM	50	ACK		
1	5	27	50	2	2	PCell	6	2	0	Fail	C	0	Present	No	4590	28	64QAM	50	ACK		
							6	2	1	Fail	C	1	Present	No	4590	28	64QAM	50	ACK		

Mitigation Level 2 (PA Sensor) – Tx Power Backoff

- DL throttling is enabled by default
  - If tm_mechanism is set to 00 and the EFS file tx_power_backoff is pushed, the MTPL backoff will kick-in along with PUCCH backoff
- Tx power or PUCCH backoff alone can be configured by pushing EFS file
  - If CR 1105869 is reverted (DL throttling at level 2 thermal mitigation)
- The PA maximum transmit power is adjusted per the parameters configured in the EFS file (tx_power_backoff) located at /nv/item_files/modem/lte/ML1/
- Values that can be configured in the .efs file are:
  - P_backoff – Initial value for Tx power backoff in dB (at each step n, the value of power backoff is n × P_backoff)
  - T_on – Length of time when the UE removes the limit on MTPL
  - T_off – Length of time when the UE reduces MTPL
  - Step_timer – Time spent in each step (see P_backoff)



Mitigation Level 2 (PA Sensor) – Tx Power Backoff (cont.)

- Structure of tx_power_backoffis located at /nv/item_files/modem/lte/ML1/

```
/* Initial backoff*/
uint16 p_backoff;
/* Maximum value of the backoff*/
uint16 p_backoff_max;
/* Time for non-backed-off value of power */
uint16 t_on;
/* Time for backed off Value of power */
uint16 t_off;
/* Timer for each step of the backoff*/
uint32 step_timer;
```


- Example
  - If the hexadecimal content of the file is 05000C00 32003200 983A0000, then the following values are set:
    - P_backoff – 5 dB (0500 for 5 dB)
    - Max_backoff – 12 dBm (0C00 for 13 dB)
    - T_on – 50 ms (3200 for 50 ms)
    - T_off – 50 ms (3200 for 50 ms)
    - Step_timer – 15 s (983 A for 15 s)

- Same default backoff values (5 dB, 10 dB, and 12 dB) are applied for each carrier for both intraband or interband UL CA.

Modem Mitigation (MSM8940 Only)

- MSM8940 is derived from MSM8937 family with Cat6 Tabasco modem
- Introduction of new Silver cores in Tabasco modem:
  - Tabasco modem has two Hexagon processors for software and firmware operations (Q6A and Q6B with two Silver cores).
  - The two Silver co-processors in Tabasco perform vector and a minimal set of controlled firmware operations, such as physical layer processing (LTE and WCDMA) and carrier aggregation.
  - Hexagon Silver core runs at the same frequency as Q6A core and it can be power collapsed independently when Hexagon is active.
  - Silver cores are power collapsed by default and come to online whenever technical areas are requested.
- As the modem use cases consume more power due to increasing downlink and uplink data rates and increasing complexity of processing algorithms, Tabasco modem has been introduced with two new modem thermal mitigation algorithms to control Hexagon Silver core die temperature.
  - Modem thermal mitigation algorithms for LTE use cases
  - Modem thermal mitigation algorithms for WCDMA use cases
  - Legacy PA thermal mitigation algorithms

Modem Hexagon Silver Core Thermal Mitigation – LTE and/or WCDMA

- Modem Hexagon Silver core thermal mitigation is required for LTE and/or WCDMA dual carrier HSDPA use cases with VDD_MSS operating point of TURBO.
- The Silver thermal mitigation module runs on the application processor and sends a QMI indication to the LTE ML1/WCDMA (DC-HSDPA) modem client when thermal mitigation is required. The goal is to allow VDD_MSS to reduce from TURBO to NOM.
- The algorithms will be triggered when the temperature sensor (tsens1) located near the Hexagon Silver cores indicates that the junction temperature has crossed the configurable mitigation threshold.

Modem Hexagon Silver Core Thermal Mitigation – LTE and/or WCDMA (cont.)

- LTE modem thermal mitigation:
  - The Silver core thermal mitigation module runs on the Apps and sends a QMI indication to the LTE ML1 modem client when thermal mitigation is required.
  - For Level 1, the secondary component carrier (SCC) is dropped by declaring vRLF, which causes the UE to report CQI = 0 to the eNodeB. This allows the VDD_MSS voltage to drop from TURBO and also reduces the Hexagon Silver processing load.
  - The second action defined is level 3 in which case the call manager will shut down the LTE data call and only allow emergency voice calls.

Mitigation level	Action	Comment
Level 0	No mitigation	No thermal condition
Level 1	Declare vRLF on SCC	Tj enters level 1 region
Level 2	Not defined	Not defined
Level 3	LTE shutdown	Emergency calls only

Modem Hexagon Silver Core Thermal Mitigation – LTE and/or WCDMA (cont.)

- WCDMA modem thermal mitigation:
  - The Silver core thermal mitigation module runs on the Apps and will send a QMI indication to the WCDMA dual carrier HSDPA modem client when thermal mitigation is required. For Tabasco, this corresponds to DC-HSDPA with Qualcomm interference cancellation and equalization (QICE) interference cancellation enabled.
  - If the junction temperature for this use case exceeds the Level 1 threshold, then the QICE algorithm will be disabled regardless of how many interfering neighbor cells are being processed by QICE. The goal is to allow VDD_MSS to reduce from TURBO to NOM.
  - The second action defined is level 3 in which case the call manager will shut down the WCDMA data call and only allow emergency voice calls.

Mitigation level	Action	Comment
Level 0	No mitigation	No thermal condition
Level 1	Disable QICE	Tj enters level 1 region
Level 2	Not defined	Not defined
Level 3	WCDMA shutdown	Emergency calls only

Modem Hexagon Silver Core Thermal Configuration

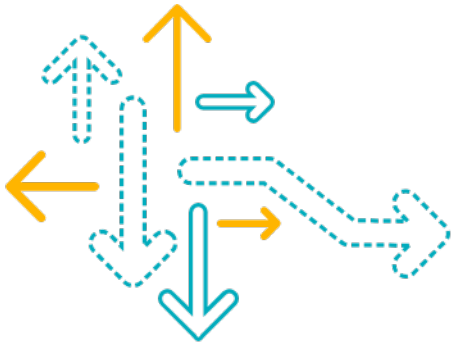
- Based on Qualcomm® Reference Design (QRD) thermal profiling, the Modem_proc mitigation threshold is configured such that, the performance of stand-alone LTE and/or WCDMA use cases are not impacted.

Sample configuration:

MODEM_PROC_TEMP_MITIGATION]

```
algo_type monitor
sampling 1000
sensor tsens_tz_sensor1
thresholds 75000 80000
thresholds_clr 70000 75000
actions modem_proc modem_proc
action_info 1 3
```


Trigger	Mitigation level	Mitigation action LTE and/or WCDMA	Comment LTE and/or WCDMA
Modem temperature	0	No mitigation	–
–	1	Drop SCell by declaring vRLF and/or Disable QICE	VDD_MSS reduced to NOM from TURBO
–	2	Unused	–
–	3	Shutdown (emergency call only)	–



Thermal Configuration

Default Thermal Configuration

- QTI provides a default thermal configuration that is embedded in the device source code. This configuration must be tuned to meet the OEM's unique requirements; refer to *Presentation: Thermal Tuning Procedure* (80-N9649-1).
- View the default thermal configuration by using the ADB command `thermal-engine-o`.
  - This ADB command prints the existing thermal rules, including QTI defaults and custom OEM settings.
- The default configuration includes:
  - Rules for junction temperature management (85°C by default)
    - Label examples – [SS-CPU0], [SS-CPU1], [SS-CPU2], [SS-CPU3], and [SS-CPU4-5-6-7]
    - These rules should not be increased from their default values.
    - The following branches contain the source code of embedded rules:
      - /vendor/qcom/proprietary/thermal-engine/ss-data.c
      - /vendor/qcom/proprietary/thermal-engine/thermal_monitor-data.c
  - Rules for skin temperature management
    - Should be added by the OEM to thermal-engine.conf and pushed to /system/etc/thermal-engine.conf
  - Other default rules should not be changed, for example, VDD_RSTR_MONITOR-TSENSX

Add Custom Thermal Configuration to Device

- Custom thermal configurations can be added to a device without recompiling the source code.
- Add a new rule by placing the new rule in the file named thermal-engine.conf and pushing it to the device (/system/etc/thermal-engine.conf) using ADB.

```
adb push <location_of_thermal-engine.conf> /system/etc/thermal-engine.conf
```

- For example, to add a rule for GPU, place the following in thermal-engine.conf:

```
[SS-GPU]
algo_type ss
sampling 65
sensor tsens_tz_sensor12
device gpu
set_point 60000
set_point_clr 57000
time_constant 0
```

- The preceding example adds a rule named [SS-GPU] to the thermal configuration.
- Reboot the device after adding or changing thermal-engine.conf.

Add Custom Thermal Configuration to Device (cont.)

- Replace a default rule by adding a rule to thermal-engine.conf with the same name as the default rule.
- For example, [SS-POPMEM] is a default rule; if a rule with the same name is added to thermal-engine.conf, the default rule is overridden.

```
[SS-POPMEM]
algo_type ss
sampling 250
sensor pop_mem
device cluster1
set_point 65000
set_point_clr 55000
time_constant 2
```


- The preceding rule overrides the default [SS-POPMEM] rule of 80°C and lowers it to 65°C.
- Reboot the device after adding or changing thermal-engine.conf.

Add Custom Thermal Configuration to Device (cont.)

- Disable a default rule by adding a rule to thermal-engine.conf with the name of the rule to disable, followed by disable 1.

```
[SS-POPMEM]
disable 1
```

- The preceding example disables the [SS-POPMEM] rule.
- Reboot the device after adding or changing thermal-engine.conf.

Thermal Configuration File Example

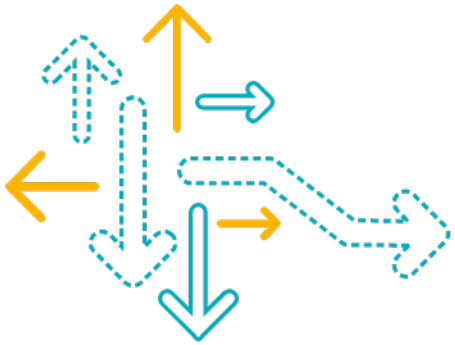
```
[lcd_gpu_management]
algo_type      monitor
sensor         tsens_tz_sensor3
sampling       1000
thresholds     50     65
thresholds_clr 45     60
actions      gpu+lcd
action_info  500000+255   333000+150
```

```
[VIRTUAL-CPUS]
algo_type virtual
trip_sensor tsens_tz_sensor5
set_point 75000
set_point_clr 65000
sensors tsens_tz_sensor5 tsens_tz_sensor6
tsens_tz_sensor7 tsens_tz_sensor8 tsens_tz_sensor9
weights sampling 50 math 2
```

```
[MODEM_PROC_TEMP_MITIGATION]
   algo_type monitor
   sampling 1000
   sensor tsens_tz_sensor1
   thresholds 75000 80000
   thresholds_clr 70000 75000
   actions modem_proc modem_proc
   action_info 1 3
```

New Embedded Thermal Rule

```
[SS-CPUS-ALL]
algo_typess
sampling 10
sensor VIRTUAL-CPUS
device cpu_voltage
set_point95000
set_point_clr65000
time_constant0
```

New Thermal Features

Case Thermistor to Control Device Skin Temperature

- New ADC channel to read external thermistor “case_therm” for skin temperature control.
- It is recommended that all OEMs include the case thermistor sensor in their design to better tune and maintain the case temperature within the specification.

Example code

```
arch/arm/boot/dts/qcom/msm8952.dtsi
   qcom,sensor-information {
   sensor_information6: qcom,sensor-information@6 {
   qcom,sensor-type = "adc“; qcom,sensor-name = "case_therm";
     };
}

arch/arm/boot/dts/qcom/msm8952-qrd-skuc.dtsi
&pm8952_vadc {
   chan@13 {
   label = "case_therm";
   reg= <0x13>;
   qcom,decimation= <0>;
   qcom,pre-div-channel-scaling = <0>;
   qcom,calibration-type = "ratiometric";
   qcom,scale-function = <2>;
   qcom,hw-settle-time = <2>;
   qcom,fast-avg-setup = <0>;
   qcom,vadc-thermal-node;
   };
};
```


Core Control Feature

- Core control is introduced to preemptively put the cores in the Offline mode and bring them back to the Online mode when necessary.
- The goal of core control is to improve thermal conditions and save power by keeping cores offline unless they are required. This also helps to improve user experience performance.
- Thermal hotplug of a CPU changes the number of CPUs available for core_control.

Note: For more details, refer to *Core Control Feature* (80-P0106-1).

Battery Current Limit (BCL)

- In some concurrent use case scenarios, there may be a high-voltage drop and excessive current drawn from the battery by CPU cores and other peripheral loads such as camera flash, GSM PA, display, and so on, especially at a low battery voltage.
- The PMIC has a BCL hardware to prevent battery undervoltage lockout (UVLO) and overcurrent protection (OCP) situations.
  - When the PMIC input voltage drops below the operational value, the UVLO circuit turns off the power to protect the device.
  - When the battery current exceeds the operating threshold for too long, the OCP in the battery breaks the circuit and the phone is shut down abruptly.
- Abrupt phone shutdown does not provide good user experience.
- The BCL software mechanism uses PMIC fuel gauging hardware to mitigate the current drawn from the battery by reducing the CPU load and ensures that OCP and UVLO do not occur.

Note: For more details, refer to *Battery Current Limit (BCL) Overview and Tuning* (80-NM328-709).

Multizone Temperature Control Engine

- Thermal issues become critical
- Latency of thermal issue detection to mitigation is high
  - Hardware tsense latency is improved from 65 ms to 4 ms
  - Software latency is the bottleneck – avg ~5 ms, worst case 30 ms
  - Thermal ramp – about 10ºC at 24 ms (75ºC to 85ºC)
- High latency requires large margins
- MTC hardware block (Tsens_wrapper) issues clock throttling commands to pulse swallower (RCGwTC)
- Software programmable throttling
  - Throttling table (steps of 3%)
  - Thresholds
  - Disable or enable
- Software DCVS loop stays as is
- Advantages
  - Simple but fast and effective
  - Generic solution for any subsystem
- By default it is enabled only at 105ºC

Multizone Temperature Control Engine (cont.)

- MTC sample configuration:

```
boot_images/core/hwengines/tsens/config/8937/TsensBootBsp.c
static const TsensBootMTCConfigType aMTCConfig[] =
{
   /* Zone 0 */
   {
      /* .bIsZoneEnabled    */ TRUE,
      /* .uPSCommandTh2Viol */ TSENS_BSP_MTC_SYS_PERF_25,
      /* .uPSCommandTh1Viol */ TSENS_BSP_MTC_SYS_PERF_50,
      /* .uPSCommandCool    */ TSENS_BSP_MTC_SYS_PERF_100,
      /* .uSensorMask       */ 0x200, // S9
      /* .bIsTH1Enabled     */ TRUE,
      /* .bIsTH2Enabled     */ TRUE,
   },
   /* Zone 1 */
   {
      /* .bIsZoneEnabled    */ TRUE,
      /* .uPSCommandTh2Viol */ TSENS_BSP_MTC_SYS_PERF_25,
      /* .uPSCommandTh1Viol */ TSENS_BSP_MTC_SYS_PERF_50,
      /* .uPSCommandCool    */ TSENS_BSP_MTC_SYS_PERF_100,
      /* .uSensorMask       */ 0x1F0, // S4, S5, S6, S7 and S8
      /* .bIsTH1Enabled     */ TRUE,
      /* .bIsTH2Enabled     */ TRUE,
   }
};
```

MSM8937 Junction Temperature Management

- New thermal rule for MSM8937 – [SS-CPUS-ALL]
  - Manages junction temperature at 85^0C
  - Optimizes junction temperature management for MSM8937's single rail architecture by use of 'cpu_voltage' device
  - The 'cpu_voltage' device addresses the thermal scenario where perf cluster reaches mitigation threshold, but voltage is still held high by unmitigated power cluster thus weakening the impact of power cluster mitigation.
  - The 'cpu_voltage' device allows the SS algorithm to step down through voltage steps that are mapped to frequencies for each cluster, ensuring that the power cluster does not hold onto high voltage and prevent power reduction
  - The sensor 'VIRTUAL-CPUS' represents the hottest tsens of any CPU tsens zone sensors
  - Single thermal rule (as seen on right) manages junction temperature for all the CPUs and replaces individual thermal rules previously seen on split rail architecture chipsets.

```
[VIRTUAL-CPUS]
algo_type virtual
trip_sensor tsens_tz_sensor5
set_point 75000
set_point_clr 65000
sensors tsens_tz_sensor5 tsens_tz_sensor6
tsens_tz_sensor7 tsens_tz_sensor8 tsens_tz_sensor9
weights sampling 50 math 2
```

```
New Embedded Thermal Rule
[SS-CPUS-ALL]
algo_typess
sampling 10
sensor VIRTUAL-CPUS
device cpu_voltage
set_point85000
set_point_clr55000
time_constant0
```

MSM8937 Junction Temperature Management (cont.)

- The ‘cpu_voltage’ device uses a voltage frequency table optimized for single rail architecture.
- The table that cpu_voltage uses is created by comparing the table of the cluster with the highest voltage requirement to the other cluster's voltage table and then determining the efficient voltage frequency pairs.

Thermal engine will get the voltage info corresponding frequencies during initialization

01 19:32:33.660 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster0[0] voltage =1230 freq 1094400
01 19:32:33.660 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster0[1] voltage =1230 freq 998400
01 19:32:33.660 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster0[2] voltage =1145 freq 902400
01 19:32:33.660 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster0[3] voltage =1135 freq 806400
01 19:32:33.660 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster1[0] voltage =1230 freq 1401000
01 19:32:33.660 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster1[1] voltage =1230 freq 1344000
01 19:32:33.660 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster1[2] voltage =1230 freq 1248000
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster1[3] voltage =1145 freq 1094400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cluster1[4] voltage =1135 freq 998400

Voltage frequency table after optimization

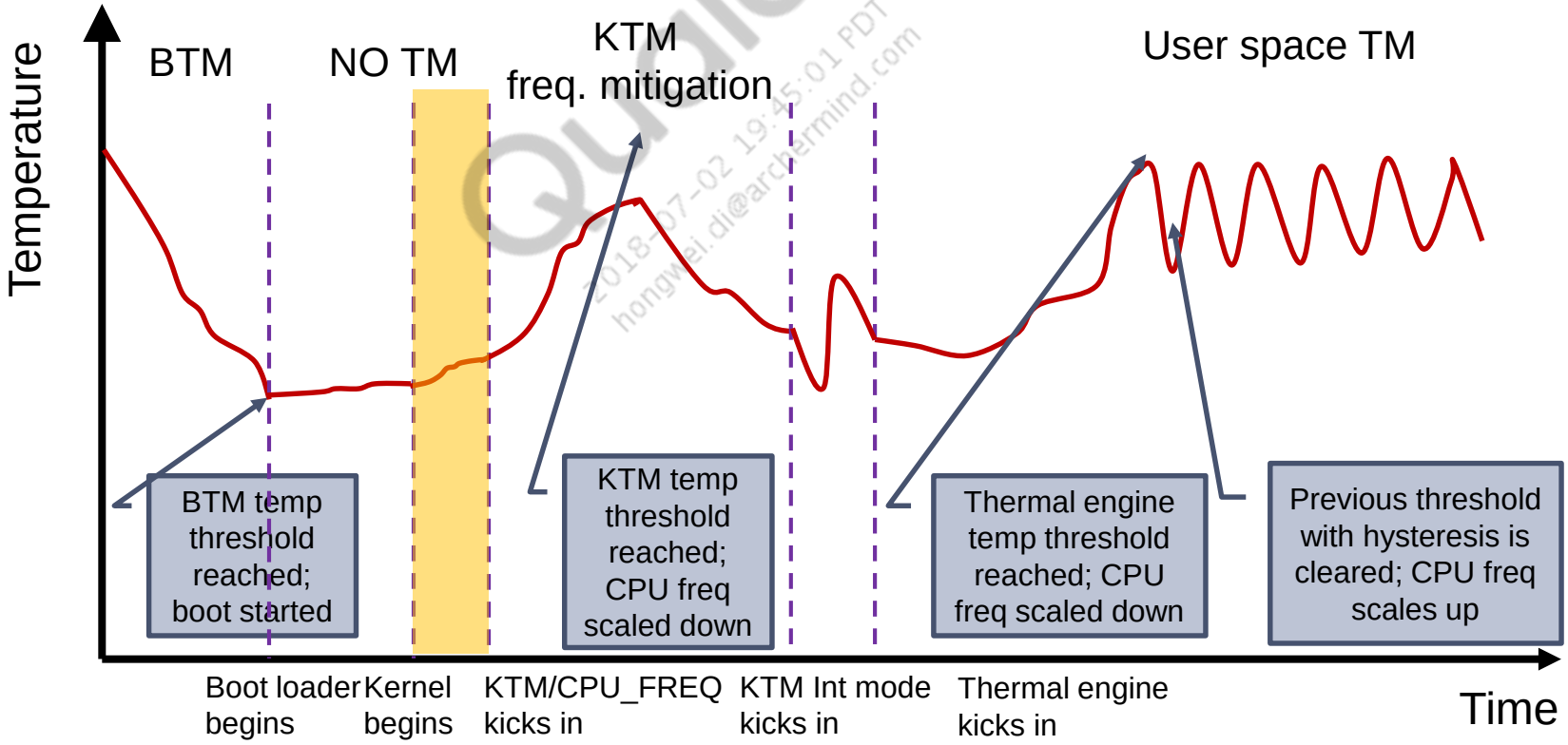
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster0[0] voltage =1230 freq 1094400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster0[1] voltage =1230 freq 998400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster0[2] voltage =1145 freq 902400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster0[3] voltage =1135 freq 806400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster1[0] voltage =1230 freq 1401000
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster1[1] voltage =1230 freq 1401000
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster1[2] voltage =1145 freq 1094400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: Sorted table cluster1[3] voltage =1145 freq 1094400

Cpu_voltage device voltage mitigation levels

01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[0]=1230 perf_lvl[0]=1094400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[1]=1230 perf_lvl[1]=998400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[2]=1145 perf_lvl[2]=902400
01 19:32:33.661 7406 7406 D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[3]=1135 perf_lvl[3]=806400

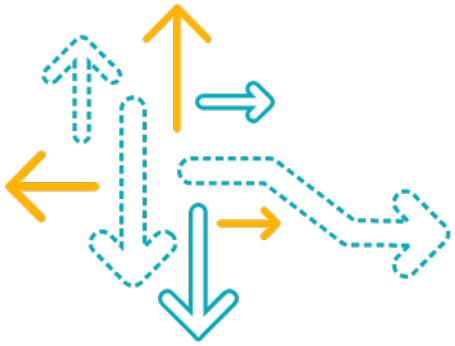
Overall Apps Processor Thermal Management Mechanism

- Three distinct TMs are provided
  - SBL temperature check
  - Rich set of kernel thermal monitor
  - Full thermal engine with KTM postboot feature enabled



Thermal Software Features and Management Devices

Feature	Description
CPU TM	Adjustment of maximum allowed operating frequency per cluster
GPU TM	Adjustment of maximum allowed operating frequency
Hotplug	Take specific core offline
CPU core shutdown	Safety mechanism to ensure CPU cores shut off before junction temperature limits are exceeded
Modem TM	Adjustment of peak data rates, maximum Tx power, and data call termination
Camcorder TM	Adjustment of encoder frame rate or encoding shutoff
WLAN TM	Adjustment of peak data rates
LCD backlight TM	Adjustment of maximum backlight intensity
Battery charging TM	Adjustment of maximum allowable charge rate
Battery current limiting	CPU mitigation based on state-of-charge level of battery
Speaker coil calibration	Automatic calibration of speaker coil resistance vs. temperature enables audio codec to protect against speaker coil damage at high temperatures and high-power output
Voltage restriction	Voltage restriction enables low operating voltage above 0°C by adjusting the required minimum voltage at temperature extremes
Kernel TM	Adjustment of maximum allowed operating frequency during kernel initialization and post-boot device protection
Override mode	Overrides thermal setpoint to allow higher performance for benchmarks or thermally aware applications
Dynamic parameter update	Important parameter sets can be updated at runtime for better OEM-specific dynamic thermal management



Thermal Debug Overview

Thermal Engine Debug Overview

- Enable more logging for KTM

```
echo 8 > /proc/sys/kernel/printk
echo 'file msm_thermal.c +p' > /sys/kernel/debug/dynamic_debug/control
```

- Thermal engine provides detailed logging on Debugging mode
- To enable Debug mode:

1. Do one of the following:
   - Keep “debug” in the first line of thermal-engine.conf and restart thermal-engine service (#thermal-engine &)
   - Start thermal engine in Debug mode manually using the following commands:

```
#stop thermal-engine (Super user mode)
#start thermal-engine -debug  &
```

   The following output is displayed in the command prompt window:

```
adb   logcat –v  time –s ThermalEngine
```

2. Enable thermal configuration on the device for thermal mitigation:

   `adb shell thermal-engine –o` (based on soc_id config is enabled, /sys/devices/soc0/sco_id)

3. Edit the configuration file as follows:

```
adb pull        /etc/thermal-engine.conf
adb               remount
<edit>
adb push        thermal-engine.conf /etc/
```


ADB Commands

- To check the GPU frequencies

```
adb shell mkdir /sys/kernel/debug
adb shell mount -t debugfs none /sys/kernel/debug
adb shell cat /sys/kernel/debug/clk/grp_3d_clk/rate
```

- To check the sensor zone type to know sensors mapping

```
cat /sys/class/thermal/thermal_zone*/type
```

For Example – tsens_tz_sensor0

```
tsens_tz_sensor4
pm8226_tz
pa_therm0
pa_therm1
```

- To check the sensor temperature

```
adb shell /sys/class/thermal/thermal_zone*/temp  *  zone number
```


Note: For more details on further debugging steps, refer to *Linux Android Software Thermal Debugging Guide* (80-NM998-1).

ADB Commands (cont.)

- To check the CPU frequency

```
adb shell  cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
adb shell cat  /sys/devices/system/cpu/cpu0/cpufreq/cpu_max_freq
adb shell cat  /sys/devices/system/cpu/cpu0/cpufreq/cpu_min_freq
```

- To check the DDR frequency

```
adb shell cat /sys/kernel/debug/clk/bimc_clk/measure x (2/1000000)
```

- To check SNoC

```
adb shell cat /sys/kernel/debug/clk/snoc_clk/measure
```

- To check PCNoC

```
adb shell cat /sys/kernel/debug/clk/pcnoc_clk/measure
```

- To check the MDP frequency

```
adb shell cat /sys/kernel/debug/clk/gcc_mdss_mdp_clk/measure
```


Key Points of Thermal Logcat

```
19:13:39.893  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster0[0] voltage =1230 freq 1094400
19:13:39.893  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster0[1] voltage =1190 freq 998400
19:13:39.893  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster0[2] voltage =1145 freq 902400
19:13:39.893  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster0[3] voltage =1135 freq 768000

19:13:39.893  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster1[0] voltage =1230 freq 1401000
19:13:39.893  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster1[1] voltage =1190 freq 1248000
19:13:39.894  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster1[2] voltage =1145 freq 1094400
19:13:39.894  D ThermalEngine: tmd_add_cpu_voltage_dev_data:  Sorted table cluster1[3] voltage =1145 freq 1094400

19:13:39.894  D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[0]=1230 perf_lvl[0]=1094400
19:13:39.894  D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[1]=1190 perf_lvl[1]=998400
19:13:39.894  D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[2]=1145 perf_lvl[2]=902400
19:13:39.894  D ThermalEngine: tmd_add_cpu_voltage_dev_data: cpu_voltage lvl_info[3]=1135 perf_lvl[3]=768000
19:13:39.894  D ThermalEngine: tmd_init_gpu_devs: gpu lvl_info[0]=450000000
19:13:39.894  D ThermalEngine: tmd_init_gpu_devs: gpu lvl_info[1]=400000000
19:13:39.894  D ThermalEngine: tmd_init_gpu_devs: gpu lvl_info[2]=375000000
19:13:39.894  D ThermalEngine: tmd_init_gpu_devs: gpu lvl_info[3]=300000000
19:13:39.894  D ThermalEngine: tmd_init_gpu_devs: gpu lvl_info[4]=216000000
19:13:39.895  I ThermalEngine: vdd_rstr_init: Init KTM VDD RSTR enabled: 0
19:13:39.897  D ThermalEngine: devices_manager_set_lvl: DEV kernel, lvl 1
19:13:39.897  D ThermalEngine: KERNEL request to keep mitigation enabled
```

Key Points of Thermal Logcat (cont.)

```
02:27:41.860 D/ThermalEngine(  605): handle_thresh_sig: SS Id SS-CPUS, Read VIRTUAL-CPUS 55000mC
02:27:41.860 I/ThermalEngine(  605): handle_thresh_sig: SS Id SS-CPUS Transition State 2
02:27:41.860 D/ThermalEngine(  605): sensors_manager_set_thresh_lvl: VIRTUAL-CPUS Hi(0) 0, Lo(1) 50000, Interval(0) 0
02:27:41.860 D/ThermalEngine(  605): update_active_thresh: VIRTUAL-CPUS Active(1), Hi(0) 2147483647, Lo(1) 50000, Interval(0) -1
02:27:41.860 I/ThermalEngine(  605): Sensor:tsens_tz_sensor5:49000 mC
02:27:41.860 I/ThermalEngine(  605): vs_get_temperature: read[0] tsens_tz_sensor5 49000 mC
02:27:41.861 I/ThermalEngine(  605): Sensor:tsens_tz_sensor7:55000 mC
02:27:41.861 I/ThermalEngine(  605): vs_get_temperature: read[2] tsens_tz_sensor7 55000 mC
02:27:41.861 I/ThermalEngine(  605): Sensor:tsens_tz_sensor8:53000 mC
02:27:41.861 I/ThermalEngine(  605): vs_get_temperature: read[3] tsens_tz_sensor8 53000 mC
02:27:41.861 I/ThermalEngine(  605): Sensor:tsens_tz_sensor9:48000 mC
02:27:41.861 I/ThermalEngine(  605): vs_get_temperature: read[4] tsens_tz_sensor9 48000 mC
02:27:41.861 D/ThermalEngine(  605): vs_get_temperature: Virtual sensor VIRTUAL-CPUS calculate results: 55000 mC
02:27:41.861 I/ThermalEngine(  605): Sensor:VIRTUAL-CPUS:55000 mC
02:27:41.861 D/ThermalEngine(  605): handle_timer_sig: SS Id SS-CPUS Read VIRTUAL-CPUS 55000mC, Err 0mC, SampleCnt 0
02:27:41.861 D/ThermalEngine(  605): handle_timer_sig: SS Id SS-CPUS, E0 0mC, E1 0mC
02:27:41.861 D/ThermalEngine(  605): devices_manager_set_op_value: DEV cpu_voltage, op_value 1290
02:27:41.861 D/ThermalEngine(  605): devices_manager_set_op_value: DEV cluster0, op_value 998400
02:27:41.861 D/ThermalEngine(  605): ACTION: CLUSTER - Setting CLUSTER[0] to 998400
02:27:41.861 I/ThermalEngine(  605): Mitigation:CLUSTER[0]:998400 Khz
02:27:41.861 D/ThermalEngine(  605): ACTION: CLUSTER - Setting CLUSTER[0] to voltage 1290 freq = 998400
02:27:41.861 I/ThermalEngine(  605): Mitigation:CLUSTER[0]: 1290 mV freq = 998400 kHz
02:27:41.861 D/ThermalEngine(  605): devices_manager_set_op_value: DEV cluster1, op_value 1248000
02:27:41.861 D/ThermalEngine(  605): ACTION: CLUSTER - Setting CLUSTER[1] to 1248000
02:27:41.861 I/ThermalEngine(  605): Mitigation:CLUSTER[1]:1248000 Khz
02:27:41.861 D/ThermalEngine(  605): ACTION: CLUSTER - Setting CLUSTER[1] to voltage 1290 freq = 1248000
02:27:41.861 I/ThermalEngine(  605): Mitigation:CLUSTER[1]: 1290 mV freq = 1248000 kHz
```

VIRTUAL-CPUS sensors threshold point calculation

cpu_voltage mitigation triggered @55C and mapping to frequency of each cluster.

```
01-03 21:30:01.948 I/ThermalEngine(286): hotplug_ktm_request: write out 2
01-03 21:30:01.958 I/ThermalEngine(286): ACTION: Hot-plugged OFF CPU[1]
01-03 21:30:01.958 E/ThermalEngine(286): TM Id HOTPLUG-CPU1 Sensor cpu1 Reading 105c
01-03 21:30:01.958 E/ThermalEngine(286): handle_thresh_sig: TM Id HOTPLUG-CPU1 Sensor cpu1 Temp 10500
01-03 21:30:01.958 E/ThermalEngine(286): TM Id 'HOTPLUG-CPU1' Sensor 'cpu1' - alarm raised 1 at 105.0°C
```

Hotplugged CPU cores at 105°C defined in `qcom,msm-thermal{ };`

Key Points of KTM Kernel Logs (Filtered by Thermal Keyword)

Line 162: [1, swapper/0][1.254433] msm-thermal qcom,msm-thermal.16: msm_thermal:Failed reading node=/soc/qcom,msm-thermal, key=qcom,rpm-phase-resource-type err=-22. KTM continues

Line 163: [1, swapper/0][1.254454] msm-thermal qcom,msm-thermal.16: msm_thermal:Failed reading node=/soc/qcom,msm-thermal, key=qcom,gfx-phase-warm-temp. err=-22. KTM continues

Line 164: [1, swapper/0][1.254505] msm-thermal qcom,msm-thermal.16: probe_vdd_mx:Failed reading node=/soc/qcom,msm-thermal, key=qcom,mx-restriction-temp. KTM continues

Failed reading nodes due to incomplete driver initialization

Line 168: [1, swapper/0][1.255635] msm_thermal:get_kernel_cluster_info CPU1 topology not initialized.

Line 424: [36, kworker/4:1][2.265345] msm_thermal:do_cluster_freq_ctrl Limiting CPU0 max frequency to 1344000. Temp:60

Line 425: [36, kworker/4:1][2.265359] msm_thermal:do_cluster_freq_ctrl Limiting CPU1 max frequency to 1344000. Temp:60

Line 426: [36, kworker/4:1][2.265369] msm_thermal:do_cluster_freq_ctrl Limiting CPU2 max frequency to 1344000. Temp:60

Line 427: [36, kworker/4:1][2.265379] msm_thermal:do_cluster_freq_ctrl Limiting CPU3 max frequency to 1344000. Temp:60

Line 428: [36, kworker/4:1][2.265389] msm_thermal:do_cluster_freq_ctrl Limiting CPU4 max frequency to 533333. Temp:60

Line 429: [36, kworker/4:1][2.265399] msm_thermal:do_cluster_freq_ctrl Limiting CPU5 max frequency to 533333. Temp:60

Line 430: [36, kworker/4:1][2.265409] msm_thermal:do_cluster_freq_ctrl Limiting CPU6 max frequency to 533333. Temp:60

Line 431: [36, kworker/4:1][2.265418] msm_thermal:do_cluster_freq_ctrl Limiting CPU7 max frequency to 533333. Temp:60

KTM Monitor mode CPU mitigation (threshold 60°C, monitors only core0 sensor) and hotplug threshold 80°C.

Line 432: [36, kworker/4:1][2.511507] msm_thermal:do_core_control Set Offline: CPU6 Temp: 81

Line 441: [36, kworker/4:1][2.761533] msm_thermal:do_core_control Set Offline: CPU5 Temp: 81

Core hotplug log

Line 1129: [1, swapper/0][19.833884] msm_thermal:interrupt_mode_init Interrupt mode init

KTM switches to Interrupt mode.

Line 1131: [1, swapper/0][19.852333] msm_thermal:disable_msm_thermal Max frequency reset for CPU0

Line 1133: [1, swapper/0][19.869191] msm_thermal:disable_msm_thermal Max frequency reset for CPU1

Line 1135: [1, swapper/0][19.889191] msm_thermal:disable_msm_thermal Max frequency reset for CPU2

Line 1136: [1, swapper/0][19.897337] msm_thermal:disable_msm_thermal Max frequency reset for CPU3

Line 1138: [1, swapper/0][19.917723] msm_thermal:disable_msm_thermal Max frequency reset for CPU4

Line 1140: [1, swapper/0][19.936190] msm_thermal:disable_msm_thermal Max frequency reset for CPU5

Line 1141: [1, swapper/0][19.946976] msm_thermal:disable_msm_thermal Max frequency reset for CPU6

Line 1143: [1, swapper/0][19.973617] msm_thermal:disable_msm_thermal Max frequency reset for CPU7

KTM releases all mitigation and sets CPU max frequency while switching to Interrupt mode and then monitors for hot plug 105ºC, core0 emergency frequency mitigation, vdd restriction, and so on.

Line 1984: [5.954276 / 01-03 06:47:31.042] msm_thermal:set_enabled enabled = 0

KTM hands over control to thermal-engine

Line 2140: [1, swapper/0][19.936190] msm_thermal:msm_thermal_bite TSENS:3 reached temperature:115. System reset

KTM triggers software reset when any of the thermal zones hit 115°C.

KTM RAM Dump Debugging

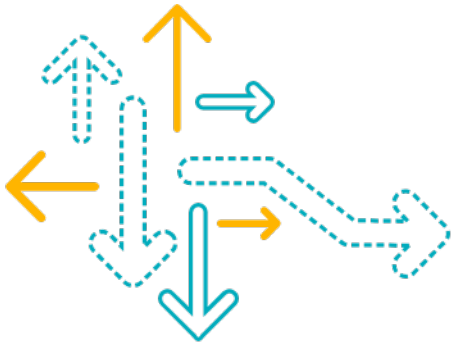
- When the Reset state is 0x1B, RAM dump is not useful because the caches are not flushed during reset.
- When the Reset state is 0x23 and if the last dmesg log says “msm_thermal:msm_thermal_bite TSENS:8 reached temperature:115. System reset”, it is a watchdog bite triggered by KTM when TSENS hits 115°C.
- KTM polls the temperature and mitigates during boot until the late_init phase of kernel boot. After late initiation of the phase, it hands over the temperature monitoring to thermal-engine (user space).
  - enabled == 0 – thermal-engine monitors temperature
  - enabled == 1 – KTM monitors temperature
- KTM has three kernel threads:
  - msm_thermal:hot-plug – Aggregates the hotplug requests to bring the CPU cores offline or online
  - msm_thermal:freq_mitig – Aggregates the scaling maximum or minimum frequency requests and mitigates the CPU frequency
  - msm_thermal:therm_monitor – Performs watchdog bite

KTM RAM Dump Debugging (cont.)

- "cpus" variable has Thermal Mitigation state for all cores
  - cpus.cpu – Logical CPU ID
  - cpus.sensor_id – TSENS monitors this particular core's temperature
  - cpus.offline – If TRUE, KTM has requested this core to be offline
  - cpus.user_offline – If TRUE, user space (thermal-engine) has requested this core to be offline
  - cpus.hotplug_thresh_clear – If TRUE, emergency hotplug threshold for this core is triggered in the hardware but not yet handled in KTM
  - cpus.user_max_freq – Holds the scaling maximum frequency requested by thermal-engine
  - cpus.user_min_freq – Holds the scaling minimum frequency requested by thermal-engine
  - cpus.max_freq – If TRUE, KTM has a request to cap the scaling maximum frequency
  - cpus.limited_max_freq – Holds the last successful scaling maximum frequency requested by KTM
  - cpus.limited_min_freq – Holds the last successful scaling minimum frequency requested by KTM
  - cpus.freq_thresh_clear – If TRUE, emergency frequency mitigation threshold for this core is triggered in the hardware but not yet handled in KTM

KTM RAM Dump Debugging (cont.)

- When user_max_freq is high, for example, 4294967295, thermal-engine does not place any request for this core; it applies for user_min_freq if the value is 0.
- When user_max(min)_freq and limited_max(min)_freq are not the same, KTM is in the process of applying the new request or the freq_mitig kthread is waiting on mutex or blocked.
- The cpus_offlined variable is a bitmask for the current thermal hotplug request for CPUs. If the value of bit 1 is:
  - 1 – Thermal has a hotplug request for this core
  - 0 – Thermal has no hotplug request for this core
- When the request in cpus_offlined does not match cpus.offline and cpus.user_offline, the hot-plug thread waits for a mutex or blocked.
- When (cpus.threshold.trip == THERMAL_TRIP_CONFIGURABLE_HI && cpus.threshold.active == 1), the emergency threshold for this core is not reached. So the temperature of this core is less than the emergency threshold (cpus.threshold.temp).



FAQs

FAQs

Q. How do I find out the thermal configuration of a device?

A. Use the following commands:

```
adb root
adb remount
adb shell
#thermal-engine –o
```

Q. How do I understand and change the thermal configuration?

A. Retrieve the Readme file about thermal configuration at /vendor/qcom/proprietary/thermal-engine/readme.txt. After changing the existing thermal configuration, push it to /system/etc/xxxx.conf and restart the device.

`adb push thermal-config-xxxx.conf /system/etc/thermal-config-xxxx.conf`

Thermal engine is restarted by stop thermal-engine and start thermal-engine (*-d option* if thermal-engine debug information is required). Use *thermal-engine -o* to check if the changes have taken effect.

Q. How do I take a thermal sensor log?

A. TSENS logger is used to take the thermal sensor log for a given period.

```
adb push msm_tsens_logging  /data
adb shell
chmod 777 /data/msm_tsens_logging
./msm_tsens_logging  a b &
```

Here “a” is a sampling time and “b” is the duration of readings in milliseconds

Example – ./msm_tsens_logging 250 36000000 &
Here 250 is the sampling interval in milliseconds, 36000000 (10 hr) is the duration that the logger runs in milliseconds, and the “&” sign makes the logger run as a background process.
To verify that the TSENS logger is running from an ADB shell, run the command `cat /data/tsens_logger.csv`.
To get tsens_logger.csv from the device, run the command `adb pull /data/tsens_logger.csv.`

FAQs (cont.)

Q. How do I restart the thermal-engine with a debug log?

A. Use the following commands to restart the thermal-engine

```
stop thermal-engine
start thermal-engine –d
```

The following command outputs the thermal-engine log information in the command window (with time):

```
adb logcat -v time -s ThermalEngine
```

Q. How do I check if the thermal configuration is working?

A. To check whether the thermal configuration is working, use the TSENS log, thermal configuration file, and other logs like kernel and logcat. Check if various mitigations are taking place at thresholds fixed by the thermal configuration; for example, for CPU hotplug, the TSENS log can be checked to see if the hotplug is happening at the given threshold temperature.

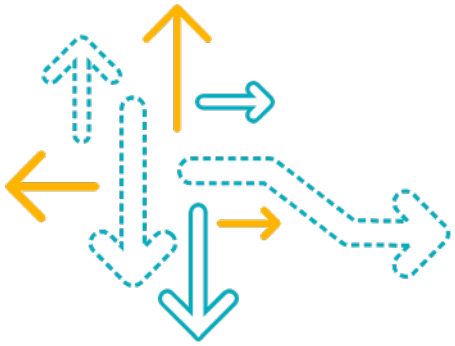
Q. What are documents that help in thermal design review/layout review procedure in early design?

A. Refer to the following documents:

- *Thermal Design Checklist* (80-VU794-21)
- *Design For Thermal: Key Requirements Why, What, Where, When, and How* (80-VU794-24)
- *Thermal Protection Algorithm Overview* (80-VT344-1)
- *MSM8974 Thermal Mitigation Algorithm* (80-N8633-6)

Q. How do I measure the skin temperature using IR camera with more accuracy?

A. Refer to *Skin temperature Measurement Procedure Using IR Camera* (80-VU794-15).



QTI Recommendations

Overview

- OEMs are advised to use the MSM internal temperature sensor for the thermal mitigation algorithm.
- For best performance and thermal stability, both thermal hardware design and proper thermal mitigations are crucial.
- Refer to *Design For Thermal: Key Requirements Why, What, Where, When, and How* (80-VU794-24)
- Incorporation of QTI's thermal mitigation algorithm is recommended.
- Consult QTI in advance if you plan to employ other thermal software solutions.
- Furthermore, it is imperative that the MSM junction temperature manages the on-die temperature sensors as they provide the resolution necessary for thermal software to react in a timely manner.
- More specifically, the junction temperature management including CPU temperature, the hottest points across most use cases, is required.
- And external sensor is used in addition to (not replacing) internal sensors to manage case or skin temperature.
- Bypassing the internal sensor or raising the TSENS threshold beyond the QTI-recommended value can cause the device to malfunction or even damage the MSM device.
- Refer to the following documents for:
  - Mechanical design – *Mobile Devices Hardware Thermal Management* (80-VU794-16) and *Coefficient of Thermal Spreading (CTS) - Figure of Merit for Mobile Thermal Management* (80-VU794-14)
  - Better thermal management – *Thermal Tuning Procedure* (80-N9649-1)

Suggestions to Control Camera Module Thermal Issues

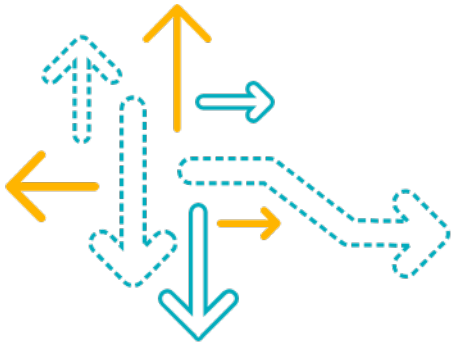
- In few devices, camera sensor can be hotspot for all camera use cases and could not be controlled by MSM thermal limiting, indicating it can be either camera sensor issue or heat spread at camera modules.
- OEMs are recommended to check:
  - Whether camera sensor is optimally configured and camera use case power numbers are as per chipset power numbers.
  - Heat spread stack up on phone is cooling the system enough for 1080p encode.
- Live shot size changed to HD 1080p to decrease power and thermal reduction significantly.
- The following are the two ways to test for decrease in live shot size:
  - adb root; adb shell setprop persist.camera.liveshot.size 1920x1080
  - Use the CMCC App to setprop the value instead of adb
- If any significant improvement is observed with reducing preview size, then set the default setting to 1080p.

References

Title	Number
Qualcomm Technologies, Inc.	
Thermal Design Checklist	80-VU794-21
Design For Thermal: Key Requirements Why, What, Where, When, and How	80-VU794-24
Thermal Protection Algorithm Overview	80-VT344-1
MSM8974 Thermal Mitigation Algorithm	80-N8633-6
Thermal Tuning Procedure	80-N9649-1
Skin Temperature Measurement Procedure Using IR Camera	80-VU794-15
Linux Android Software Thermal Debugging Guide	80-NM998-1
Mobile Devices Hardware Thermal Management	80-VU794-16
Coefficient of Thermal Spreading (CTS) - Figure of Merit for Mobile Thermal Management	80-VU794-14
Core Control Feature	80-P0106-1
Battery Current Limit (BCL) Overview and Tuning	80-NM328-709

References (cont.)

Acronym or term	Definition
ACK	Acknowledgment
BCL	Battery current limiting
BTM	Boot thermal management
CC	Carrier components
DL	Downlink
DTM	Dynamic thermal management
KTM	Kernel thermal monitor
MTPL	Maximum transmit power limit
NACK	No acknowledgment
OCP	Overcurrent protection
PMIC	Power management integrated circuit
PUCCH	Physical uplink common control channel
QICE	Qualcomm interference cancellation and equalization
SCC	Secondary component carrier
UE	User equipment
UL	Uplink
UVLO	Undervoltage lockout



Questions?

https://createpoint.qti.qualcomm.com

