MSM8974 Thermal Management Algorithm

80-N8633-6 D

August 19, 2013

Submit technical questions at:
https://support.cdmatech.com/

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

Contents

1 Introduction.. 6

1.1 Purpose.. 6
1.2 Scope... 6
1.3 Conventions 6
1.4 References... 7
1.5 Technical assistance.. 7
1.6 Acronyms.. 7

2 Purpose of Thermal Management.. 8

2.1 Overview of thermal issues... 8
2.2 Thermal management physical design and layout techniques 8
2.3 Thermal management algorithm goals.. 9
2.4 Thermal management algorithm prerequisites... 9

3 Thermal Software for Android™ 10

3.1 Overview.. 10
3.2 Architecture 10
3.3 Thermal sensors 12
3.3.1 Embedded thermal sensors 12
3.3.2 Thermistors.. 13
3.4 Boot mitigation 14
3.5 Kernel mitigation 15
3.6 Thermal engine 16
3.6.1 Architecture 16
3.6.2 Threshold control... 17
3.6.3 Dynamic control 18
3.6.4 Sensor manager.. 20
3.6.5 Device manager 21
3.7 Battery Current Limiting (BCL) 25
3.7.1 Architecture 25
3.8 Modem thermal mitigation 26
3.9 Common modem thermal mitigation NV settings 27
3.9.1 NV 947 – GPRS-enabled anite GCF 27
3.9.2 EFS file to change CFM clients... 27
3.9.3 LTE technology-specific NVs/EFS 27
3.9.4 NV 65675 – CFM monitor enable masks 27
3.9.5 NV 65611 – LTE FC MAC BSR target data rates.................................... 28
3.9.6 NV 65676 – CFM state remaining time.. 28
3.9.7 EFS file for Tx backoff.. 29

3.9.8 WCDMA/HSPA technology-specific NVs/EFS 30
3.9.9 1X/DO technology-specific NVs/EFS 30
3.10 LTE thermal management details 30
3.10.1 UL flow control and Tx backoff behavior 30
3.11 WCDMA/HSPA thermal management details 33
3.11.1 RLC data throttling and Tx backoff behavior 33
3.12 TD-SCDMA thermal management details 34
3.13 1X/DO thermal management details 34
3.14 Configuration for thermal engine 35
3.14.1 Thermal configuration file format 36
3.15 Debugging 46
3.15.1 Logging 46
3.16 Tuning tips for thermal management 47

A Thermal Configuration Code-Based Defaults 48

Figures

Figure 3-1 Thermal software architecture....10
Figure 3-2 MSM8974 TSENS sensor locations....12
Figure 3-3 Boot thermal mitigation....14
Figure 3-4 Thermal engine....16
Figure 3-5 Threshold algorithm example....17
Figure 3-6 Dynamic control example....18
Figure 3-7 BCL architecture....25
Figure 3-8 UL flow control....30
Figure 3-9 UL flow control....31
Figure 3-10 UE states – Mitigation 1....32
Figure 3-11 UE states – Mitigation 2....33
Figure 3-12 TDS maximum Tx power limiting....34

Tables

Table 1-1 Reference documents and standards....7
Table 3-1 Example threshold control settings....17
Table 3-2 Thermal management devices....21
Table 3-3 Battery management levels....22
Table 3-4 WLAN management levels....22
Table 3-5 Camcorder management levels....23
Table 3-6 Modem thermal mitigation feature set....26
Table 3-7 TD-SCDMA NV parameter NV 71523....34
Table 3-8 Algorithm instance fields....36
Table 3-9 Themal management devices....36
Table 3-10 Sensor names....38
Table 3-11 Monitor algorithm instance fields....39
Table 3-12 DTM algorithm instance fields....40
Table 3-13 PID algorithm instance fields....41
Table 3-14 Speaker coil calibration fields....42
Table 3-15 Virtual sensor fields....45

Revision history

Revision	Date	Description
A	Sep 2012	Initial release
B	Feb 2013	Numerous changes were made to this document; it should be read in its entirety.
C	Feb 2013	Updated Figure 3-2
D	Aug 2013	Updated with latest features available in thermal management

1 Introduction

1.1 Purpose

As Qualcomm Technologies, Inc. (QTI) chipsets continue to advance with heightened capabilities, performance, and concurrency, there is a need to have in place a thermal management framework to deal with high-power dissipation use cases. The thermal issue manifests in case temperatures of devices approaching or exceeding requirements of Underwriters Laboratories (UL) (and other worldwide regulatory bodies), carriers, and customers. The primary ways of dealing with the increased power requirements and associated heat are, first, efficient power management as described in [Q4] and [Q5], and second, mechanical designs that efficiently dissipate the heat created, as described in [Q2] and [Q3]. In addition, QTI provides thermal mitigation algorithms to deal with those instances where the temperature exceeds the desired operating levels. The software-based thermal management algorithm reduces the thermal load on the system by throttling performance of various features

This document describes the mechanisms necessary for customizing the thermal management algorithm on devices using supported Qualcomm ICs.

1.2 Scope

This document is intended for all engineers who use the MSM8974 configuration.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Shading indicates content that has been added or changed in this revision of the document.

1.4 References

Reference documents are listed in Table 1-1. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-1 Reference documents and standards

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Application Note: Thermal Design Considerations*	80-VU794-5
Q3	*Thermal Design Checklist*	80-VU794-21
Q4	*Presentation: Resource Power Manager (RPM) Overview and Debug*	80-VP169-1
Q5	*Presentation: MSM8x60/APQ8x60 Linux Power Management Debugging Guide*	80-VR629-2
Q6	*Application Note: MSM8974 v2 Clock Plan and Process Voltage Scaling*	80-NA157-154
Standards		
S1	*Information Technology Equipment – Safety*	IEC 60950-1:2005
S2	*Safety Specification*	UL1905

1.5 Technical assistance

For assistance or clarification on information in this document, submit a case to QTI at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support Service website, register for access or send email to support.cdmatech@qti.qualcomm.com.

1.6 Acronyms

For definitions of terms and abbreviations, see [Q1].

2 Purpose of Thermal Management

The thermal management algorithm provides a safety net, preventing the UE from causing damage to components or injury/discomfort to users. To accomplish this, power dissipation must be lowered to keep the temperature at acceptable levels. The algorithm carefully monitors key temperatures within the device and can be configured to adjust the performance capabilities of subsystems, thereby reducing power dissipation and heat generated.

Finding the appropriate thermal and performance tradeoff requires thermal calibration of the device. This is achieved by executing different combinations of scenarios on the device.

2.1 Overview of thermal issues

Present thermal analysis shows several potential causes driving the need to manage thermal:

- Market demand for feature-rich devices increases thermal susceptibility.
- High-power consumption use cases, e.g., SVLTE, SVDO, Wi-Fi hotspot, 3D gaming, HD video, etc., generate increased heat dissipation.
- Smaller/thinner designs lead to smaller PCBs and concentrated power density.
- Enabling higher UL data rates tends to cause higher Tx power, leading to more PA thermal dissipation.
- Touch temperature of the device's external casing may cause customer concerns; industry norms can limit smartphone external temperatures to 45°C (typically, ambient + 20°C) and external USB dongle temperatures to 75°C per [Q4] and [S1].

2.2 Thermal management physical design and layout techniques

To limit the impact of the thermal environment on the device, several techniques are suggested and recommended:

- Perform thermal analysis techniques during product design and development stages.
- Investigate device power and thermal budget from the analysis to locate projected hotspots.
- Implement efficient thermal management methods in the device mechanical structure.
- Ensure PAs reside in a location where heat is easily dissipated.
- Extensively use thermal vias along with thermal paths within the PCB, allowing for lower thermal resistance of the device.
- Add heat spreaders to remove heat from top side of thermally dense packages

- Ensure no other large power-dissipating components reside near or on the other side of the board from the PA and the MSM™ chipset.
- Use thermal dissipating techniques in the industrial design.

NOTE: See [Q2] and [Q3] for more information on thermal design.

2.3 Thermal management algorithm goals

Thermal management is targeted to:

- Protect components from exceeding thermal design limits
- Ensure compliance with external case and touch temperature requirements from customers, carriers, and standard organizations (UL, PCI Express), as well as user expectations
- Minimize the risk of power-limit constraints
- Manage the thermal risk and performance tradeoffs during concurrent operations
- Enable customer tailoring of response to thermal increases and methods for power reduction

2.4 Thermal management algorithm prerequisites

To properly manage the thermal scenarios of a device, the temperature must be measured from different thermally active points. This is done by sensors already present on the chipset and by placing thermistors at thermally active points, e.g., PA, PMIC, battery, etc. Currently, the thermal algorithm works only on PA thermistors reading.

Preferred sensor locations are listed below. Each of these sensors can usually be read by the Housekeeping ADC (HKADC) contained in QTI chipsets.

- Thermistor in close proximity to the PAs – The PA is the primary component contributing to thermal increase in a device, easily dissipating well over 1 W in poor RF conditions.
- Thermistor or internal sensors in MSM chipsets.
- Third-party components with potential for high power dissipation such as camera sensors.
- Other locations, e.g., battery, PMIC, etc.

QTI recommends placing thermistors near thermally active devices.

3 Thermal Software for Android™

3.1 Overview

Thermal management software is a configurable framework that provides the capability to control the temperature response of the entire system. The framework uses board-level thermistors and on-die temperature sensors to monitor the thermal response of the system. If a temperature is sensed above preconfigured thresholds the framework can direct subsystems to reduce performance in an effort to control heat dissipation. The framework provides the ability to configure which subsystem(s) respond to the temperature event and the level of the response.

3.2 Architecture

Figure 3-1 shows the four blocks of the thermal management framework: the thermal engine, sensor drivers, modem thermal manager, and other thermal management devices. In addition to these key components, there are management components addressing boot and kernel initialization.

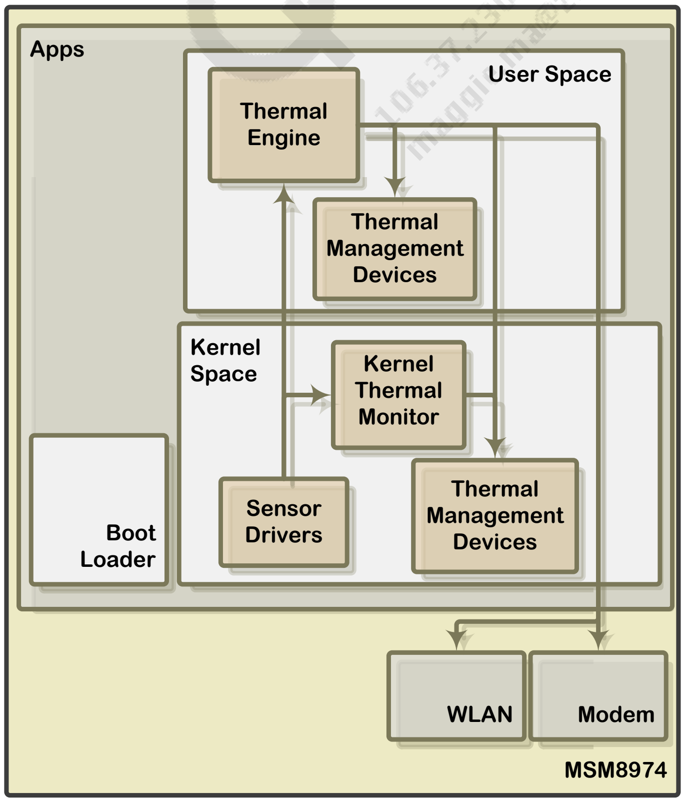

Figure 3-1 Thermal software architecture

The thermal engine runs as a super-user process in the Linux Android (LA) user space and is the central controller for thermal management. The thermal engine initializes the system on startup. The configuration of thresholds, set points, and management devices is read from defaults in the code and a configuration file in the Linux file system. The parameters are used to set interrupt thresholds in the hardware for temperature sensors. These parameters need to be tuned for each unique industrial design to achieve maximum performance levels while maintaining desired thermal specifications.

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

- Temperature sensors embedded in the MSM die which are connected to a hardware component called TSENS
- Temperature sensors placed on the PCB or within the PMIC which are connected through analog inputs (HKADC) in the PMIC. The PCB sensors in the reference design are thermistors placed close to the modem Power Amplifiers (PAs).

Thermal management devices are software components that control hardware with high-power densities, i.e., CPUs, GPUs, and PAs.

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

3.3 Thermal sensors

3.3.1 Embedded thermal sensors

Thermal sensors are embedded in the logic die at various high thermal density locations, i.e., CPU hotspots. Figure 3-2 shows the 11 on-die sensor locations on the MSM8974. The hardware block controlling the sensors, TSENS, includes one main sensor of the 11 sensors. The remaining 10 remote sensors are in other thermal hotspot locations and are connected to the TSENS block. Each sensor has a high and low threshold which can be set to trigger an interrupt. The hardware block polls each sensor in a round-robin fashion. The polling rate is configurable down to 62 ms per round per sensor. The hardware block is in the always-on power domain, so temperature sensors can be monitored at all times.

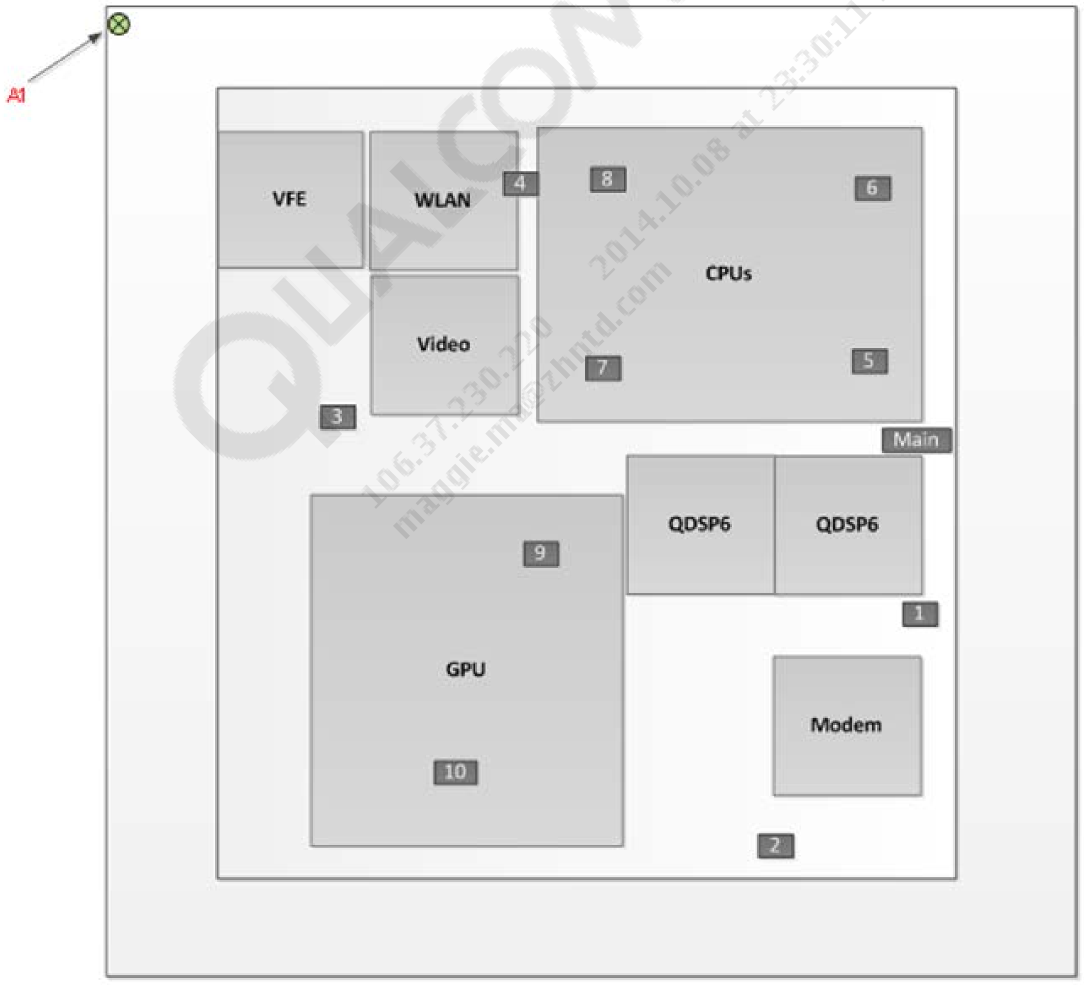

Figure 3-2 MSM8974 TSENS sensor locations

3.3.1.1 TSENS calibration

The physical sensors on the die are analog devices. To attain the highest possible accuracy, each sensor is calibrated. The calibration procedure is completed as part of chip manufacturing. The procedure stores the calibration data in the QFPROM (fuse blow) for each sensor.

3.3.1.2 TSENS kernel driver

TSENS hardware in the MSM8974 is controlled by a software driver running in the LA kernel space. The driver reads all calibration data from the QFPROM on initialization. The driver uses the data to convert ADC reads into temperatures reported to the thermal engine. The upper layers of the thermal engine interface with the driver to set the thresholds for interrupts and polling rates for the sensors.

3.3.2 Thermistors

The thermal engine monitors temperatures available from board-level thermistors. Typically, there are one or more thermistors near the PAs for the cellular RF. In addition to PA thermistors there are thermistors associated with the battery, PMIC, and XO. Other thermistors can be added if spare analog inputs on the PMIC are available in the design.

3.3.2.1 HKDAC

The analog inputs used for thermistors are monitored by the HKDAC block inside the PMIC. The HKADC module also supports high and low threshold interrupts for individual inputs.

3.3.2.2 HKADC kernel driver

The HKADC is controlled by a driver in the Linux kernel. The driver configures thresholds for each input passed down from the thermal engine and converts readings into temperatures which are passed up to the thermal engine. The conversion of ADC readings into temperatures uses tables specific to the thermistor used on the board. If the thermistor used in the design is different than the reference design thermistor, the table needs to be updated to get accurate temperature readings.

3.4 Boot mitigation

The boot thermal mitigation algorithm, illustrated in Figure 3-3, provides an initial temperature check of the die sensor embedded in CPU0 to ensure the device is within a required operational range for startup. The high temperature bound is set to ensure the device is operational until the next available mitigation algorithm (kernel thermal driver) can monitor and adjust performance. The low temperature bound is set to ensure the device maintains timing closure at the operational voltages set in the boot code.

If one of the limits is not met, the boot code attempts to bring the device into the operational temperature range by waiting in a loop. If the maximum attempts are reached before temperature is within the normal range, the system will shut down.

Configuration of the temperature limits and the delays must be hard coded in the SBL since no file system support exists when the algorithm executes. The algorithm is contained in /core/boot/secboot3/src/boot_thermal_management.c.

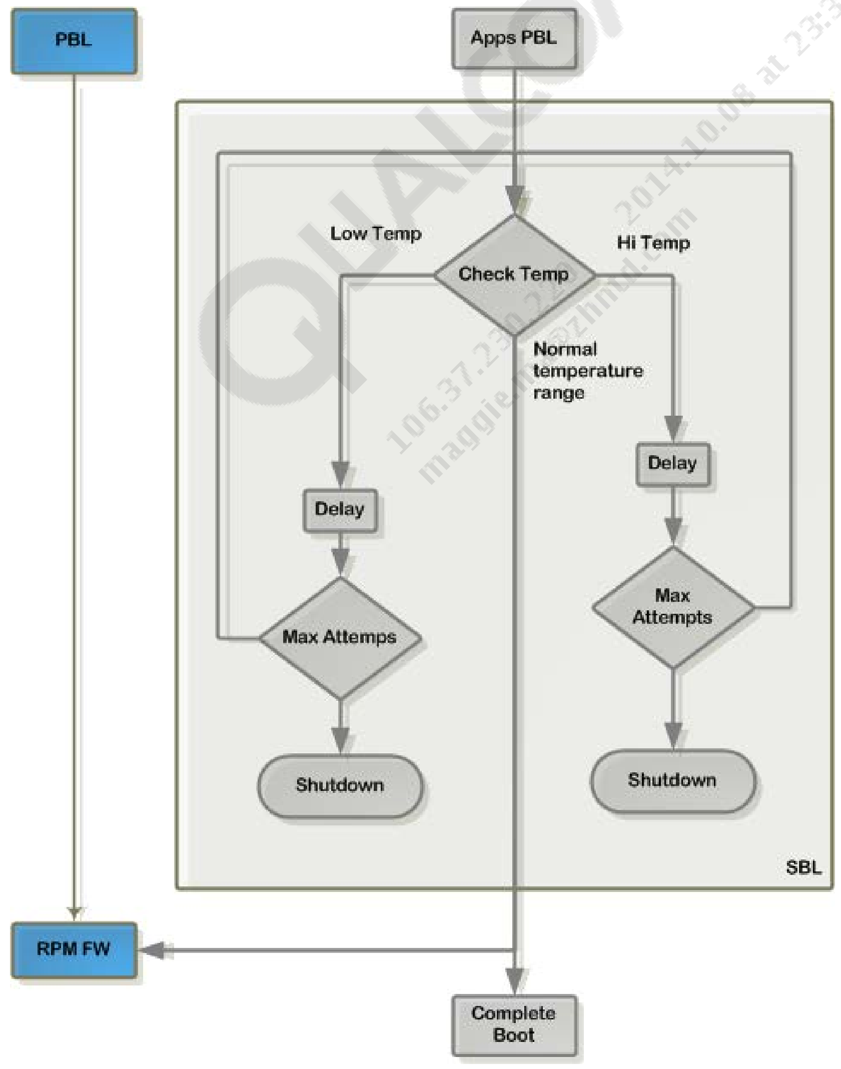

Figure 3-3 Boot thermal mitigation

3.5 Kernel mitigation

NOTE: Numerous changes were made in this section.

Kernel mitigation provides thermal management during kernel initialization prior to the initialization of the main thermal management algorithm which requires services, i.e., the file system, to be operational. The algorithm controls the core frequencies and power state of cores. The parameters for configuration of the algorithm are part of the device tree initialization for msm-thermal shown below.

```
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>;
                …
      };
```

An embedded temperature sensor used to control the algorithm is defined as CPU0 sensor (sensor 5) in the sensor-ID field. If the temperature exceeds the specified level in limit-temp, the maximum allowed CPU frequency will be reduced. The reduction continues while temperature is above the limit at each polling interval. The polling interval is defined in the poll-ms field. If the temperature drops below the sum of the limit-temp plus the temp-hysteresis, then the maximum allowed CPU frequency will be increased. The changes in CPU frequency up or down are one step in the DCVS table frequencies per sample period.

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

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

The algorithm is contained in /drivers/thermal/msm_thermal.c and the associated parameters are defined in arch/arm/boot/dts/msm8974.dtsi.

3.6 Thermal engine

3.6.1 Architecture

The thermal engine (Figure 3-4) is the main thermal monitor running during device operation. It resides in the Linux user space as a daemon process and performs two main functions:

- Monitoring temperature
- Controlling management devices

Two components provide these functions:

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

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

- Threshold control
- Dynamic control

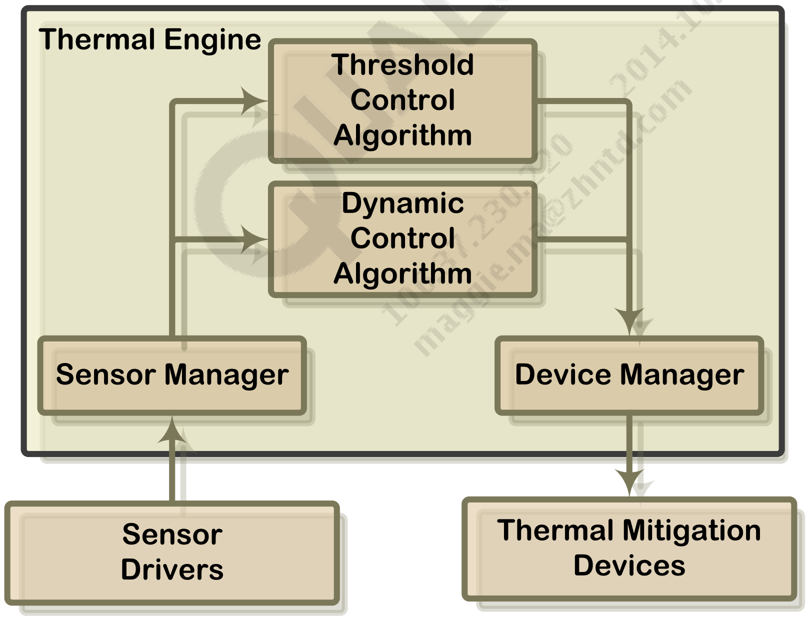
Thermal Engine
Threshold Control Algorithm
Dynamic Control Algorithm
Sensor Manager
Device Manager
Sensor Drivers
Thermal Mitigation Devices

Figure 3-4 Thermal engine

3.6.2 Threshold control

NOTE: Numerous changes were made in this section.

The threshold control algorithm performs mitigation based on preconfigured set points. When a temperature threshold is reached, a management device is set to a predetermined performance level. The threshold/performance pairs are preconfigured and stored in a configuration file or hard-coded. Details for configuring the threshold algorithm are explained in Section 3.14. Example threshold control settings for a GPU are shown in Table 3-1 and in Figure 3-5.

Table 3-1 Example threshold control settings

Threshold number	Threshold	Mitigation level
1	70°C	333 MHz
2	77°C	200 MHz
3	85°C	100 MHz

In this example, when the temperature reaches 70°C, the GPU maximum allowable frequency is capped to 333 MHz. This 333 MHz cap is not enough to control the temperature rise. As the temperature rises to 77°C, a second threshold is reached and the GPU maximum frequency is capped at 200 MHz. Again, the temperature continues to rise until the third threshold at 85°C is reached, then GPU maximum frequency is capped at 100 MHz. At this point the temperature stabilizes.

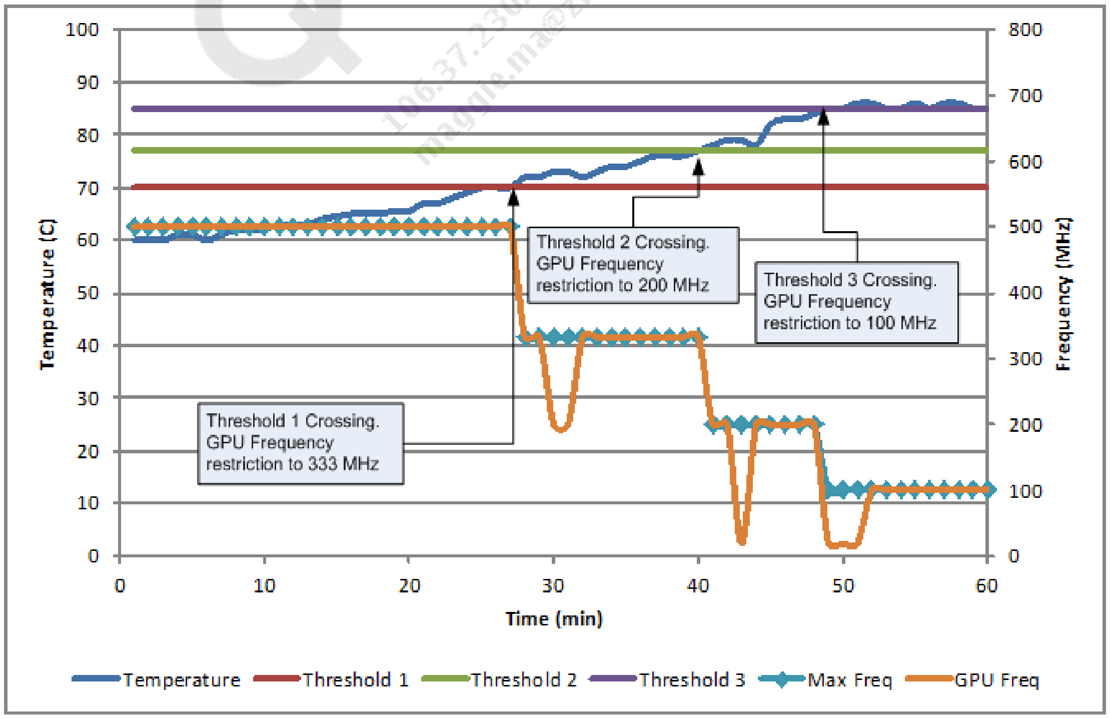

Figure 3-5 Threshold algorithm example

3.6.3 Dynamic control

The dynamic algorithm adjusts performance based on the difference between a sensor measurement and a desired set point temperature. The algorithm only controls GPU and CPU max frequencies. If the measured temperature is above the set point, the algorithm steps the maximum allowed frequency down. The algorithm continues to monitor the temperature and adjust the frequency until the measured temperature is at or below the set point. Details for configuring the dynamic algorithm are explained in Section 3.14.

In the example shown in Figure 3-6, the CPU temperature set point is 75°C. In this example, the device is capable of a maximum CPU frequency of 1.5 GHz. Note that the actual maximum performance of the commercial device will be different. While the temperature measured by the controlling sensor is below 75°C, there is no restriction placed on the maximum allowed CPU frequency. Some 37 sec into the example, the temperature crosses the set point. The maximum allowed CPU frequency starts to step down one step at a time through the possible DCVS frequencies. The timing of the steps is set by the sampling time in the configuration file (see Section 3.14.1). Once the temperature returns to the set point, the adjustments to the maximum frequency stop as there is no difference between measured temperature and set point temperature. As the temperature falls below 75°C, the maximum allowed frequency is allowed to increase. There is also a clearing set point that places the sensor monitoring back into a Wait-for-Interrupt mode instead of polling the sensor at the sampling rate. The clearing set point is not shown in this example.

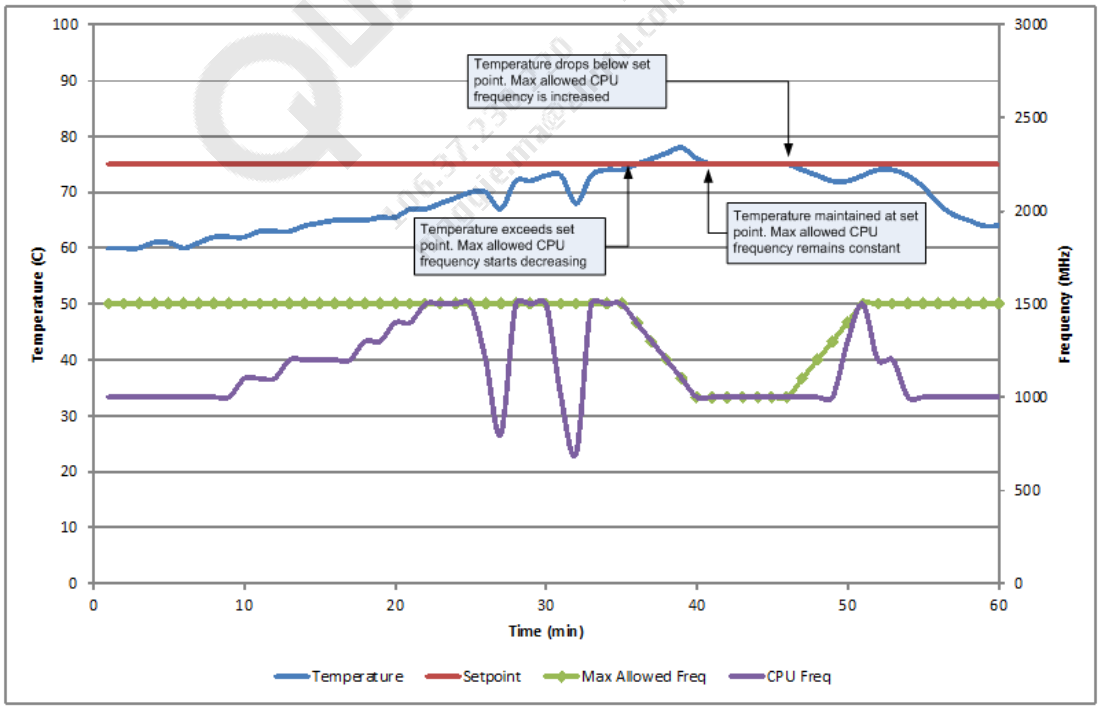

Figure 3-6 Dynamic control example

The dynamic control algorithm can be configured to operate in two modes. The first mode is called Dynamic Thermal Management (DTM). DTM is used by default in the reference software to control the CPU frequency. The second mode uses the classical control theory Proportional, Integral, Derivative (PID)-based algorithm to maintain the set point temperature.

3.6.3.1 DTM

NOTE: Numerous changes were made in this section.

When used to control the CPU cluster, DTM adjusts the maximum allowed CPU frequency up and down based on a temperature set point. Each adjustment in the maximum allowed CPU frequency is based on the available frequency steps from the DCVS frequency table. If the measured temperature is above the set point, then the maximum allowed CPU frequency is reduced by one step. Likewise, if the temperature is below the set point and there is a restriction in place, the maximum allowed CPU frequency is increased by one step. This algorithm can also control maximum allowed GPU frequency.

3.6.3.2 PID control

The typical PID controller consists of an error function derived from the system parameters and a controller output evaluated using the error function. The error function for CPU thermal management is derived from the on-die temperature sensors and a configurable set point temperature threshold. If T(k) is the measured temperature at time k and T_{th} is the set point temperature threshold, then the error function e(k) used at the time k is the difference of the temperatures as given by the equation:

$$e(k) = T_{th} - T(k)$$

The PID controller computes the controller output, q(k), at time instant k based on the above error function using the equation:

$$q(k) = K_p.e(k) + K_i.T_s.\sum_{i=k-n}^{k} e(i) + \frac{K_d}{T_s}.[e(k) - e(k-1)]$$

The controller output consists of three components: the proportional (P), the integral (I), and the derivative (D). Each of the components is scaled by the parameters Kp, Ki, and Kd. The values of the three parameters must be chosen based on the dynamics of the system, either through experiments or simulations. In the above equation, Ts is the sampling period and n is the duration over which the error samples are summed. The sampling duration can be chosen based on the rate at which the temperature of a component is expected to change. The controller output depends on the current error value, the historic n error values, and the rate of change of the error.

The PID controller output suggests an appropriate adjustment based on whether the current temperature is above or below the set point threshold, T_{th}. The controller output can be translated into the maximum allowed CPU frequency.

3.6.3.2.1 Integral windup

The integral component in the PID equation tends to accumulate large errors, which results in overshooting or undershooting control. This is commonly known as integral windup. To overcome integral windup, the control equation is modified to use a first derivative of the controller. This velocity form of the PID controller is given by:

$$q(k) = q(k-1) + K_p.[e(k) - e(k-1)] + K_i.T_s.e(k) + \frac{K_d}{T_s}.[e(k) - 2e(k-1) + e(k-2)]$$

The velocity PID equation replaces the proportional term by the first difference, the integral term by the current error value, and the derivative term by the second difference of the error. Furthermore, the controller output at the previous time, instant k-1, is added to the current output. Because the controller output contains the output value from the previous time instant, the output can be nonzero even after reaching the intended set point threshold. Note that a nonzero controller output means that an action is applied to CPU management. This can result in oscillations in the controller output around the set point. To avoid the oscillations, the previous output value is excluded every time the set point is reached.

3.6.3.2.2 Choosing the PID constants

As mentioned, the PID constants, i.e., K_p, K_i, and K_d, must be chosen based on the system dynamics. The default constants used for CPU thermal management were tuned based on the Zeigler-Nichols (Z-N) method. Experiments and simulations show that adjustments to the default constants have very little impact on performance. Therefore, it is recommended not to change the default values.

3.6.4 Sensor manager

The sensor manager monitors sensor inputs from on-die temperature sensors, board-level thermistors, and current measurement systems. During initialization, the sensor manager configures the hardware to generate interrupts for temperature events. The levels are determined based on the settings in the thermal configuration file. When a hardware interrupt occurs, the sensor manager alerts the algorithms and sets up the next threshold according to the configuration file.

3.6.4.1 Virtual sensors

NOTE: This section was added to this document revision.

Virtual sensors provide the ability to combine inputs from several sensors into a single sensor to drive mitigation actions. The virtual sensor output is a weighted average of the input sensors. An example virtual sensor configuration is shown in 3.14.1.7.3.

3.6.5 Device manager

The device manager handles communication with all of the available thermal management devices. The thermal management devices provide cooling to the overall system by reducing power consumption through reduced performance. Each subsystem has a different method for reducing overall power consumption as explained in the following sections. The available devices are listed in Table 3-2.

Table 3-2 Thermal management devices

Device	Description
CPU	Sets the maximum allowable CPU frequency of one or all CPU cores or power collapses a CPU core
GPU	Sets the maximum allowable GPU frequency
Modem	Reduces data throughput, restricts maximum Tx power, and sets modem in Limited Service mode (E911)
Battery charging	Restricts maximum charge rate
WLAN	Reduces throughput or sets WLAN in Idle Mode Power Save (IMPS)
Camcorder	Restricts the FPS of the encoded video or shutdowns recording
LCD	Reduces the LCD backlight brightness
Voltage restriction	Restricts the minimum allowed voltage on the common digital rail
Shutdown	Shutdowns the device

3.6.5.1 CPU management device

CPU thermal management reduces the maximum allowed frequency of CPU cores. It does not override the DCVS setting, but rather restricts the maximum allowed frequency the DCVS can operate.

The threshold algorithm sets the maximum allowed frequency for all CPUs. The available levels for the maximum allowed frequencies are any of the DCVS frequencies. Multiple thresholds can be configured to allow the maximum allowed frequency to step down to lower levels as temperatures increase.

The dynamic algorithm can restrict the maximum allowed frequency of individual CPUs and/or all CPUs at once. Selection of the maximum allowed frequency is set algorithmically by the dynamic control algorithm. Only a temperature set point needs to be selected.

CPUs can also be power collapsed at a configurable temperature threshold.

3.6.5.2 GPU management device

GPU management sets the maximum allowable GPU frequency based on the configured temperature thresholds. The frequency steps available are the complete set from the GPU DCVS list. See [Q6] for details.

3.6.5.3 Modem management device

Modem management supports four levels for thermal adjustment:

- Level 0 – No restriction, full modem performance
- Level 1 – Requests that the modem run the data throughput reduction algorithms
- Level 2 – Requests that the modem run the Tx power reduction algorithms
- Level 3 – Puts the modem into Limited Service mode, in which only E911 calls are allowed

Section 3.8 provides a detailed explanation of these levels and associated.

3.6.5.4 Battery charging management device

Battery charging management restricts the maximum allowed battery charge current. Table 3-3 lists the five available management levels.

Table 3-3 Battery management levels

Level	Maximum charge rate (mA)
0	No restriction
1	1100
2	700
3	600
4	325
5	0

3.6.5.5 WLAN management device

WLAN management adjusts the allowed data throughput. The levels supported for WLAN are shown in Table 3-4. In addition to setting the desired levels, a flag in the WLAN configuration file must be set for WLAN thermal management to take effect. The flag can be found in the /data/misc/wifi/WCNSS_qcom_cfg.ini file. The gThermalMitigationEnable flag must be set to 1 to enable WLAN management.

Table 3-4 WLAN management levels

Management level	Description
0	Normal operation – No mitigation
1	Disables aggregation
2	Controls flow by sleeping for 100 ms if the Tx frame count is larger than 16 during the previous 300 ms time frame, duty cycle is updated to 3/4, VoIP call will work without any voice chopping; disables aggregation.
3	Controls flow by sleeping for 500 ms if the Tx frame count is larger than 10 during the previous 500 ms time frame, duty cycle is updated to 1/2, VoIP call might be disconnect, slower best effort traffic; disables aggregation
4	Forces IMPS

3.6.5.6 Camcorder management device

Camcorder management adjusts the fps of the recorded video or completely shuts down the camcorder. Table 3-5 lists the supported camcorder management levels. There are two options for restricting the fps. Either employ frame skipping at the output of the video front end, or adjust the camera sensor. The advantage of using frame skipping is that the camcorder pipeline does not need to be stopped to make the adjustment, so no glitch will be recorded; however, the power savings are not as much.

Table 3-5 Camcorder management levels

Level	Description	fps
0	No mitigation	NA
1	Frame skip level 1	24
2	Frame skip level 2	15
3	Sensor restriction level 1	24
4	Sensor restriction level 2	15
5	Shut down camcorder	0

3.6.5.7 LCD management device

LCD management adjusts the maximum allowed backlight brightness. The brightness level can be set from the brightest at 255 to off at 0. To prevent continuous adjustment of the display brightness in response to temperature fluctuations, the dynamic control algorithm is not available for LCD management.

3.6.5.8 Shutdown management device

Shutdown management allows the thermal management system to shut down the device if temperatures exceed the configured threshold. This typically occurs only in very high ambient conditions. The threshold for this should be set in order to protect the device's hardware from damage.

3.6.5.9 Override

NOTE: This section was added to this document revision.

Override control is used to adjust the thermal thresholds or set points used in any configuration instance. The override occurs when the thermal engine receives an override control flag from the multiprocessor control daemon. See Section 3.14.1.7 for an example of override control.

3.6.5.10 Speaker coil calibration

NOTE: This section was added to this document revision.

The speaker coil calibration device is used in conjunction with the feedback speaker protection feature available in the codec. The codec requires an accurate value for a speaker coil resistance at a known temperature to enable the best performance of the feedback protection algorithm. This management device provides the ability for the audio subsystem to acquire an accurate temperature estimation for the speaker coil. There is a default configuration for the speaker coil calibration device used to generate an accurate estimation of coil temperature for the QTI reference platform. Tuning may be required to achieve the best results for speaker coil temperature estimation on an OEM platform. The default configuration for speaker coil calibration is explained in Section 3.14.1.6.

3.6.5.11 Voltage restriction management device

NOTE: Numerous changes were made in this section.

Voltage restriction is implemented to ensure silicon timing constraints are met. This feature limits the minimum allowed voltage when die temperatures are below 0°C. The limits for this feature are set in code and should not be adjusted.

3.7 Battery Current Limiting (BCL)

3.7.1 Architecture

NOTE: Numerous changes were made in this section.

Typically, the MSM8974 chipset is used in mobile devices powered by a single cell battery. Single cell batteries have a limited current supply capability that, in some rare instances, may be below the power consumption requirements of the MSM8974 chipset. This greatly depends on a number of factors, such as type of battery used, the battery state of charge, battery temperature, etc. If the system concurrencies are such that current draw exceeds the battery capability, there will be a voltage dip which may result in a device reset. To prevent this device reset, the battery current limiting feature provides a method for limiting peak currents. The feature is based on the Battery Monitoring System (BMS). The BMS provides key parametric data on the state of the battery, i.e., average current consumption, battery open circuit voltage, and battery resistance. The data is accessed by the BCL to monitor the average current capabilities of the battery and limit performance if capability is exceeded. The limiting is done through the thermal engine. The output from BCL appears as another sensor input into the thermal framework.

Figure 3-7 illustrates the BCL architecture.

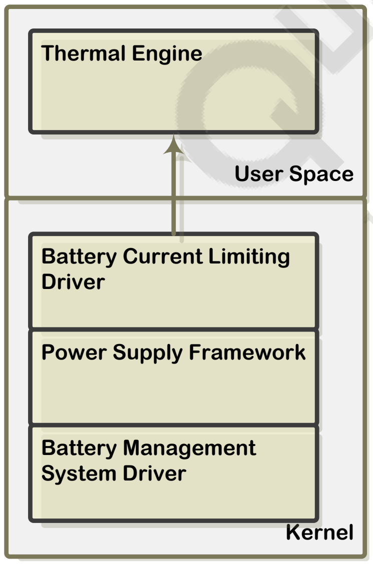
Thermal Engine
User Space
Battery Current Limiting Driver
Power Supply Framework
Battery Management System Driver
Kernel

Figure 3-7 BCL architecture

3.8 Modem thermal mitigation

In addition to the thermal engine running in the application core, there are additional algorithms that run in the modem to manage thermal response for each radio technology. The modem algorithms have additional configuration parameters defined in modem EFS. Modem thermal management consists of three distinct mitigation levels:

- UL data throughput keeping the original power class reduces the requirement of higher Tx power to close the reverse link and DL data throughput eases the power/MIPS requirements of the modem processor
- Maximum Tx power limiting reduces the PA Tx power, similar to SAR reduction
- Limited Service mode restricts all data calls

Table 3-6 Modem thermal mitigation feature set

Feature	Radio technology				Notes
	LTE	WCDMA/ HSPA	DO	1X	
Thermal Emergency state; call teardown supervision and E911 allowance	Supported	Supported	Supported	Supported	
UL data throttling	Supported	Supported	Supported	Supported	
DL data throttling	NA	Supported	Supported	Supported	
Interference cancellation disabling	NA	NA	NA	NA	
Rx diversity disabling	NA	NA	NA	NA	
Tx power scaling – Duty-cycle back off	Supported	Supported	Supported	Supported	Primarily used when data throttling is not effective or possible at edge of cell

3.9 Common modem thermal mitigation NV settings

The following sections describe NV items applicable to all radio technologies. For thermal mitigation to work on the modem, the following prerequisites must be present:

- A test SIM, i.e., SIM with MCC in range of 1 to12, should *not* be used. A test SIM will enable a GCF test flag resulting in disabled thermal mitigation.
- Data throttling requires the network/test box to support dynamic scheduling.

3.9.1 NV 947 – GPRS-enabled anite GCF

This NV is required to enable/disable the GCF Testing mode in the device. The NV is set as:

NV 947 – <GCF_flag>

If GCF Testing mode is enabled, i.e., the GCF flag is set, the thermal mitigation algorithm is disabled; it is required to disable the GCF flag by setting NV 947 to 0.

3.9.2 EFS file to change CFM clients

Currently, if the thermald config file only has modem mitigation levels 2 and 3 defined, with 1 not defined, then for LTE RAT, data throttling, i.e., a fake MAC_BSR, also begins as part of level 2 (tx_back_off). To enable tx_back_off modem mitigation, without data throttling, in level 2 use the following EFS file:

- EFS location – /nv/item_files/modem/utils/cfm/
- EFS file name – cfm_client_enabled_masks
- Hex content – FFFFFFFE

3.9.3 LTE technology-specific NVs/EFS

This information will be provided in a future revision of this document.

3.9.4 NV 65675 – CFM monitor enable masks

- By default (if the NV item is not written), all monitors including the thermal monitor are enabled (one still needs to configure the DEM thermal threshold, as described in Section 3.3).
- To disable the thermal mitigation for LTE, set the 2 LSBs of the mask to 0, i.e., 0xFFFFFFFC.
- To revert to the default mask, remove the EFS file /nv/item_files/modem/utils/cfm/ cfm_monitor_enabled_masks using QPST/EFS explorer.
- The only supported values are 0xFFFFFFFC (disable LTE) and 0xFFFFFFFF (enable all monitors in all modes); all other values are not supported and may cause unknown behavior.

3.9.5 NV 65611 – LTE FC MAC BSR target data rates

By default (if the NV item is not written), the default target MAC-level data rate configuration for UL data throttling is:

```
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 Mbps)
Uint32 target_rate[1] = 5000 (40 Mbps)
Uint32 target_rate[2] = 3125 (25 Mbps)
Uint32 target_rate[3] = 1250 (10 Mbps)
Uint32 target_rate[4] = 625 (5 Mbps)
Uint32 target_rate[5] = 125 (1 Mbps)
Uint32 target_rate[6] = 63 (500 kbps)
Uint32 target_rate[7] = 13 (100 kbps)
Uint32 target_rate[8] = 6 (50 kbps)
Uint32 target_rate[9] = 1 (10 kbps)
```

To change the data rate configuration, write to num_state, default_state, target_rate[0]…target_rate[num_state-1]; the data rate is expressed in bytes per ms.

To revert to the default configuration, remove the file /nv/item_files/modem/lte/common/lte_fc_macul_target_rates using QPST/EFS Explorer.

For flow control target data rates (NV 65611), the default value is 10 states, as described above. The default rate is currently set to 1 Mbps. The minimum rate can be set to a lower value, but the recommendation is to set it to a value that would meet the requirement for control channels and delay-sensitive applications. Setting this to 0 is not recommended, as it would shut down the UL and could cause the call to eventually drop. The default state should be chosen carefully to guarantee a timely response that reduces the temperature. By setting the default state to 5 in the default configuration, the data rate is initially reduced to 1 Mbps after UL flow control is triggered.

3.9.6 NV 65676 – CFM state remaining time

By default (if the NV item is not written), the thermal step timer is set to 15 sec.

The thermal step timer can be changed from 0 to 255 sec by writing to the NV item.

To revert to the default step timer, remove the EFS file /nv/item_files/modem/utils/cfm/cfm_thermal_step_timer_in_sec using QPST/EFS Explorer.

This timer controls how fast the UL data rate is increased or decreased (depending on the state). It is currently set to 15 sec based on the observations from test results. This setting should correspond to the time it takes for the temperature sensors to converge to a new temperature value.

3.9.7 EFS file for Tx backoff

For LTE, when the threshold for Tx backoff is crossed (NV 6242), the PA power is adjusted per the parameters configured in the EFS file. tx_power_backoff located in /nv/item_files/modem/lte/ML1/.

- P_backoff – Initial value for Tx power backoff in dB
  - Recommended initial value – 5 dB
  - At each step, n, the value of power backoff is *n x P_backoff*
  - n is increased by one upon each expiration of step_timer
- T_on – The length of time when the UE removes the limit on MTPL
  - Recommended initial value – 50 ms
- T_off – The length of time when the UE reduces MTPL
  - Recommended initial value – 50 ms
- Step_timer – Time spent in each step (see P_backoff)
  - Recommended initial value – 15 sec
- P_min – The minimum value for MTPL
  - Recommended initial value – 10 dBm

The structure of the EFS file in order is:

```

/* 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;

```

An example of the file's hex content is:

05000C00 32003200 983A0000

- 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 sec (983A for 15 sec)

3.9.8 WCDMA/HSPA technology-specific NVs/EFS

None

3.9.9 1X/DO technology-specific NVs/EFS

None

3.10 LTE thermal management details

3.10.1 UL flow control and Tx backoff behavior

UL flow control consists of:

- Temperature monitor (currently only PA temperature is used to trigger uplink data throttling)
- Centralized flow control manager
- LTE MAC BSR control
- Tx power backoff

Figure 3-8 shows the UL flow control.

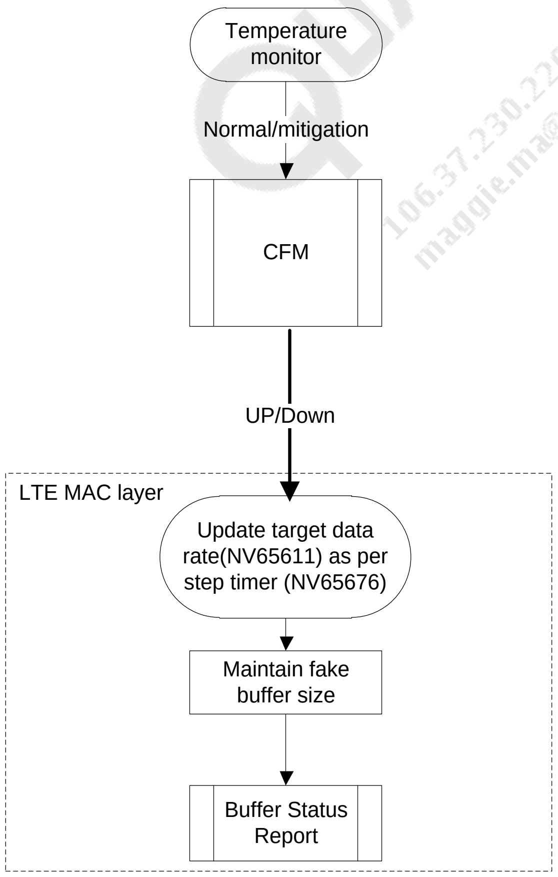
Temperature monitor
Normal/mitigation
CFM
UP/Down
LTE MAC layer
Update target data rate(NV65611) as per step timer (NV65676)
Maintain fake buffer size
Buffer Status Report

Figure 3-8 UL flow control

In this section, the UL throughput in each flow control state is reviewed. The flow control state is derived from the input of one or more thermal sensor subsystems and potentially other resources. Since the focus is on the thermal sensor, only the transitions related to the PA temperature monitor are described.

As shown in Figure 3-9, UL flow control can be in any of four states: Normal, MAC_BRS, Tx backoff, or Shutdown.

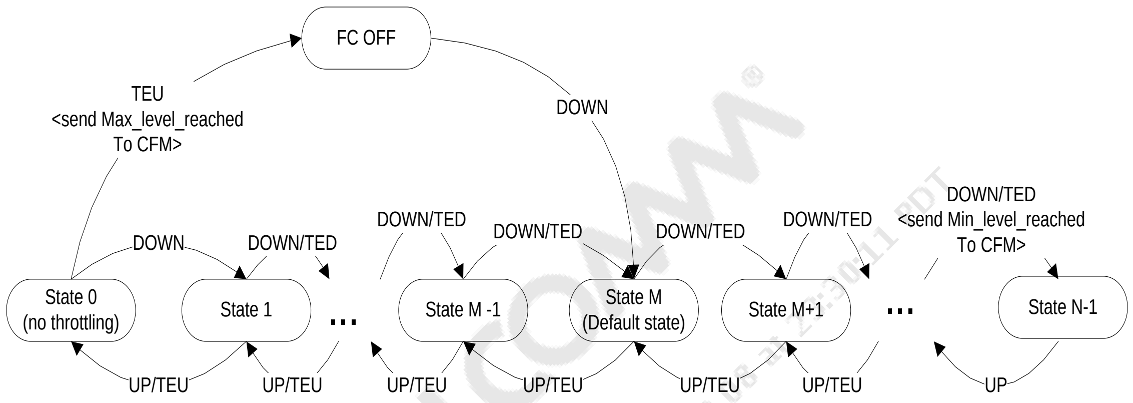

Figure 3-9 UL flow control

As shown in Figure 3-10 and Figure 3-11, the UE's flow control can be categorized in four different states depending on its thermal state.

- Normal state (FC_OFF) – This is the default state of the UE. In this state, the UE is below the configured temperature threshold (PA mitigation threshold), and the UE operates in complete accordance with 3GPP specifications. Once the configured threshold is crossed, the UE moves to the Flow Control (FC_ON) state.
- MAC BSR state – Mitigation 1
    - Upon entering this state from the FC_OFF state, if LTE is up and the MAC_BSR client is registered with the CFM, a DOWN command is sent to the MAC_BSR client.
    - Upon entering this state from the Tx Backoff state, if LTE is up and the Tx_backoff client is registered with the CFM, an UP command is sent to the Tx_backoff client.
    - While in the MAC_BSR state:
        - If a normal indication is received from the temperature monitor, and
            - If the MAC_BSR client is registered with the CFM, an UP command is sent to the MAC_BSR client.
            - If the MAC_BSR client is not registered with the CFM, then the UE transitions to the FC_OFF state.
        - If a mitigation indication is received from the temperature monitor, and if the MAC_BSR client is registered, a DOWN command is sent to the client.
    - If the MAC_BSR client registers with CFM while CFM is in this state, the MAC_BSR client receives a DOWN command immediately after registration.

- Upon receiving the Tx backoff indication from the temperature monitor, the UE transitions to the Tx_backoff state.
- Upon receiving an emergency indication from the temperature monitor, the UE transitions to the Shutdown state.

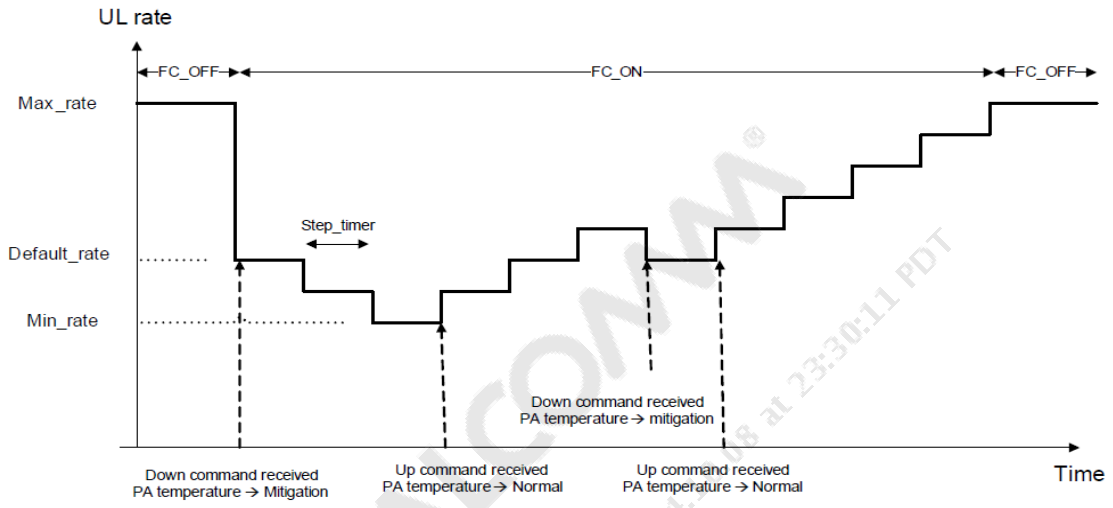

Figure 3-10 UE states – Mitigation 1

- Tx_backoff state – Mitigation 2
  - Upon entering the Tx_backoff state from the Shutdown state, it is assumed that LTE is not active in the Shutdown state, so no action is necessary when transitioning to this state from the Shutdown state.
  - Upon entering Tx_backoff from the MAC_BSR state, if LTE is up and the Tx_backoff client is registered with CFM, a DOWN command is sent to the Tx_backoff client.
  - While in the Tx_backoff state:
    - If a mitigation indication is received from the temperature monitor, and if the Tx_backoff client is registered with CFM, an UP command is sent to the Tx_backoff client; CFM transitions to the MAC_BSR state.
    - If a normal indication is received from the temperature monitor, and
      - If the MAC_BSR client is registered, CFM transitions to the MAC_BSR state.
      - If the MAC_BSR client is not registered, CFM transitions to the FC_OFF state.
    - If the MAC_BSR client registers with CFM while CFM is in this state, the MAC_BSR client receives a DOWN command immediately after registration.
    - If the Tx backoff client registers with CFM while CFM is in this state, the Tx_backoff client receives a DOWN command immediately after registration.
    - Upon receiving an emergency indication from the temperature monitor, the UE transitions to the Shutdown state.

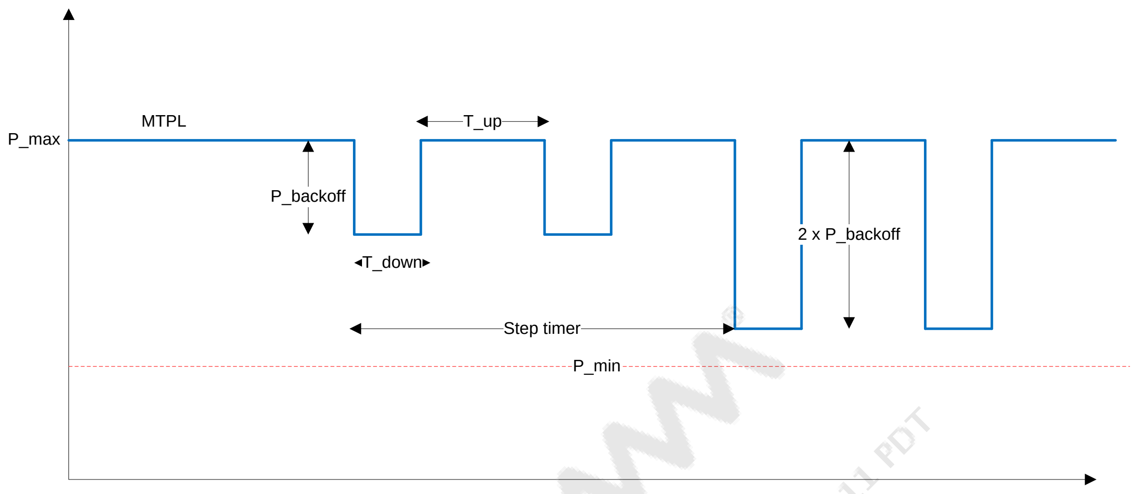

Figure 3-11 UE states – Mitigation 2

- Shutdown – Emergency state
  - The behavior in the Emergency state is to drop the RRC connection; the PA is not shutdown in this state.

3.11 WCDMA/HSPA thermal management details

3.11.1 RLC data throttling and Tx backoff behavior

The objective of RLC flow control is to reduce power consumption associated with HSPA, since processing higher data rates leads to increased power consumption.

In the DL direction, based on the commands from the DEM module, the RLC flow control function controls the peer side RLC (UTRAN) through Windows commands when entered into the Mitigation state. A sufficient decrease in the RLC window size, i.e., closing the window, decreases the amount of data sent to the UE and the processing required. The window size is decreased gradually in steps whenever the periodic timer expires in the Mitigation state. If the device enters the Emergency state, the RLC window size with value 1 is sent to the peer entity in order to go to the lowest possible data throttling. Conversely, if the basic state is entered, the data flow can be increased by increasing the window size, i.e., opening the window.

A similar process of window size adjustment is used in the UL direction. In this case, since the transmitting entity is located on the UE itself, the window size can be directly controlled without the need for sending any commands over the air to the peer side RLC.

Tx power backoff happens upon entering the Mitigation 2 state by:

- Initializing step counter and MTPL backoff value
- Calling RF API to reduce the MTPL by an initial backoff value
- Triggering duty-cycle and step timers

Upon exiting the Mitigation 2 state, the UE calls the RF API to restore MTPL to a normal value before backoff.

3.12 TD-SCDMA thermal management details

NOTE: This section was added to this document revision.

TDS has one NV 71523 to control the MTPL backoff feature, as illustrated in Figure 3-12 and described in Table 3-7.

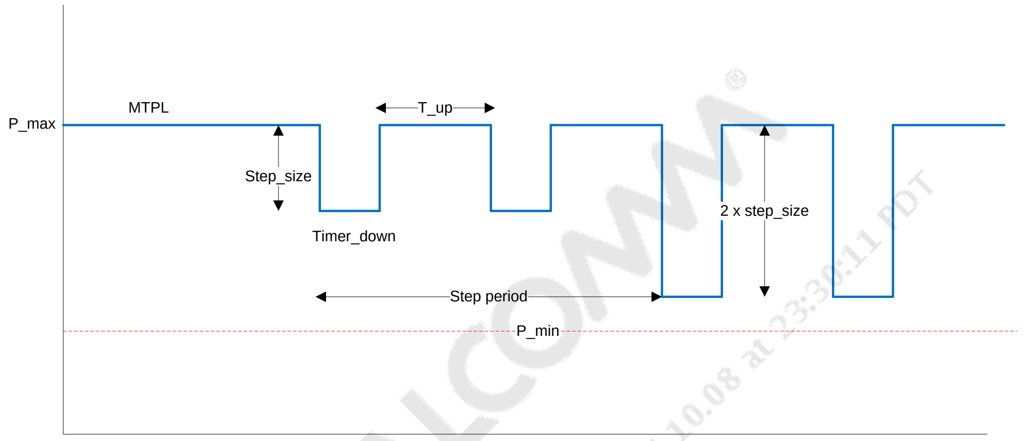

Figure 3-12 TDS maximum Tx power limiting

Table 3-7 TD-SCDMA NV parameter NV 71523

Type	Name	Default	Units	Description
uint16	step_size	5	dB	MTPL reduced power step size
uint16	step_period	10000	ms	Period to change the MTPL reduced power step
uint16	Timer_up	50	ms	Timer counting when MTPL at normal level
uint16	Timer_down	400	ms	Timer counting when MTPL at reduced level
uint16	Max_reduce_mtpl	255	dB	Maximum reduction of MTPL

3.13 1X/DO thermal management details

The thermal sensor for the power amplifier subsystem is integrated with the Congestion Control Manager (CCM). The CCM impacts reverse link flow control only in DO/1X operation. The CCM reduces the RL data rate so that the heat generated from the PA can be reduced to an acceptable level to maintain the maximum possible data rate in the system.

If PA sensor mitigation is triggered, the RL limits the maximum allowed payload size, or packet size, on certain carriers at the Reverse Link MAC (RMAC) layer in the DO protocol stack. The RMAC performs QoS-based packet prioritization. By allowing a certain packet size, QoS may be maintained. If PA sensor critical is triggered on DO, RMAC on DO sets the maximum payload size of all active carriers to 0 except SLP. The SLP carrier is kept at the default maximum payload size.

If PA sensor mitigation/critical is triggered on 1X, the device shall stop requesting R-SCH, which stops the reverse link traffic on 1X. When the PA sensor Normal state is triggered, the device shall resume R-SCH processing.

3.14 Configuration for thermal engine

NOTE: Numerous changes were made in this section.

Configuration of thermal engine consists of selecting the algorithms, the management devices, the temperature levels and the response.

Each thermal management device listed in Section 3.6.5 can be set up to reduce the overall temperature of the system. The temperature and amount of contribution is set in the thermal configuration file. One approach to controlling a management device is to use the input from a sensor directly monitoring the thermal response of the management device, e.g., sensor ID 5 is typically used to control CPU0 as it is located at the hot spot of CPU0, as shown in Figure 3-2. A second approach is to use a sensor that is impacted by the overall thermal response of the system and is indirectly impacted by the management device. Both approaches are valid and are used for different reasons. The CPU management direct control case, i.e., sensor ID 5 controlling CPU0, would typically be used to ensure the silicon junction temperatures do not exceed specified limits. Alternatively, indirect control of the CPU cluster would better manage the overall skin temperature of the system. A sensor which is not impacted by the high power cores or an external board-level thermistor would be better suited to indirect control, i.e., sensor ID 3. To determine the threshold or set point for indirect control, determine the offset temperature between the sensor and the hotspot of the skin using the sensor measurements and measurements from the thermocouples placed at the skin hotspot.

The best performance for the system is achieved by having multiple overlapping thermal controls configured, as the thermal response of the system will be different based on the use case, e.g., a use case with a purely CPU intensive workload will have a different thermal response than a GPU-bound use case. To maintain thermal specification within tolerances, a configuration for the GPU sensors to control thermal response for the graphics use case is needed in addition to a configuration for the CPU intensive case. The thermal configuration file allows multiple thermal configurations to be set. It is even possible to have different sensors control the same thermal management device. This is useful for thermal management devices that add to the overall thermal issue, i.e., the LCD or battery charging.

Configurations for any thermal management device can be done by adding default algorithm instances in the code and recompiling. This feature allows licensees to move the entire thermal configuration file into the compiled code to better hide thermal management configuration from users. Appendix A lists the default management instances. Any instance defined in the code can be overridden by simply adding an instance with the same label (see Table 3-8) and adding any fields with new values under that label. It is also possible to turn off instances defined in the code by using the disable field.

3.14.1 Thermal configuration file format

The configuration file consists of sections identifying the configuration for an instance of a thermal algorithm. Each section defines the necessary parameters to control one instance of one of the thermal algorithms. Prior to the algorithm sections, there are two optional fields. The first is the debug field. If a line is added to the configuration file with the word debug, the thermal engine runs in debug mode, which produces more detailed diagnostic messages in the log files. The second optional field is the default sampling rate used by any of the algorithm instances unless the instance specifically defines the sampling rate in the instance. Table 3-8 defines the fields common to all algorithm instances. Some fields are optional.

Table 3-8 Algorithm instance fields

Field name	Values	Units	Description
Instance label	Unique identifier	Text	Unique identifier with maximum length of 31 characters
algo_type	pid, ss, monitor	Text	
Disable	Disable	Text	Optional field used to disable/override default configurations built into code
Sensor	See Section 3.14.1.2	Text	tsens_tz_sensor0 through tsens_tz_sensor10, bcl, PMIC_THERM, PA_THERM0, PA_THERM1
Override	Temperature	Deg C	Optional field adds the temperature to all set points or thresholds defined in the instance when the override flag is received from multiprocessor control daemon
Descending	Descending	Text	Optional field used to set triggering when temperature falls below threshold instead of the default rise above threshold triggering
Sampling	60 to 65535	ms	Time between sampling intervals
Algo type specific	See Sections 3.14.1.3 to 3.14.1.5	—	—

3.14.1.1 Thermal management devices

Each algorithm instance is used to configure one or more management devices. Table 3-9 lists all of the available management devices that can be used in configuration instances. Possible mitigation levels for the devices are listed.

Table 3-9 Themal management devices

Action	Description	Comment
cpuN	Sets the maximum allowable frequency for individual CPUs	▪ N – 0 to 3, indicating the CPU number
Cpu	Sets the maximum allowable frequency for all CPUs at the same level	—
Gpu	Sets the maximum allowable frequency for GPU	—
Wlan	Sends a mitigation request to WLAN	See Table 3-4

Action	Description	Comment
Lcd	Sets the backlight intensity; will not adjust the maximum value over the currently configured brightness set by the user	Values 0 to 255 ▪ 0 – Backlight off ▪ 255 – Maximum brightness
Modem	Sends mitigation request to the modem on standalone design	Levels ▪ 0 – No mitigation ▪ 1 – Data throttling ▪ 2 – Tx power back off ▪ 3 – Limited Service mode; only E911 calls allowed
Battery	Sends mitigation request to battery charging	See Table 3-3
Camcorder	Sends mitigation request to the camera subsystem to reduce the fps running on the camera pipeline	—
Shutdown	Sends a shutdown request to the kernel with a delay to power down the phone	Delay, in ms
hotplug_N	Power collapse one of the CPU cores	N is 1 to 3, indicating which CPU to hotplug; note that CPU 0 cannot be hotplugged
Fusion	Sends mitigation request to the modem on fusion design	Same as modem but refers to APQ8064+MDM9615-based modem; levels are: ▪ 0 – No mitigation ▪ 1 – Data throttling ▪ 2 – Tx power back off ▪ 3 – Limited Service mode; only E911 calls allowed
Bcl	Sets thresholds for current consumption management	—
Report		The threshold crossing information is sent across an abstract local socket THERMALD_UI in new line-separated string format. The report action runs upon clearing or triggering a level, slightly different from the other actions which are run upon reaching the level, e.g., clearing level n+1 or triggering level n. Parameters are sent in the following order: ▪ sensorname – Name of the sensor that is reporting ▪ temperature – Current temperature ▪ current_threshold_level – Current threshold level triggered or cleared ▪ is_trigger – True on level trigger; false on level clearing

3.14.1.2 Sensor names

The names listed in Table 3-10 are all of the available sensors in QTI reference platforms. Some of the sensors have aliases defined in the code that can be used instead of the full names.

Table 3-10 Sensor names

Name	Comment
tsens_tz_sensor0	This sensor is in the main block in Figure 3-2
tsens_tz_sensor1	—
tsens_tz_sensor2	—
tsens_tz_sensor3	Aliased in the code as pop_mem. On QTI reference platforms, this sensor closely tracks the pop memory case temperature. This should be validated on OEM device.
tsens_tz_sensor4	—
tsens_tz_sensor5	Aliased in the code as cpu0
tsens_tz_sensor6	Aliased in the code as cpu1
tsens_tz_sensor7	Aliased in the code as cpu2
tsens_tz_sensor8	Aliased in the code as cpu3
tsens_tz_sensor9	—
tsens_tz_sensor10	—
Pmic_therm	—
pa_therm0	GSM/UMTS/LTE PA
pa_therm1	LTE PA – Applicable to only specific RF platforms

3.14.1.3 Monitor algorithm field information

The additional fields for configuring a monitor instance are listed in Table 3-11. There are some specific rules that must be followed for monitor instances listed here:

- Monitor instance fields accept a list of space separated values for up to 8 thresholds.
- Actions can be a list of management devices; in other words, for each of the 8 possible thresholds there can be a list of actions and the corresponding action_info.
- The actions field lists the mitigation devices acted upon at each threshold; the parser for the configuration file is not complex; therefore, these fields are not completely free form.
  - The rules for actions and actions_info within a threshold group must be separated by only the + character and no spaces.
- Actions must be explicitly stated for each threshold; device settings are not implicitly reset to previous thresholds, except upon clearing the lowest threshold.
- Each action must have a corresponding action_info.
- The action_info must also be separated by only the + character and no spaces.

An example monitor configuration is shown in Section 3.14.1.3.1.

Table 3-11 Monitor algorithm instance fields

Field name	Values	Units	Description
Thresholds	-273000 to 273000	°mC	Threshold to activate
thresholds_clr	-273000 to 273000	°mC	Threshold to clear
Actions	See Table 3-9	—	Action to act upon at thresholds; multiple entries allowed are separated by + with no spaces; this should be same for all threshold levels
action_info	See Table 3-9	—	Information used by action; multiple entries allowed are separated by + with no spaces

3.14.1.3.1 Monitor algorithm instance example

This example has the unique label lcd_gpu_management. The monitor algorithm (algo_type = monitor) controls the temperature by sampling sensor ID 3 (sensor = tsens_tz_sensor3) once per second (sampling = 1000). Sensor ID 3 is by the display subsystem, as shown in Figure 3-2, where the thermal response of the sensor was determined to be offset from the systems surface temperature by 25°C; thus, thresholds are configured to reduce the GPU and LCD performance as the sensor approaches 70°C, thereby maintaining the surface temperature below 45°C. The first threshold group is configured at 50°C. In this group, the GPU is restricted to 500 MHz and the LCD is configured to 255. These are the maximum settings for both management devices. This threshold group starts the sampling process for this sensor at a rate of 1 Hz to enable log messages for temperature. The second threshold group is configured at 65°C. In this group, the GPU is restricted to 333 MHz and the LCD backlight is set to 150. The threshold group clears these settings back to the settings in the first threshold group when the temperature reaches 60°C.

```
[lcd_gpu_management]
algo_type         monitor
sensor            tsens_tz_sensor3
sampling          1000
thresholds        50000      65000
thresholds_clr    45000      60000
actions           gpu+lcd    gpu+lcd
action_info       500000+255 333000+150
```

3.14.1.3.2 Monitor algorithm modem control instance

This example defines a modem control instance using a thermistor as the sensor. The monitor algorithm (algo_type = monitor) controls the temperature by sampling the PA thermistor 0 (sensor = pa_therm0) once per second (sampling = 1000). The PA thermistor 0 was determined to be offset from the systems surface temperature by 30°C; thus, thresholds are configured to reduce the modem throughput and maximum allowed Tx power. The first threshold group is configured at 70°C. In this group, the modem data throughput reduction algorithm is triggered (action_info = 1). The second threshold group is configured at 80°C. In this group, the modem Tx power reduction algorithm is triggered (action_info = 2).

```
[modem]
algo_type        monitor
sensor           pa_therm0
sampling         1000
thresholds       70000    80000
thresholds_clr   65000    75000
actions          modem    modem
action_info      1         2
```

3.14.1.4 DTM algorithm field information

DTM is the recommended control algorithm for CPU and GPU.

Table 3-12 DTM algorithm instance fields

Field name	Values	Units	Description
Device	—	—	
set_point	-273000 to 273000	°mC	Threshold to activate
set_point_clr	-273000 to 273000	°mC	Threshold to clear
Time_constant	—	—	Multiplier of sampling period for hold off adjustments

3.14.1.4.1 DTM algorithm instance example

In this example, the label is surface_control_dtm. This name must be unique from other instance labels. The DTM algorithm (algo_type = ss) controls the temperature by sampling sensor ID 3 (sensor = tsens_tz_sensor3) once per second (sampling = 1000). The DTM control adjusts the maximum allowed CPU frequency of all CPUs together (device = cpu). Sensor ID 3 is by the display subsystem, as shown in Figure 3-2, where the thermal response of the sensor was determined to be offset from the systems surface temperature by 25°C; thus, the controlling set point to maintain the surface temperature at 45°C is set at 70°C. The clearing set point is set at 55°C, which is the temperature where the DTM controller will stop adjusting the maximum allowed frequency and let it return to maximum.

```
[surface_control_dtm]
algo_type          ss
sensor             tsens_tz_sensor3
device             cpu
sampling           1000
set_point          70000
set_point_clr      55000
```

3.14.1.5 PID algorithm field information

The PID algorithm can also be used for CPU and GPU control.

Table 3-13 PID algorithm instance fields

Field name	Values	Units	Description
Device	—	—	Only valid for CPU or cpu_all
set_point	-273000 to 273000	°mC	Threshold to activate
set_point_clr	-273000 to 273000	°mC	Threshold to clear
p_const	—	—	Proportional constant
i_const	—	—	Integral constant
d_const	—	—	Derivative constant
i_samples	—	—	Number of integral samples
dev_units_per_calc	—	—	Number of units to adjust based on the output of the algorithm

3.14.1.5.1 PID algorithm instance example

In this example, the label is surface_control_pid. This name must be unique from other instance labels. The PID algorithm (algo_type = pid) controls the temperature by sampling sensor ID 5 (sensor = tsens_tz_sensor5) at 65 ms (sampling = 65). The PID control adjusts the maximum allowed CPU frequency of CPU0 (device = cpu0). Sensor ID 5 is at the hotspot of CPU0, as shown in Figure 3-2; thus, the controlling set point maintains the CPU0 junction temperature at 95°C. The clearing set point is set at 55°C, which is the temperature where the PID controller will stop adjusting the maximum allowed frequency based on this sensor and let it return to maximum.

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

3.14.1.6 Speaker coil calibration field information

The speaker coil calibration instance uses a unique set of fields to define the configuration. The instance is defined in the default configuration included in the code with a label SPEAKER-CAL.

Table 3-14 Speaker coil calibration fields

Field name	Subfield name	Values	Units	Description
sampling	Stage 1		ms	Stage 1 sampling.
	Stage 2		ms	Long sampling period
	samples			Number of long samples ±2
	delay	1800000	ms	Delay from bootup prior to starting calibration procedure; 30 minute default
sensor		-273000 to 273000	°mC	
sensors		-273000 to 273000	°mC	
temp_range	spread	-273000 to 273000	°mC	Sensors list temp spread
	delta		°mC	Single sensor ±delta from max sensor in list
	long delta		°mC	Long sample delta allowed
max_temp		-273000 to 273000	°mC	
offset		-273000 to 273000	°mC	

The speaker coil calibration feature estimates the temperature of the speaker coil based on the available temperature sensors in the system. On the QTI reference design, the die sensors and external thermistors can be used in the estimation. The best estimate is obtained when the device is in a state of thermal equilibrium—no activity is adding heat to any part of the system and the device temperature is not changing. In other words, the device would be in equilibrium if it was powered off or is in an idle state for a long period of time. Once the device is in equilibrium, estimation of the speaker coil temperature is a matter of knowing the offset from other temperature sensors.

To determine if the device is in equilibrium, the thermal engine uses the following criteria based on the default configuration instance:

1. Wait for 30 min (sampling.delay field) after initialization.
2. All die sensors (listed in the sensors field) must be within 6ºC (temp_range.spread field) and must be below 45ºC (max_temp). If not, then check again in 30 sec (sampling.Stage 1).
3. All die sensors (listed in sensors field) must be within 10ºC (temp_range.delta field) of the external sensor (sensor field).
4. All die sensors (listed in sensors field) must be within 2ºC of each other (temp_range.long_delta) for 10 samples (sampling.samples) sampled at 30 sec (sampling. Stage 2).
5. If these conditions all pass then the estimated speaker coil temperature will be the lowest reading from the sensors (listed in the sensors field) plus -4ºC (offset).

3.14.1.6.1 Speaker coil calibration default

Speaker coil calibration is configured by default in the code with the label SPEAKER-CAL. Note that the sensors line needs to be one contiguous line without any returns in the configuration file. CPU0 and CPU1 are not included in the list, as they are impacted by power consumed to run the algorithm. Also, tsens_tz_sensor0 is located close to these CPUs and is impacted.

```
[SPEAKER-CAL]
sampling    30000 30000 10 1800000
sensor      pm8941_tz
sensors     tsens_tz_sensor1 tsens_tz_sensor2 tsens_tz_sensor3
            tsens_tz_sensor4 tsens_tz_sensor7 tsens_tz_sensor8
            tsens_tz_sensor9 tsens_tz_sensor10
temp_range  6000 10000 2000
max_temp    45000
offset      -4000
```

3.14.1.7 Override control field information

Override control allows for runtime adjustments to the thermal set points and thresholds based on the multiprocessor control daemon override flag.

3.14.1.7.1 Override control example with DTM

In this example, DTM is controlling all CPUs to 70ºC. If the multiprocessor control daemon sets the override flag for the thermal engine, the DTM control set point for this instance would become 80ºC and the clearing set point would be 65ºC.

```
[surface_control_dtm]
algo_type          ss
sensor             tsens_tz_sensor3
override           10000
device             cpu
sampling           1000
set_point          70000
set_point_clr      55000
```

3.14.1.7.2 Override control example with monitor

In this example, a monitor algorithm is controlling battery charging. If the multiprocessor control daemon sets the override flag for the thermal engine, the monitor thresholds and threshold clear temperatures would all be reduced by 10ºC.

```
[battery_charging_control]
algo_type        monitor
sensor           tsens_tz_sensor3
override         -10000
sampling         1000
thresholds       70000    80000
thresholds_clr   65000    75000
actions          battery  battery
action_info      3         4
```

3.14.1.7.3 Virtual sensor

A virtual sensor provides a temperature based on the weighted average of several physical sensor inputs.

Table 3-15 Virtual sensor fields

Field name	Values	Units	Description
Name	Text		Unique name for the virtual sensor
Algo type	Virtual		Defines the instance type to be a virtual sensor configuration
trip_sensor			Sensor which triggers the virtual sensor to start generating temperature output
set_point		°mC	Trip sensor temperature at which to start generating temperature output
set_point_clr		°mC	Trip sensor temperature at which to stop generating temperature output
sensors			Array of sensors to calculate weighted temperature sum
weights			Array of weights associated with sensors array
sampling		ms	Rate for the data to be generated from the virtual sensor

3.14.1.7.4 Virtual sensor configuration example

```
[Virtual_skin_temperature]
algo_type          virtual
trip_sensor        tsens_tz_sensor3
set_point          60000
set_point_clr      50000
sensors            tsens_tz_sensor0, tsens_tz_sensor3, tsens_tz_sensor6
weights            30, 60, 10
sampling           5000
```

3.14.1.8 Editing configuration file

To edit the configuration file, execute the following commands while connected to the device through a USB:

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

3.15 Debugging

The current debugging methodology is:

1. FAQ
2. Logs available:
   a. Logcat – Format, message meaning, key messages, etc.
   b. Kernel logs
   c. QXDM logs
3. Tools – Benchmark apps, etc.
4. ADB commands
5. Test data
   a. Examples of all TMDs
   b. Dashboard

3.15.1 Logging

The thermal engine has minimal logging, unless it is started in Debug mode or a debug flag is in the first line of the configuration file. Since the thermal engine runs at startup, to turn on full logging, the daemon needs to be stopped and restarted.

```
adb shell stop thermal-engine
adb shell
# /sys/bin/thermal-engine—debug &
```

The following command outputs the thermal engine log information in the command window:

```
adb logcat -s ThermalEngine
```

Timestamps can also be output by adding the –v time option:

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

3.16 Tuning tips for thermal management

NOTE: Numerous changes were made in this section.

- Pay attention to Cx cutoffs for turbo for GPU and CPU.
- Configure both GPU and CPU mitigation. GPU mitigation helps to achieve lower skin temperature for the same performance level in many graphics use cases.
- Configure battery charging and LCD brightness. Reducing power consumed in charging and LCD backlight typically has lower user experience impact than reducing the CPU performance.
- Pay attention to modem use cases. High data throughput modem use cases can generate significant heat buildup in the area of the power amplifiers.
- Configure junction temperature sensors for all cores. Very aggressive benchmark tests can result in the device reaching junction temperature limits before skin temperature limits. The framework can easily be configured to keep both limits within specifications.
- Configure the main skin control loop to operate on a quit sensor. CPU sensors are *not* good candidates to control device skin temperature due to very fast thermal rise in reaction to CPU loads.
- Configure hotplug controls for CPU 1 to 3.
- Use DTM control for CPU and GPU control. DTM control has been shown to provide the highest performance capability under high loads and reduce temperature swings due to thermal limiting.

A Thermal Configuration Code-Based Defaults

The default management devices configured in the code are:

```
[PID-CPU0]
 algo_type          pid
 sensor             cpu0
 device             cpu0
 sampling           1000
 set_point          95000
 set_point_clr      75000
 p_const            2.0
 i_const            0.5
 d_const            1.0
 i_samples          10
dev_units_per_calc 5000
[PID-CPU1]
 algo_type          pid
 sensor             cpu1
 device             cpu1
 sampling           1000
 set_point          95000
 set_point_clr      75000
 p_const            2.0
 i_const            0.5
 d_const            1.0
 i_samples          10
dev_units_per_calc 5000
[PID-CPU2]
 algo_type          pid
 sensor             cpu2
 device             cpu2
 sampling           1000
 set_point          95000
 set_point_clr      75000
 p_const            2.0
 i_const           0.5
 d_const           1.0
 i_samples         10
dev_units_per_calc 5000
[PID-CPU3]
 algo_type         pid
 sensor            cpu3
 device            cpu3
 sampling          1000
 set_point         95000
 set_point_clr     75000
 p_const           2.0
 i_const           0.5
 d_const           1.0
 i_samples         10
dev_units_per_calc 5000
[PID-POPMEM]
 algo_type         pid
 sensor            pop_mem
 device            cpu
 sampling          1000
 set_point         95000
 set_point_clr     75000
 p_const           2.0
 i_const           0.5
 d_const           1.0
 i_samples         10
dev_units_per_calc 5000
[HOTPLUG-CPU1]
 algo_type        monitor
 sensor           cpu1
 sampling         1000
 thresholds       115000
 thresholds_clr   95000
 actions          hotplug_1
 action_info      1
```

```
[HOTPLUG-CPU2]
 algo_type        monitor
 sensor           cpu2
 sampling         1000
 thresholds       115000
 thresholds_clr   95000
 actions          hotplug_2
 action_info      1

[HOTPLUG-CPU3]
 algo_type        monitor
 sensor           cpu3
 sampling         1000
 thresholds       115000
 thresholds_clr   95000
 actions          hotplug_3
 action_info      1
```